Auton Robot
DOI 10.1007/s10514-016-9561-5

Vision-based and IMU-aided scale factor-free linear velocity estimator

Rafik Mebarki[1] · **Vincenzo Lippiello**[1] · **Bruno Siciliano**[1]

Received: 20 May 2015 / Accepted: 17 March 2016

Abstract This paper presents a new linear velocity estimator based on the unscented Kalman filter and making use of image information aided with inertial measurements. The proposed technique is independent of the scale factor in case of planar observed scene and does not require a priori knowledge of the scene. Image moments of virtual objects, i.e. sets of classical image features such as corners collected online, are employed as the sole correcting information to be fed back to the estimator. Experimental results performed with a quadrotor equipped with a fisheye camera highlight the potential of the proposed approach.

Keywords UAV quadrotors · Velocity estimation · Computer vision · Data fusion · Kalman filter

1 Introduction

Nowadays, the development of unmanned aerial vehicles (UAVs) is reaching remarkable proportions. In particular, vertical take-off and landing (VTOL) UAVs, such as rotary-wing vehicles, possess the hovering capability which makes them appealing if not unique candidates for numerous potential applications, in both civil and industrial scenarios, as well as for both outdoors and indoors missions.

Linear velocity measurement is an essential information for the control system of UAVs (Castillo et al. 2004). For instance, in a visual servoing scenario (Espiau et al. 1992), knowledge of the linear velocity will suffice to position the vehicle to a desired location. Although the linear velocity could be inferred from global positioning system (GPS)'s position measurements, GPS is an active sensing modality relying on an external source (satellite) of information. GPS is therefore unreliable at low altitudes, in urban and indoor areas, suffers from signal cut (Prasad et al. 2008), has its performance affected by weather conditions, and is typically available at low frequency (1 Hz).

Inertial measurement units (IMU) provide attitude and acceleration measurements at relatively high frequency, which yields an important sensing modality for the UAV attitude control. Nevertheless, the measure of the acceleration is relayed with a bias and large noise, and thus numerical integration thereof to obtain the linear velocity leads to drifts quickly growing over time, especially for low-cost IMUs.

On the other hand, vision relays wealthy information with pixel-order resolution, involves low power consumption, and is passive. Moreover, modern technology affords cameras with increased streaming rate and resolution, yet with lighter weight and lower energy consumption. These characteristics make cameras as a good sensing modality for UAVs applications.

The onboard linear-velocity estimation problem triggered different research works, leveraging vision and IMU for the purpose. In Shen et al. (2011), a simultaneous localization and mapping (SLAM) fuses IMU and laser range sensor data with an extended Kalman filter (EKF) to obtain quadrotor pose and velocity.

This paper is an extended version of (Mebarki and Lippiello 2014) that received the Best Paper Award of the 12th IEEE International Symposium on Safety, Security, and Rescue Robotics held in Hokkaido, Japan.

✉ Rafik Mebarki
rafik.mebarki@unina.it

Vincenzo Lippiello
vincenzo.lippiello@unina.it

Bruno Siciliano
bruno.siciliano@unina.it

[1] PRISMA Lab, Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione, Università degli Studi di Napoli Federico II, Via Claudio 21, 80125 Naples, Italy

Published online: 21 April 2016

In Achtelik et al. (2011), an EKF is used to fuse vision, IMU, and air pressure data in a SLAM framework for pose estimation. It is based on a linear model of the vehicle motions, thus not benefiting from larger reactivity and domain of convergence of a nonlinear model. In Kneip et al. (2011) the linear velocity is provided as a closed-form solution from three consecutive measures of IMU and vision. A similar solution is presented in Lippiello and Mebarki (2013), but spherical visual features are adopted, while in Mebarki and Siciliano (2013), Mebarki et al. (2015) an image-based nonlinear observer is proposed.

A large number of research works focused on optical flow (Ma et al. 2003). In Grabe et al. (2012), by assuming the scene to be planar, the *continuous homography constraint* (Ma et al. 2003) is used and the linear velocity is provided as a closed-form solution, involving the angular velocity from the IMU. In Weiss et al. (2012), Honegger et al. (2013), optical flow and IMU data are fused instead. In Zhao et al. (2015), optical flow along with barometer measures is exploited in an EKF to estimate the velocity, but the reliability of using a barometer in indoor environments is still an issue. In Mourikis and Roumeliotis (2007), an EKF fuses IMU data with a set of visual features observed in a number of successive image sequences for state estimation. It is assumed that a prior estimate of the 3D positions of the visual features with respect to a global frame is first extracted or known. The EKF update measurements consist of the image coordinates of the visual points but also combined to those 3D position approximations. A similar algorithm is applied in Mourikis et al. (2009) for spacecraft planetary landing application.

In this work image moments are employed as the sole feedback correcting-measurements for velocity estimation. However, classical surface image moments imply that contrasted (physical) objects be in the scene and their image sections be properly segmented. In our previous work (Mebarki and Lippiello 2014), synthetic and engineered objects of circular shape and black color have been introduced into the scene to cope with the issue. The solution we propose in this paper overcomes this limitation such that it only needs that natural feature points be present in the image. The proposed algorithm fits virtual objects to the observed visual points, like corners, collected online during the flight and clustered accordingly to their position in the image. Specifically, each cluster would be considered as a virtual object to which a convex hull is fitted and tracked through the successive images. For each virtual object, the image moments are then computed from the convex hull's vertexes representing its contour.

Due to the nonlinearity of the system an unscented Kalman filter (UKF) (Julier and Uhlmann 1997) is adopted. IMU measurements are involved only in the update phase, while for the correction phase only image information is injected. The proposed algorithm has the remarkable property of being independent of the scale factor when considering a planar scene or when the virtual objects are planar. Moreover, it is computationally cheap, such that its complexity is linear with the number of virtual objects constructed from the clusters of image features.

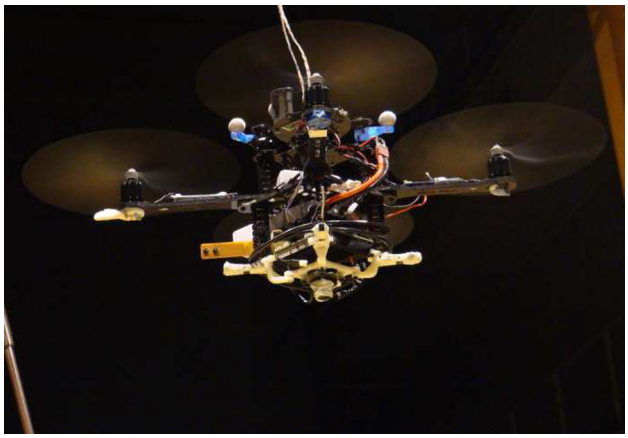


Fig. 1 The quadrotor used for the experiments is endowed with a down-oriented fisheye camera

The validity and potentiality of the proposed approach is confirmed by both simulation and experimental tests on a real flying quadrotor equipped with a down-oriented fisheye camera and a low-cost IMU (see Fig. 1).

The remainder of the paper is structured as follows. In Sect. 2, image moments along with their kinematic model are revisited. The state and measurement vectors along with the estimation model for UKF are presented in Sect. 3. Section 4 shows how virtual objects are extracted from natural visual features (corners) and upon which image moments are then computed to be used as measurements in the UKF. Simulation and experimental results on a real flying quadrotor are reported and discussed in Sects. 5 and 6, respectively, before drawing concluding remarks in Sect. 7.

2 Image moments

Let us first assume that an object is observed by the camera. The projection of this object on the current image frame is denoted by $\mathcal{S}$, as shown in Fig. 2. Let $\boldsymbol{c} = (x, y) \in \mathbb{R}^2$ be an image point corresponding to the 3D point $\boldsymbol{p} \in \mathbb{R}^3$ of the object, i.e. $\boldsymbol{c}$ lies on section $\mathcal{S}$. Image moment m_{rj} of order $(r + j)$ associated to $\mathcal{S}$ is a function of the image coordinates (x, y) of the points lying on $\mathcal{S}$ as follows (Hu 1962):

$$m_{rj} = \iint_{\mathcal{S}} x^r \, y^j \, dx \, dy. \tag{1}$$

Its time variation $\dot{m}_{rj}$ can be written in terms of the image points velocity as follows (Chaumette 2004):

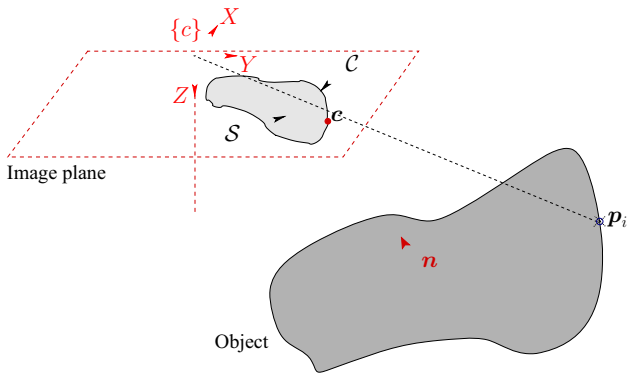


Fig. 2 Image section $\mathcal{S}$ along with its contour $\mathcal{C}$

$$\dot{m}_{rj} = \iint_{\mathcal{S}} \left(\frac{\partial f}{\partial x}\dot{x} + \frac{\partial f}{\partial y}\dot{y} + f(x,y)\left(\frac{\partial \dot{x}}{\partial x} + \frac{\partial \dot{y}}{\partial y}\right)\right) dxdy \tag{2}$$

where $f(x, y) = x^r\, y^j$, and $\dot{\boldsymbol{c}} = (\dot{x}, \dot{y})$ represents the image velocity of point $\boldsymbol{p}$.

Let $\{c\}$ be a Cartesian frame attached to the camera, such that its Z-axis coincides with the camera optical axis. Let ${}^c\boldsymbol{p} = (X, Y, Z) \in \mathbb{R}^3$ be the 3D coordinates of $\boldsymbol{p}$ in $\{c\}$, consider a perspective camera projection model such that $\boldsymbol{c} = (X/Z, Y/Z)$, and assume the object surface is planar. Thus $\boldsymbol{p}$ satisfies

$$\frac{1}{Z} = a\,x + b\,y + c, \tag{3}$$

with (a, b, c) parameters defining the plane. Let then $\mathbf{v} = (\boldsymbol{v}^\top\ \boldsymbol{\omega}^\top)^\top \in \mathbb{R}^6$ be the velocity of the camera, which is defined with respect to fixed inertial frame $\{i\}$ and expressed in frame $\{c\}$, such that $\boldsymbol{v} = (v_x\ v_y\ v_z)^\top \in \mathbb{R}^3$ corresponds to the linear velocity, and $\boldsymbol{\omega} = (\omega_x\ \omega_y\ \omega_z)^\top \in \mathbb{R}^3$ to the angular velocity. Therefore, under the assumption of motionless objects, time variation $\dot{m}_{rj}$ is given as a function of the camera velocity $\mathbf{v}$ in the following linear form (Chaumette 2004):

$$\dot{m}_{rj} = \boldsymbol{L}_{m_{rj}}\mathbf{v} \tag{4}$$

where $\boldsymbol{L}_{m_{rj}} = (m_{vx}\ m_{vy}\ m_{vz}\ m_{\omega x}\ m_{\omega y}\ m_{\omega_z}) \in \mathbb{R}^{1\times 6}$ is referred to as the interaction matrix associated with moment m_{rj}, with

$$\begin{cases} m_{vx} = -a\,m_{r,j} - r\left(a\,m_{r,j} + b\,m_{r-1,j+1} + c\,m_{r-1,j}\right) \\ m_{vy} = -b\,m_{r,j} - j(a\,m_{r+1,j-1} + b\,m_{r,j} + c\,m_{r,j-1}) \\ m_{vz} = (3+r+j)(a\,m_{r+1,j} + b\,m_{r,j+1} + c\,m_{r,j}) \\ \qquad -c\,m_{r,j} \\ m_{\omega x} = (3+r+j)\,m_{r,j+1} + j\,m_{r,j-1} \\ m_{\omega y} = -(3+r+j)\,m_{r+1,j} - r m_{r-1,j} \\ m_{\omega z} = r\,m_{r-1,j+1} - j\,m_{r+1,j-1}. \end{cases} \tag{5}$$

In the following section the above relationship is employed to construct the transition model for UKF estimation.

3 Estimation model

The analytical models describing the time update of each of the variables involved in the image moments variation are derived in this section. After this, both the state and measurements models, that will be used in a UKF algorithm to estimate linear velocity $\boldsymbol{v}$, will be derived.

3.1 Plane orientation

Let ${}^i\boldsymbol{p} \in \mathbb{R}^3$ be the 3D coordinates of $\boldsymbol{p}$ in the fixed inertial frame $\{i\}$. Let $\boldsymbol{n} \in \mathbb{R}^3$ be the normal to the object surface, which is assumed planar (see Fig. 2). It is expressed in the inertial frame and thus is constant. Since $\boldsymbol{p}$ lies on the object surface, it satisfies:

$$\boldsymbol{n}^\top ({}^i\boldsymbol{p} - {}^i\overline{\boldsymbol{p}}) = 0, \tag{6}$$

with ${}^i\overline{\boldsymbol{p}}$ another point lying on the object surface. Point $\boldsymbol{p}$ coordinates in camera frame express as a function of its coordinates in inertial frame $\{i\}$ as:

$${}^c\boldsymbol{p} = {}^c\boldsymbol{R}_i\,{}^i\boldsymbol{p} + {}^c\boldsymbol{t}_i, \tag{7}$$

where ${}^c\boldsymbol{R}_i \in \mathcal{SO}_3$ and ${}^c\boldsymbol{t}_i \in \mathbb{R}^3$ are the rotation matrix and translational vector representing the orientation and translation of inertial frame $\{i\}$ with respect to camera frame $\{c\}$, respectively. Substituting ${}^i\boldsymbol{p}$ with (7) in (6) yields

$$\acute{\boldsymbol{n}}^\top\,{}^c\boldsymbol{p} = \eta, \tag{8}$$

with

$$\begin{cases} \acute{\boldsymbol{n}} = {}^c\boldsymbol{R}_i\,\boldsymbol{n} \\ \eta = \acute{\boldsymbol{n}}^\top\,{}^c\boldsymbol{t}_i + \boldsymbol{n}^\top\,{}^i\overline{\boldsymbol{p}} \ \in \mathbb{R}, \end{cases} \tag{9}$$

where vector $\acute{\boldsymbol{n}}$ corresponds to the expression of normal vector $\boldsymbol{n}$ in camera frame, hence this quantity varies with the vehicle (camera) motion. Identifying the second equation of (9) with (3), it can be deduced that parameters a, b, and c express in terms of $\boldsymbol{n}$ as follows:

$$\boldsymbol{q} = \frac{1}{\eta}\acute{\boldsymbol{n}} = (a\ b\ c)^\top \in \mathbb{R}^3. \tag{10}$$

Time differentiating $\acute{\boldsymbol{n}}$ and η gives, after using the classical relationships stating that ${}^i\dot{\boldsymbol{R}}_c = {}^i\boldsymbol{R}_c\,[\boldsymbol{\omega}]_\times$ and that ${}^c\dot{\boldsymbol{t}}_i = -\boldsymbol{v} + [\boldsymbol{\omega}]_\times\,{}^c\boldsymbol{R}_i\,{}^i\boldsymbol{t}_c$:

$$\dot{\acute{\boldsymbol{n}}} = -[\boldsymbol{\omega}]_\times \acute{\boldsymbol{n}}, \tag{11}$$

$$\text{and} \quad \dot{\eta} = -\acute{\boldsymbol{n}}^\top \boldsymbol{v}, \tag{12}$$

where $[\boldsymbol{a}]_\times$ denotes the skew-symmetric matrix of vector $\boldsymbol{a}$. It represents the cross product in matrix form. Finally, time variation $\dot{\boldsymbol{q}}$ of $\boldsymbol{q}$ is derived from (10) after exploiting (11) and (12), as follows:

$$\begin{aligned}\dot{\boldsymbol{q}} &= \frac{1}{\eta}\dot{\acute{\boldsymbol{n}}} - \frac{1}{\eta^2}\dot{\eta}\,\acute{\boldsymbol{n}} \\ &= \frac{1}{\eta}(-[\boldsymbol{\omega}]_\times \acute{\boldsymbol{n}}) - \frac{1}{\eta^2}\acute{\boldsymbol{n}}(-\acute{\boldsymbol{n}}^\top \boldsymbol{v}) \\ &= -[\boldsymbol{\omega}]_\times \boldsymbol{q} + \boldsymbol{q}\,\boldsymbol{q}^\top \boldsymbol{v}.\end{aligned} \tag{13}$$

3.2 Velocity time update

Assume that the IMU accelerometer is calibrated. Let $\boldsymbol{a}_m \in \mathbb{R}^3$ be the measurements from the IMU accelerometer. They are expressed in the IMU frame. Let ${}^i\hat{\boldsymbol{v}}$ be the estimate of linear velocity ${}^i\boldsymbol{v}$ expressed in the inertial (global) frame. Then, time variation ${}^i\dot{\hat{\boldsymbol{v}}}$ of ${}^i\hat{\boldsymbol{v}}$ can be obtained as follows:

$$\begin{aligned}\hat{\boldsymbol{a}} &= {}^i\hat{\boldsymbol{R}}_c^\top \boldsymbol{g} + {}^c\boldsymbol{R}_{IMU}(\boldsymbol{a}_m - \hat{\boldsymbol{b}}_a) \\ {}^i\dot{\hat{\boldsymbol{v}}} &= (-{}^i\boldsymbol{\omega} \times {}^i\hat{\boldsymbol{v}} + \hat{\boldsymbol{a}})\end{aligned} \tag{14}$$

where ${}^cR_{IMU}$ corresponds to the constant orientation matrix from the camera frame to the IMU Cartesian frame (can be a priori known). As for vector $\hat{\boldsymbol{a}}$, it corresponds to nothing but the IMU acceleration expressed in the camera frame, $\hat{\boldsymbol{b}}_a \in \mathbb{R}^3$ is an estimate of the corresponding bias, and ${}^i\boldsymbol{\omega}$ corresponds to UAV (IMU) angular velocity expressed in the inertial frame. It is worth highlighting the importance of considering the estimate of the velocity in the inertial (constant) frame instead for instance the camera frame. When expressing in the camera frame indeed, the time variation of the angular velocity would appear, which yields the obtained model less interesting in view of the unnecessary introduced noise, inherent to the acceleration measurements.

3.3 State vector of the Kalman filter

Assume that N sections $\mathcal{S}$, i.e. virtual objects, are detected in the current image frame and tracked in the time, as described in more details in Sect. 4. Define vector $\boldsymbol{m}_i \in \mathbb{R}^6$ enclosing image moments up to second order computed on the ith section, such that[1]:

$$\boldsymbol{m}_i = \left(m_{00,i}\; m_{10,i}\; m_{01,i}\; m_{20,i}\; m_{11,i}\; m_{02,i}\right)^\top \in \mathbb{R}^6. \tag{15}$$

As will be explained in more detail later in Sect. 3.4, the first three components of $\boldsymbol{m}_i$ are relevant for estimating linear velocity $\boldsymbol{v}$, while the last three could be used to estimate angle vector $\boldsymbol{q}$.

For use within the Kalman filter (KF) we propose the state vector $\boldsymbol{x}$ to be defined as

$$\boldsymbol{x} = \left(\boldsymbol{v}^\top\; \bar{\boldsymbol{x}}_1^\top\; \bar{\boldsymbol{x}}_2^\top \cdots \bar{\boldsymbol{x}}_N^\top\right)^\top \in \mathbb{R}^{3+9N}, \tag{16}$$

with

$$\bar{\boldsymbol{x}}_i = (\boldsymbol{q}_i^\top\; \boldsymbol{m}_i^\top)^\top \in \mathbb{R}^9, \tag{17}$$

where $\boldsymbol{q}_i$ is the angle vector associated with the ith section and is defined by (10). Notice that all of the $\bar{\boldsymbol{x}}_i$ share the same linear velocity $\boldsymbol{v}$. This means that the higher the number of image sections, the better the robustness in estimating $\boldsymbol{v}$ with respect to measurement noise.

Next, by applying first-order Euler integration to (14), time-discrete update $\boldsymbol{v}_{k+1}$ of velocity $\boldsymbol{v}$ at time $k+1$ is given by

$${}^i\boldsymbol{v}_{k+1} = {}^i\boldsymbol{v}_k + \Delta_t\left(-{}^i\boldsymbol{\omega} \times {}^i\hat{\boldsymbol{v}} + \hat{\boldsymbol{a}}\right), \tag{18}$$

where ${}^i\boldsymbol{v}_k = {}^i\boldsymbol{R}_c\,\boldsymbol{v}_k$, and Δ_t corresponds to the sampling time. Notice that since the feedback measurements consist of image information, the estimation is performed at the camera streaming rate. Similarly, from (13) discrete update $\boldsymbol{q}_{i,k}$ of $\boldsymbol{q}_i$ can be expressed as

$$\boldsymbol{q}_{i,k+1} = \boldsymbol{q}_{i,k} + \Delta_t\left(-[\boldsymbol{\omega}_k]_\times \boldsymbol{q}_{i,k} + \boldsymbol{q}_{i,k}\,\boldsymbol{q}_{i,k}^\top \boldsymbol{v}_k\right). \tag{19}$$

Also, from (4), discrete update $\boldsymbol{m}_{ik}$ of $\boldsymbol{m}_i$ expresses

$$\boldsymbol{m}_{i,k+1} = \boldsymbol{m}_{i,k} + \Delta_t\,\boldsymbol{L}_{\boldsymbol{m},ik}\,\mathbf{v}_k, \tag{20}$$

where $\mathbf{v}_k = (\boldsymbol{v}_k^\top\; \boldsymbol{\omega}_k^\top)^\top$ and $\boldsymbol{L}_{\boldsymbol{m},ik} = (\boldsymbol{L}_{m_{i,00},k}^\top\, \boldsymbol{L}_{m_{i,10},k}^\top\, \boldsymbol{L}_{m_{i,01},k}^\top\, \boldsymbol{L}_{m_{i,20},k}^\top\, \boldsymbol{L}_{m_{i,02},k}^\top\, \boldsymbol{L}_{m_{i,11},k}^\top)^\top$, with $\boldsymbol{L}_{m_{i,rj},k}$ given by (5) computed at discrete time k, such that $m_{i,rj,k}$ denotes moment m_{rj} of the ith section at time k.

3.4 Measurements

The proposed measurements vector is based on image moments extracted from the current image frame as follows:

$$\boldsymbol{z} = \left(\boldsymbol{z}_1^\top \boldsymbol{z}_2^\top \cdots \boldsymbol{z}_N^\top\right)^\top \in \mathbb{R}^{6N}, \tag{21}$$

such that $\boldsymbol{z}_i$ encloses moments corresponding to the ith of the N sections extracted from the image, as detailed in the next section. In particular, the following quantities are chosen:

[1] $m_{rj,i}$ denotes the $(r+j)$th order moment associated to the ith section.

$$\boldsymbol{z}_i = \begin{pmatrix} \sqrt{a_i} \\ \bar{x}_i \\ \bar{y}_i \\ \eta_{20,i} \\ \eta_{11,i} \\ \eta_{02,i} \end{pmatrix} \in \mathbb{R}^6, \tag{22}$$

where a_i is the area of the ith section, $(\bar{x}_i, \bar{y}_i)$ its center of gravity in the image, and $(\eta_{20,i}, \eta_{11,i}$, and $\eta_{02,i})$ are second order moments invariant to scale and translation. Notice that all these quantities can be expressed in terms of image moments only. More precisely, the first three components of $\boldsymbol{z}_i$ express in terms of moments up to the first order as follows (Hu 1962):

$$\begin{aligned} a_i &= m_{00,i} \\ \bar{x}_i &= m_{10,i}/m_{00,i} \\ \bar{y}_i &= m_{01,i}/m_{00,i}, \end{aligned} \tag{23}$$

while the last three components are given as follows:

$$\eta_{rj} = \frac{\mu_{rj}}{a^{1+\frac{r+j}{2}}}, \tag{24}$$

where μ_{rj} are centered image moments, which are invariant to translation, defined as follows:

$$\mu_{rj} = \iint_{\mathcal{S}} (x - \bar{x})^r \, (y - \bar{y})^j \, dx \, dy. \tag{25}$$

They can be expressed in terms of image moments only, as follows:

$$\mu_{rj} = \sum_{k}^{r} \sum_{h}^{j} \binom{r}{k} \binom{j}{h} \left(-\bar{x}\right)^{(r-k)} \left(-\bar{y}\right)^{(j-h)} m_{kh} \tag{26}$$

with $\binom{\cdot}{\cdot}$ denoting the binomial coefficient. Notice that the first three components $(\sqrt{a_i}\,, \bar{x}_i, \bar{y}_i)$ would be the main actors in the estimation of $\boldsymbol{v}$. Indeed, the area variation corresponds mainly to motions along the optical axis of the camera, that is component v_z of $\boldsymbol{v}$. On the other hand, center of gravity $(\bar{x}_i, \bar{y}_i)$ is correlated mainly to the motions along the camera plane axes, that is velocities v_x and v_y. This shows the relevance of the area and center of gravity in the estimation of linear velocity $\boldsymbol{v}$. Moreover, invariant second order moments η_{rj} have been introduced mainly to extract an estimate of $\boldsymbol{q}_i$. Indeed, second order image moments describe the orientation of an observed object in the image which intuitively would be mainly correlated to the camera orientation, hence to $\boldsymbol{q}$. Also the invariance properties of η_{rj} would yield the first three components of $\boldsymbol{z}_i$ decoupled from the last three, which would enhance the estimation performance.

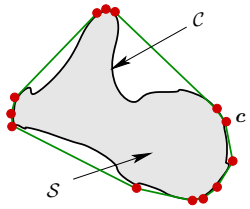


Fig. 3 Sketch of a polygon (convex hull) fitting the points of contour $\mathcal{C}$. The vertexes defining the polygon are shown in *red-filled circles* (Color figure online)

The computation of image moments needed by the algorithm during run time is not performed by using formula (1) as it is, but exploiting only the contour points. Specifically, instead of the whole points lying on a considered section, only the vertexes of the convex hulls (i.e. polygons) are used. This reduces the computational complexity from square to linear. Accordingly, let the extracted contour be characterized by a set of m image points, where $\boldsymbol{c}_j = (x_j, y_j)$, with $j = 1, \ldots, m$, being the jth point (see Fig. 3). Add $\boldsymbol{c}_0 = \boldsymbol{c}_m$ as the $(m+1)$th point, so that all these points form a closed contour. Then, relationship (1) expressing an $(r+j)$th order image moment recast as follows (Steger 1996):

$$\begin{gathered} m_{rj} = \sum_{i=1}^{m} (y_i - y_{i-1}) \, \sigma_{i,rj}, \text{ with } \sigma_{i,rj} = \sum_{k=0}^{r+1} \sum_{l=0}^{j} a_{kl} \, f_x \, f_y \\ \begin{cases} f_x = x_i^k \, x_{i-1}^{r+1-k}, \quad f_y = y_i^l \, y_{i-1}^{j-l} \\ a_{kl} = \frac{1}{(r+j+2)(r+1)} \binom{r+1}{k} \binom{j}{l} \Big/ \binom{r+j+1}{k+l}, \end{cases} \end{gathered} \tag{27}$$

where $x^\beta = y^\beta = 0$ for $\beta < 0$ is considered.

3.5 Kalman filter-based estimation

In this section the analytical model to perform Kalman Filter-based estimation is presented. Expressions (18), (19), (20), and (21) can be recast in the following transition and observation relationships:

$$\begin{cases} \boldsymbol{x}_{k+1} = \boldsymbol{g}(\boldsymbol{x}_k, \, \boldsymbol{\omega}_k, \, \boldsymbol{a}_k) \in \mathbb{R}^{3+9N} \\ \boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k) \in \mathbb{R}^{6N} \end{cases}, \tag{28}$$

where

$$\begin{aligned} \boldsymbol{g} &= (\boldsymbol{g}_v^\top \; \boldsymbol{g}_1^\top \cdots \boldsymbol{g}_N^\top)^\top \\ \boldsymbol{g}_v &= {}^i\boldsymbol{v}_k + \Delta_t \left(- {}^i\boldsymbol{\omega} \times {}^i\hat{\boldsymbol{v}} + \hat{\boldsymbol{a}} \right) \in \mathbb{R}^3 \\ \boldsymbol{g}_i &= \begin{pmatrix} \boldsymbol{q}_{i,k} - \Delta_t \, [\boldsymbol{\omega}_k]_\times \, \boldsymbol{q}_{i,k} + \Delta_t \, \boldsymbol{q}_{i,k} \boldsymbol{q}_{i,k}^\top \, \boldsymbol{v}_k \\ \boldsymbol{m}_{i,k} + \Delta_t \, \boldsymbol{L}_{\boldsymbol{m},ik} \, \mathbf{v}_k \end{pmatrix} \in \mathbb{R}^9 \end{aligned} \tag{29}$$

and

$$\begin{aligned} \boldsymbol{h}(\boldsymbol{x}_k) &= \left(\boldsymbol{h}_1(\bar{\boldsymbol{x}}_1)^\top \; \boldsymbol{h}_2(\bar{\boldsymbol{x}}_2)^\top \cdots \; \boldsymbol{h}_N(\bar{\boldsymbol{x}}_N)^\top \right)^\top \\ \boldsymbol{h}_i(\boldsymbol{x}_i) &= \left(\sqrt{\bar{x}_{i,4}} \;\; \frac{\bar{x}_{i,5}}{\bar{x}_{i,4}} \;\; \frac{\bar{x}_{i,6}}{\bar{x}_{i,4}} \;\; \eta_{xi,20} \;\; \eta_{xi,11} \;\; \eta_{xi,02} \right)^\top, \end{aligned} \tag{30}$$

with $\bar{x}_{i,r}$ the rth element of $\bar{\boldsymbol{x}}_i$ at time-sample k. Recall that $\bar{\boldsymbol{x}}_i$ is defined by (17). As for $\eta_{xi,rj}$, it simply corresponds to η_{rj} given by (24) but is expressed in terms of the elements of $\boldsymbol{m}_i$ given by (15) and enclosed in $\bar{\boldsymbol{x}}_i$.

As can be seen from (28) and (29), along with all the involved variables introduced above, the obtained model is independent of the scale factor and this constitutes a novelty with respect to the state-of-the-art. Since the model is nonlinear, a UKF algorithm is adopted to perform the estimation (Julier and Uhlmann 1997). Doing so allows to better handle the nonlinearities than for instance EKF does. Our implementation of the UKF follows that described in Crassidis and Markley (2003), which, in contrast to the original version presented in Julier and Uhlmann (1997), does not require the state vector or the covariance matrices be augmented, thus reducing the computational complexity.

Furthermore, it is worth noting that the computational complexity of the proposed estimator (model) is linear with respect to the number N of the (virtual) objects and not in terms of the number of observed feature points. This considerably downscales the problem. In addition, the different computations for the estimation can be optimized. Indeed, each of three phases corresponding to the computation, propagation, and projection of the $[2(3+9N)+1]$ sigma points of the UKF can be performed in parallel (multi-threading). This means that the computational complexity related to the sigma points would not be substantially affected when larger number of (virtual) objects is considered. Only two parts of the pipeline can not be readily parallelized: the computation of the state covariance matrix square root at each of the prediction and update phases.

4 Virtual contour extraction

There are different ways to approach the issue of segmenting an object from an image. To our knowledge, in most of the state-of-the-art works the segmented section corresponds to an image of a physical object or a part of it. In the present work, we propose a different approach that consists of extracting contours of virtual objects. The algorithm fits contours (polygons, more precisely) to sets of the detected visual features (corners in this work). These corners do not necessarily belong to a same physical object and thus the corresponding polygon delimits a virtual object. There are at least two advantages of doing so. The first one is that the proposed estimation can handle any image provided of course that some feature points (contrast) are present. The second advantage is the reduced computational complexity. Simply extracting corners is clearly faster than segmenting the contours of physical objects from the scene. After the contour has been extracted, image moments are then computed for the image section delimited but that contour.

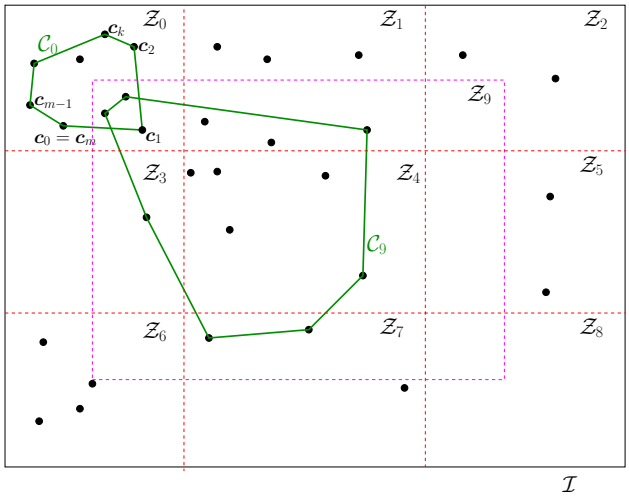


Fig. 4 Sketch of the proposed contour extraction principle. The current image frame, $\mathcal{I}$, is partitioned into N zones, denoted by $\mathcal{Z}_i$'s. Here we consider $N = 10$ zones. The first nine zones are delimited by the *vertical and horizontal red-dashed lines*, while the 10th is indicated with the *rectangle in magenta*. The detected features are represented with *black dots* (*tiny filled circles*). Note that a feature point might be assigned to different zones, such that in our example zones $\mathcal{Z}_0$ and $\mathcal{Z}_9$ share three features. Then, each zone's set of features is fitted with a contour, denoted by $\mathcal{C}$. Specifically, the contour consists in a convex polygon (plotted in *green*), whose vertexes are defined by the most external points of the considered zone (convex-hull). Only two contours are sketched for the sake of illustration (Color figure online)

The algorithm principle and data flow is presented with more details in the sequel.

Firstly, the acquired image frame is divided into different zones as sketched in Fig. 4. In this work, we choose to define 10 zones, denoted by $\mathcal{Z}_i$'s, with $i = 0, \ldots, 9$. Nine of them correspond to rectangles resulting from dividing the image height and width into three equivalent columns and rows. The tenth zone corresponds to a larger rectangle centered in the image, so as to capture more feature points. Another possibility is to merge adjacent zones if few features were detected, yet a large number of further possibilities could be considered.

Assume that feature points have been detected in the current image frame. Then, each of these points will be assigned to one of the defined zones according to its image coordinates, as sketched in Fig. 4. Next, for each zone a contour $\mathcal{C}$ is evaluated by considering the convex hull of the observed points in the region. We use OpenCV for the purpose. The points lying on the polygon are denoted by $\boldsymbol{c}_k$ in Fig. 4, with $k = 0, \ldots, m$, and are sorted counter-clockwise. Recall that such points have already been introduced earlier and were similarly denoted by $\boldsymbol{c}$, as depicted in Figs. 2 and 3.

In this work we adopt ORB detector for contour extraction (Rublee et al. 2011), which is reported to feature both the performance of SIFT and the speed of FAST. The extrac-

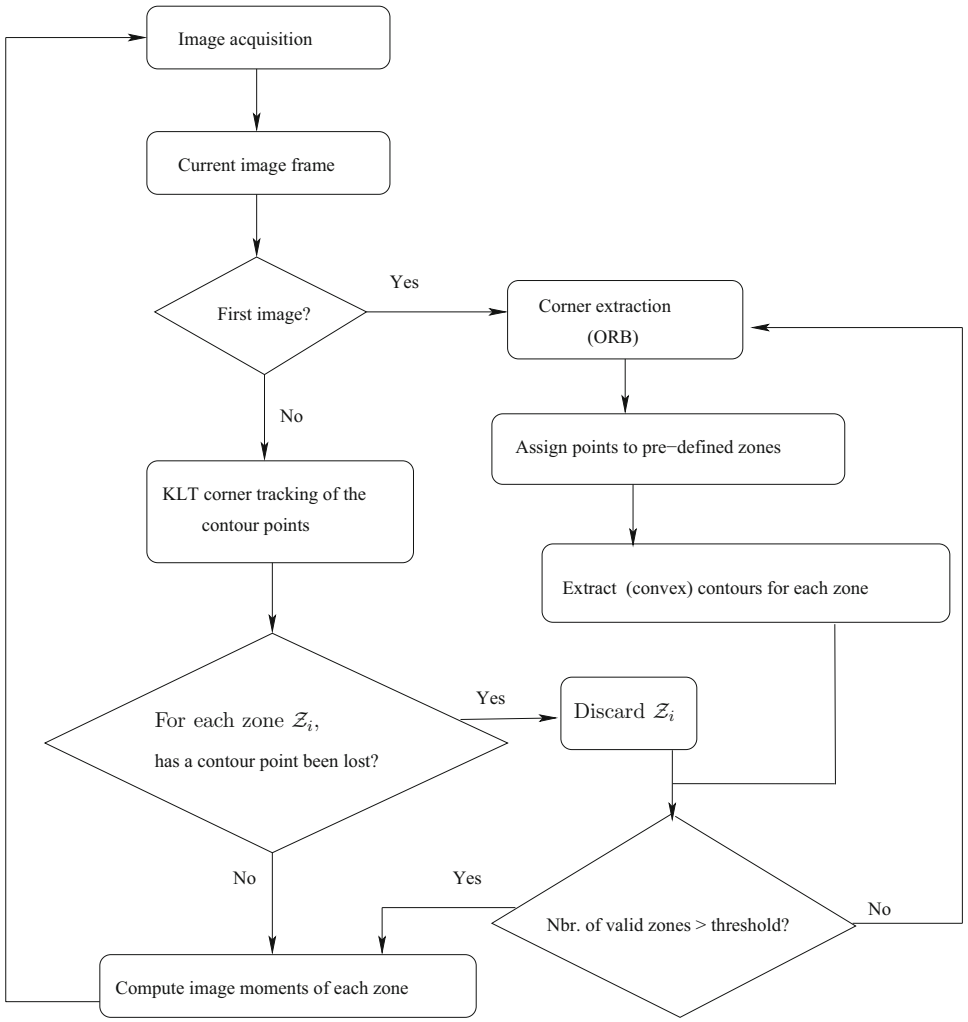


Fig. 5 Flow chart of the proposed image contours extraction and tracking

tion process is performed only at the initial time and when the number of detected points for a region drops under a defined threshold. Indeed, once a contour is extracted, it is then only tracked through the subsequent frames with a well-known Kanade–Lucas–Tomasi (KLT) feature tracker (Lucas and Kanade 1981). It is worth pointing out that by performing tracking instead of image detection, the algorithm considerably gains in terms of speed. Moreover, in our implementation the tracking of all the contours is performed in parallel (multi-thread), which again speeds up the algorithm. Finally, contour points $\boldsymbol{c}$'s are then readily used to compute the image moments using formula (27).

A flowchart summarizing the different steps of the whole image contours extraction and tracking is drawn in Fig. 5. The whole algorithm is written in C++ using OpenCV, and ROS[2] and is open-source. Figure 6 shows a real image sequence on which the contours obtained with the described algorithm are displayed.

[2] http://www.ros.org.

5 Simulation results

The algorithm proposed in this paper contains two main and subsequent phases: the image elaboration part wherein virtual objects (polygons) are assigned to the collected image corners as described in Sect. 4; and the KF phase that uses the image moments computed in the previous phase to get an estimate of the linear velocity as described in Sect. 3.5. It is worth recalling that these two phases are not strictly linked in the sense that the image elaboration can be performed according to different techniques. For instance, a user might prefer to directly segment physical objects from the scene and then compute their moments. Though, this requires that real-physical objects be present in the scene; a constraint that has been relaxed with our approach such that only visual points would suffice, even if they do not correspond to the same object.

In this section we report numerical simulation results to verify the validity of the KF phase. As for the image elaboration part, it is tested instead directly in the real scenario as

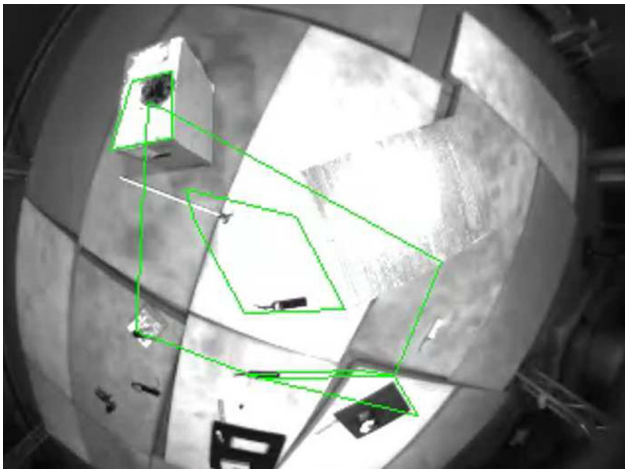


Fig. 6 Image frame from the onboard fisheye camera on which are displayed four extracted contours (*polygons in green*) by the proposed algorithm (Color figure online)

reported in the next section. We assume that physical objects are in the scene. For simplicity, each object is described by its contour, which consists in a polygon whose vertexes are pre-defined. The objects are motionless. The images of these objects are simulated by perspective projection of the different vertexes. Then image moments are computed and fed to the proposed UKF.

5.1 Camera frame rate of 25 Hz

We first consider the case where the images are provided at the rate of 25 Hz. Each observed object has its vertexes lying on the ground, thus respecting the coplanar constraint given by (3). Each of the involved measurements is relayed with a bias and an additive white Gaussian noise (AWGN) of a certain standard deviation (SD). In details, the roll, pitch, and yaw Euler angles of the UAV attitude measurements from the onboard IMU are each corrupted with a bias of 1°, and an AWGN of 1° SD. Likewise, the components of the angular velocity measurements from the IMU gyro are corrupted with a bias of 0.57 °/s , and an AWGN of 2°/s SD. Since the current version of the algorithm uses the value of the commanded thrust (as more described in Sect. 6), the latter is assumed corrupted with a bias of 2 % the nominal value (that equal for hovering, i.e. $m\,g$), and an AWGN of 2 % the nominal value as SD. Finally, the image coordinates are corrupted with an AWGN of SD 5 % the nominal value.

It is assumed that six planar objects lie on the ground and are observed buy the camera onboard the vehicle. The objects have been defined and positioned randomly in the scene. While the vehicle achieves a sinusoidal motion around a position high by 6 m from the ground, the proposed estimator is fed with the images of these six objects. Corresponding results are shown in Fig. 7. We can see that the proposed algorithm yields good estimates of the linear velocity, though the presence of large noise. As can be seen indeed from Fig. 7d, e, and f, the measurements errors on the attitude exceed 5° at least. Likewise, Fig. 7g, h, and i reveal that the measurements error on the angular velocity exceed 7°/s at least. The large measurements error corrupting the force (i.e. acceleration) applied to the vehicle and given by (32) can be appreciated from Fig. 7j–l. The evolution of the state covariance matrix's diagonal elements are shown in Fig. 8.

5.2 Camera frame rate of 100 Hz

Now we consider the same scenario of Sect. 5.1, except that the camera is 4 times faster; generating images at 100 Hz. This allows us to check that the performance of the estimator is correlated to the images streaming rate. Corresponding simulation results are provided in Fig. 9. We can see from Fig. 9a–c that the filter yields indeed better velocity estimates than in the previous case of 25 Hz.

5.3 Non-coplanar feature points

Finally, we consider the scenario where the (virtual) objects are no longer planar; that is, the vertexes defining the object contour are not coplanar. This violates constraint (3) based on which the transition function (20) relating the image moments time variation has been derived. It is however worth remarking that doing so can in fact be seen as equivalent to applying an EKF to a nonlinear system. Indeed, model (3) would represent a linearization of the true nonlinear model of the scene. Nevertheless, in contrast to EKF where all the involved dynamics are linearized, the UKF adopted in this work keeps the other states with their original nonlinear model; only a subset would be concerned by the linearization. To summarise, our algorithm would still hold in case the scene is not planar, similarly as an EKF would behave when applied to a nonlinear system, except that the former would better handle the overall nonlinearities.

The simulations reported in this section allow to verify the above statement. The same scenario as in the precedent simulation is considered, except that the vertex points no longer lie on the same plane but are at different elevations (up to 4 m of difference in height). Corresponding results are shown in Fig. 10. We can see indeed that the estimator still yields good estimates of the linear velocity, although the non-planarity of the objects is severe.

6 Experimental results

This section reports results obtained using the proposed method on a real hardware system. The objective is to esti-

Fig. 7 Simulation results obtained with the proposed estimator fed with images at 25 Hz. The simulation scenario and conditions are described in Sect. 5.1. Six planar objects lie in the scene observed by the onboard camera. **a–c** Estimated $\hat{\boldsymbol{v}}$ vs. actual velocity $\boldsymbol{v}$. Vector $\boldsymbol{r}_{xyz} \in \mathbb{R}^3$ corresponds to the XYZ Euler orientation of the vehicle attitude. Vector $\mathbf{f}_0$ corresponds to $\mathbf{f}/m$, where force $\mathbf{f}$ is given by (32). Remark that except for velocity $\boldsymbol{v}$, $\hat{a}$ denotes the measurement used by our estimation algorithm while a denotes instead the actual (true) value. **n** 3D view, where both the path followed by the UAV and the observed objects lying on the ground are depicted. The relative altitude of the UAV from the ground is indicated with the palette on the right of the figure. **m** Image frame from the onboard camera. The estimation state-covariance matrix is shown in Fig. 8

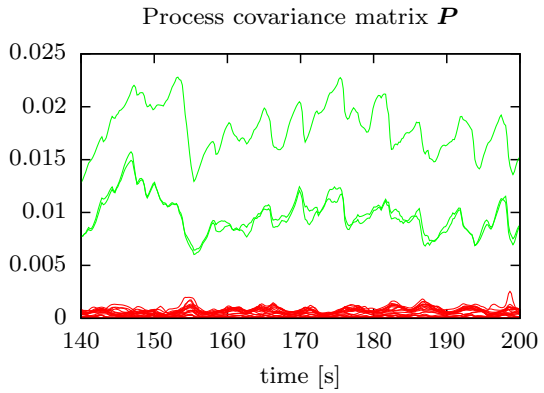


Fig. 8 State covariance matrix's diagonal elements; following results of Fig. 7. The first three elements (i.e. those related to linear velocity $\boldsymbol{v}$) can be distinguished with their *green color* (Color figure online)

mate the linear velocity of a UAV quadrotor using only onboard sensing: vision aided by IMU.

6.1 Experimental setup

In these experiments we employ a Pelican quadrotor[3] (see Fig. 1), of roughly 1.5 kg total mass, flying in a GPS-denied and poorly-textured environment (see Fig. 6). A UI-1221LE uEYE video camera equipped with a fisheye lens is attached to the bottom of the vehicle. The video camera streams images at the rate of 16 Hz with a resolution of 640 × 480. Note that no prior knowledge of the scene has been considered in the estimation. The validity of the proposed method is highlighted by comparing to ground truth velocity extracted from an Optitrack motion capture system, consisting of 20 infrared cameras covering the UAV operational space.

Since the accelerometer at our possession does not provide measurements with enough reliability, we employ instead the value of the commanded thrust as a substitute. We propose indeed to leverage the equations of motions modeling the dynamics of the vehicle. Specifically, a vehicle of mass m would have its acceleration expressed in terms of the actual total thrust $u \in \mathbb{R}$ applied to the vehicle. We would have (Shakernia et al. 1999; Hamel and Mahony 2002):

$$\dot{\boldsymbol{v}} = -\boldsymbol{\omega} \times \boldsymbol{v} + \frac{1}{m}\mathbf{f}, \tag{31}$$

where $\mathbf{f} \in \mathbb{R}^3$ corresponds to the total force applied to the vehicle. It can be indeed expressed as a function of total thrust u applied to the vehicle as follows:

$$\mathbf{f} = -u\,\mathbf{e}_3 + m\,g\,\mathbf{R}^\top \mathbf{e}_3 \in \mathbb{R}^3, \tag{32}$$

where $\boldsymbol{R}$ is the rotation matrix defining the orientation of the Cartesian frame attached to the body frame of the vehicle with respect to inertial frame $\{i\}$, g is the gravity term, while $\boldsymbol{e}_3 = (0\ 0\ 1)^\top$. Next, applying first-order Euler integration to (31), time-discrete update $\boldsymbol{v}_k$ of velocity $\boldsymbol{v}$ at time k writes

$$\boldsymbol{v}_{k+1} = \boldsymbol{v}_k + \Delta_t \left(-[\boldsymbol{\omega}_k]_\times \boldsymbol{v}_k + \frac{1}{m}\mathbf{f}_k \right), \tag{33}$$

where $\boldsymbol{\omega}_k$ and $\mathbf{f}_k$ correspond to the discrete updates of $\boldsymbol{\omega}$ and $\mathbf{f}$, respectively. Finally, the estimation model is obtained by only substituting $\boldsymbol{g}_v$ involved in (29) with the following:

$$\boldsymbol{g}_v = \boldsymbol{v}_k - \Delta_t\,[\boldsymbol{\omega}_k]_\times \boldsymbol{v}_k + \Delta_t \frac{1}{m}\mathbf{f}_k \in \mathbb{R}^3. \tag{34}$$

The gains of the UKF have been empirically tuned as follows. The parameters α, β, and κ are set to $1\mathrm{e}^{-3}$, 2, and 0, as classically considered. The state covariance matrix is set to $\boldsymbol{R}_v = \mathrm{diag}(1\mathrm{e}^{-3}\boldsymbol{I}_3,\ \boldsymbol{R}_{v1},\ \boldsymbol{R}_{v2}, \ldots, \boldsymbol{R}_{vN})$, with $\boldsymbol{R}_{vi} = \mathrm{diag}(1\mathrm{e}^{-8}\boldsymbol{I}_3,\ 1\mathrm{e}^{-8}\boldsymbol{I}_6)$, such that $i = 1, \ldots, N$. The measurement covariance matrix is set to $\boldsymbol{R}_n = \mathrm{diag}\big(\boldsymbol{R}_{n1}, \boldsymbol{R}_{n2}, \ldots, \boldsymbol{R}_{nN}\big)$, such that $\boldsymbol{R}_{ni} = 5\mathrm{e}^{-5}\boldsymbol{I}_6$, with $i = 1, \ldots, N$. As for the initial values, the state covariance matrix is set to $\boldsymbol{P} = \mathrm{diag}(5\mathrm{e}^{-4}\boldsymbol{I}_3,\ \boldsymbol{P}_1, \ldots, \boldsymbol{P}_N)$, such that $\boldsymbol{P}_i = \mathrm{diag}(1\mathrm{e}^{-6}\boldsymbol{I}_3,\ 1\mathrm{e}^{-6}\boldsymbol{I}_6)$, with $i = 1, \ldots, N$. The initial estimate of the state vector is set to $\hat{\boldsymbol{x}}_{t_0} = \mathbf{0}_{3+9N}$. To compute the square root of the covariance matrix during run time, we use Cholesky decomposition. The GSL C++ open source library[4] is used for that purpose.

Regarding corner extraction, ORB detector is assigned with a maximum number of 100 features. The extraction procedure is repeated each time the number of valid contours (contours being tracked) of a region drops under a certain threshold. A contour (a polygon more precisely) is declared valid when: i) the number of its vertexes is higher than three (to fully define an object); ii) all its vertexes are being correctly tracked. If only one vertex is lost during KLT tracking, then the corresponding contour is declared invalid and a new contour is searched in the region. A total number of roughly 40 corners (vertexes) were tracked in this experiment.

6.2 Planar scene

A scene almost planar is initially considered. Corresponding estimation results are shown in Fig. 11. We can see from Fig. 11a and b that the proposed algorithm is able to estimate relatively well the velocities along the image plane v_x and v_y. The algorithm is also able to recover velocity v_z along the image optical axis (which also corresponds mainly to the elevation velocity), but not as well as for v_x and v_y (see

[3] http://www.asctec.de.

[4] GSL - GNU Scientific Library: http://www.gnu.org/software/gsl/.

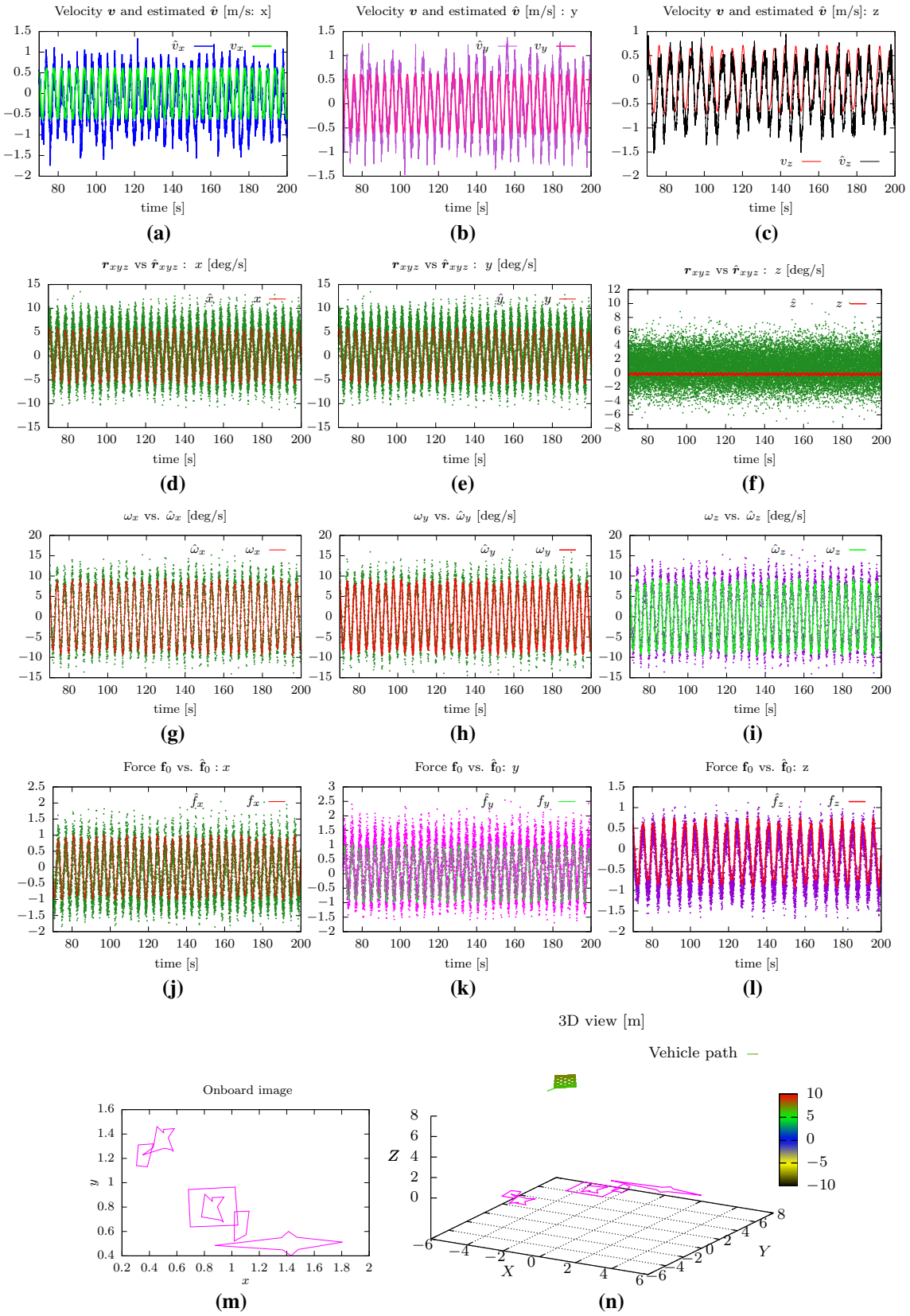


Fig. 9 Simulation results obtained with the proposed estimator fed with images at 100 Hz as described in Sect. 5.2

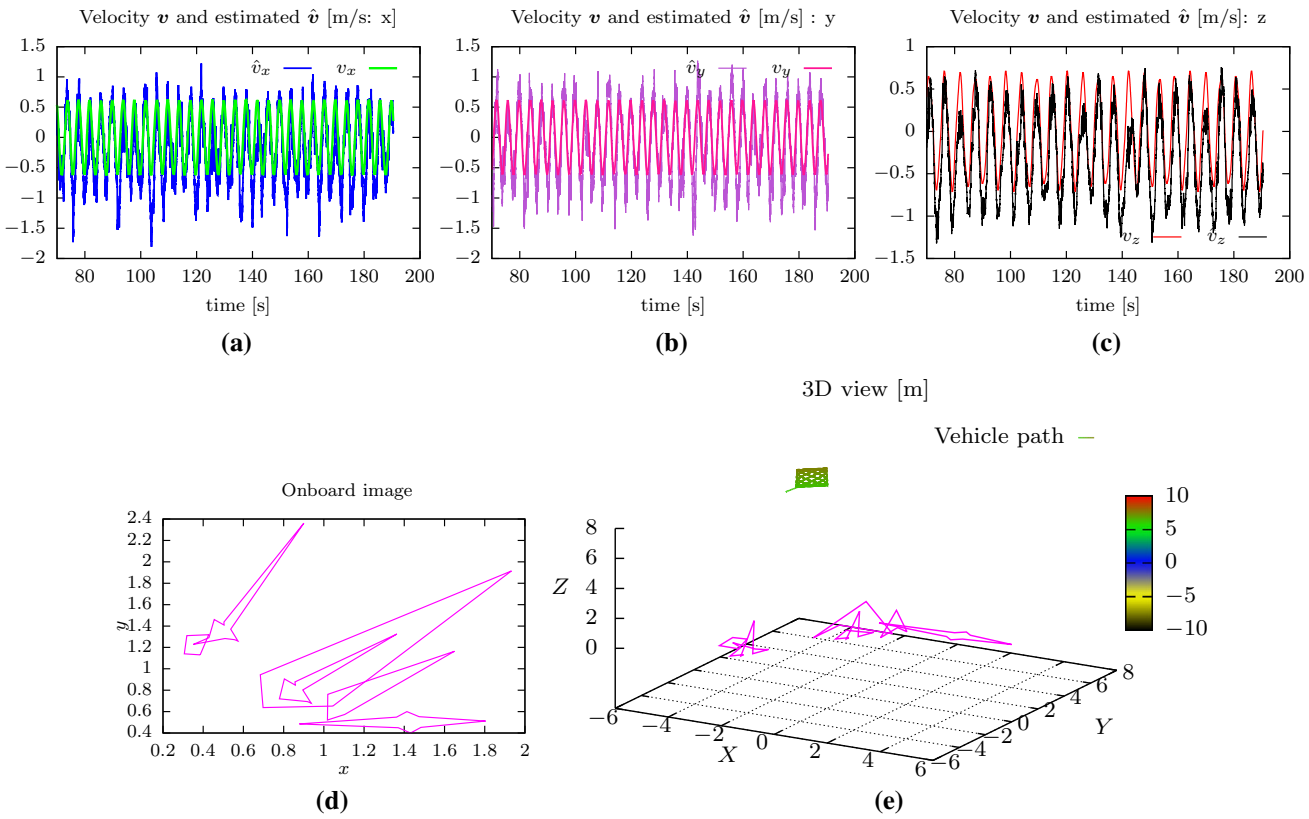


Fig. 10 Simulation results in case of non-planar objects. The images are provided at the rate of 100 Hz

Fig. 11c). This is mainly due to the errors from estimating the acceleration based on the actuated thrust.

A more sophisticated and accurate estimation of the actual thrust taking care for example of the battery charge level could improve the performance of the proposed velocity estimator. For example, the two instants where the algorithm did not estimate well v_z are around time 20 s and time 43 s. At those instants, in fact, the vehicle was randomly and purposefully shoved up by pulling on the rope attached to its frame from top to inject disturbances into the system. Thus, the generated motion is due to the external force of the pull, instead to the commanded thrust as our algorithm assumes. However, whenever the rope is freed and the UAV is in a descent manoeuvre, the algorithm yields a good estimate of v_z, since in this case the motions are due to the actuated thrust. These results highlight the shortcoming of relying only on the thrust value. A high-quality accelerometer would instead capture all these motions and would yield better results.

We also notice some spikes on the estimated velocity, especially on $\hat{v}_x$ and $\hat{v}_y$. We owe this mainly to the image tracking, more precisely to the shakiness of the tracked contours. Indeed the KLT tracker keeps jumping around the area of some corners. This is likely due to the low quality of the relayed images, in addition to the change in brightness due to illumination. Note that for each iteration it took roughly 4 ms to achieve the KF phase (after image elaboration) on an Intel CORE i3 processor.

6.3 Non-planar scene

Though the transition model relating the image moments time update has been derived upon assumption that the scene is planar, it is expected that the update phase of the algorithm that uses only image information would correct the estimate from any errors due to the non-satisfaction of that constraint, as has been described in Sect. 5.3. In this sense, we test the proposed method on an object non-planar. A box of roughly 0.6 m height has been added to the originally horizontal scene, leading locally to a non-planar scene (see the box at the top left of Fig. 6). Corresponding estimation results are reported in Fig. 12 where one can see that the linear velocity is still recovered. This confirms that the proposed algorithm is applicable to a not perfectly planar scenario, i.e. that the sensibility of the algorithm to the planarity of the scene is small.

6.4 Final considerations

Finally, and more remarkably, it is worth pointing out that the proposed estimator is able to recover the scale of the linear

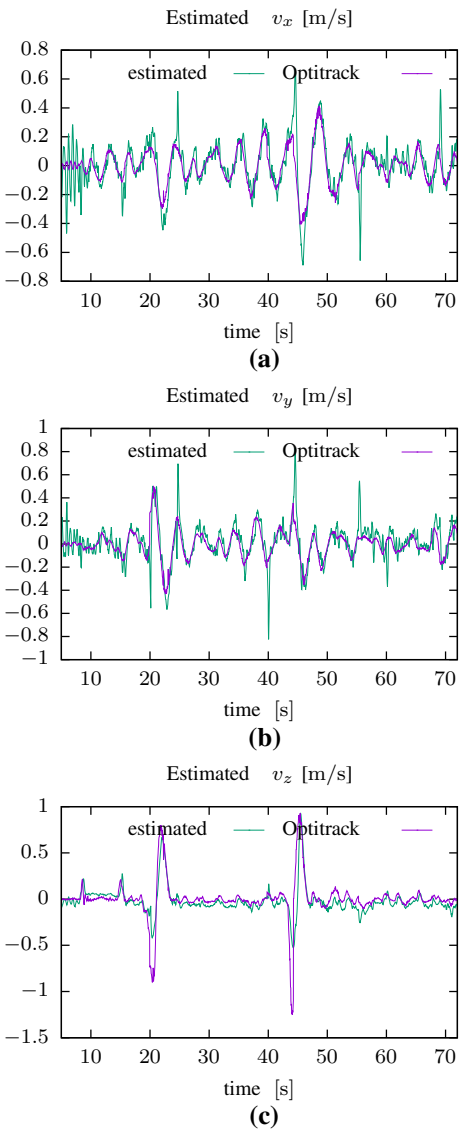


Fig. 11 Estimate $\hat{\boldsymbol{v}}$ of linear velocity $\boldsymbol{v}$ compared to ground truth from optitrack motion capture system in case of planar scene

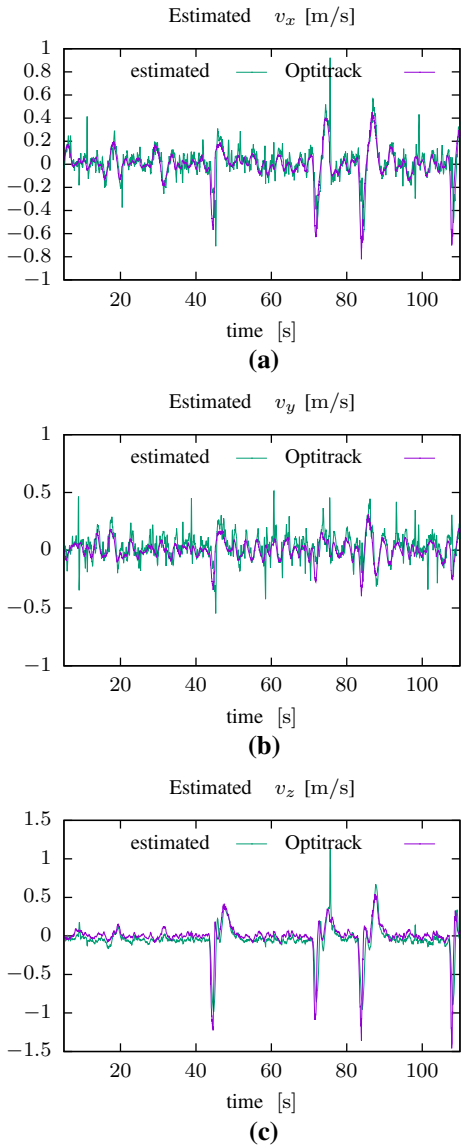


Fig. 12 Estimated velocity from a non-planar scene (i.e. non coplanar points)

velocity without any knowledge or explicit estimation of the depth. This confirms that the algorithm is independent of the scale factor, as analytically proved from the obtained model used for Kalman estimation. This positive trait is indeed the result of involving the image area (moments) as a state and measurement element in the estimator. The area variations, to recall, are indeed mainly correlated to the motions along the image depth.

The current formulation of the estimator assumes calibrated IMU measurements: attitude as well as acceleration. This might be convenient for a user desiring to employ specific calibration tools of the IMU. Notice also that the experimental results reported here are obtained with our estimator fed with only raw IMU data. In the next works, we will augment the estimator state and measurements in such a way that both the accelerometer and gyroscope have their biases estimated and their signals filtered in real time. We also plan to recover even the camera position based on this formulation.

Finally, shakiness of the tracked contours would lead to noisy image moments, especially for those of second order, which would be reflected negatively on the estimator performance. Change in brightness (due mainly to light reflection) makes the situation worse. The spikes would also be caused by UKF re-initialization, which is triggered by each corner extraction. We need to smooth the change in brightness, handle appropriately the transition phases, and reject the outliers. This will be the subject of our next works.

7 Conclusion

In this work we developed a novel method for estimating the unavailable linear velocity of a UAV equipped with onboard vision and IMU, in a GPS-denied and unknown environment. The algorithm is general in the sense that it only requires that visual corners be detected and tracked for a certain time interval. We proposed a combination of image moments as the sole feedback for the estimation, such that a UKF has been adopted for the propose. The light computational complexity of the algorithm has been highlighted, making it appealing for autonomous robotics. Both simulation and experimental results on a real hardware system equipped with only a low-cost IMU confirmed the validity of the proposed approach. The proposed algorithm is independent of the scale factor.

In the near future, we will employ a better quality IMU and the algorithms will be fed with measurements from calibrated sensors. In addition, we will rely on an accelerometer, in contrast to the present work where only an approximate from the commanded thrust has been injected. We will then benchmark the proposed algorithm with respect to the state-of-the-art.

Acknowledgments The research leading to these results has been supported by the ARCAS and SHERPA collaborative projects, which have received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreements ICT-287617 and ICT-600958, respectively. The authors are solely responsible for its content. It does not represent the opinion of the European Community and the Community is not responsible for any use that might be made of the information contained therein.

References

Achtelik, M., Achtelik, M., Weiss, S., & Siegwart, R. (2011). Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments. In *IEEE International Conference on Robotics and Automation*.

Castillo, P., Dzul, A., & Lozano, R. (2004). Real-time stabilization and tracking of a four-rotor mini rotorcraft. *IEEE Transactions on Control Systems and Technology*, *12*, 510–516.

Chaumette, F. (2004). Image moments: A general and useful set of features for visual servoing. *IEEE Transactions on Robotics*, *20*(4), 713–723.

Crassidis, J. L., & Markley, F. L. (2003). Unscented filtering for spacecraft attitude estimation. *Journal of Guidance, Control, and Dynamics*, *6*, 536–542.

Espiau, B., Chaumette, F., & Rives, P. (1992). A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, *8*, 313–326.

Grabe, V., Bulthoff, H., & Giordano, P. (2012). Robust optical-flow based self-motion estimation for a quadrotor UAV. In *IEEE International Conference on Intelligent Robots and Systems* (pp. 2153–2159).

Hamel, T., & Mahony, R. (2002). Visual servoing of an under-actuated rigid body system: An image based approach. *IEEE Transactions on Robotics and Automation*, *18*, 187–198.

Honegger, D., Meier, L., Tanskanen, P., & Pollefeys, M. (2013). An open source and open hardware embedded metric optical flow CMOS camera for indoor and outdoor applications. In *IEEE International Conference on Robotics and Automation*.

Hu, M. K. (1962). Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, *8*, 179–187.

Julier, S., & Uhlmann, J. (1997). A new extension of the kalman filter to nonlinear systems. In *11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls*.

Kneip, L., Martinelli, A., Weiss, S., Scaramuzza, D., & Siegwart, R. (2011). Closed-form solution for absolute scale velocity determination combining inertial measurements and a single feature correspondence. In *IEEE International Conference on Robotics and Automation* (pp. 4546–4553).

Lippiello, V., & Mebarki, R. (2013). Closed-form solution for absolute scale velocity estimation using visual and inertial data with a sliding least-squares estimation. In *21st Mediterranean Conference on Control and Automation* (pp. 1261–1266).

Lucas, B. D., & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *7th International Joint Conference on Artificial Intelligence* (pp. 674–679).

Ma, Y., Soatto, S., Kosecka, J., & Sastry, S. S. (2003). *An Invitation to 3-D Vision: From Images to Geometric Models*. New york: Springer.

Mebarki, R., & Lippiello, V. (2014). Image moments-based velocity estimation of UAVs in GPS denied environments. *IEEE International Symposium on Safety, Security, and Rescue Robotics* (pp. 1–6).

Mebarki, R., Lippiello, V., & Siciliano, B. (2015). Nonlinear visual control of unmanned aerial vehicles in GPS-denied environments. *IEEE Transactions on Robotics*, *31*(4), 1004–1017.

Mebarki, R., & Siciliano, B. (2013). Velocity-free image-based control of unmanned aerial vehicles. In *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics* (pp. 1522–1527).

Mourikis, A. I., & Roumeliotis, S. I. (2007). A multi-state constraint Kalman filter for vision-aided inertial navigation. In *IEEE International Conference on Robotics and Automation* (pp. 3565–3572).

Mourikis, A. I., Trawny, N., Roumeliotis, S. I., Johnson, A. E., Ansar, A., & Matthies, L. (2009). Vision-aided inertial navigation for spacecraft entry, descent, and landing. *IEEE Transactions on Robotics*, *25*, 264–280.

Prasad, J., Calise, A. J., Johnson, E. N., Sattigeri, R., & Moon, J. (2008). Flight demonstration of an adaptive guidance controller for autonomous formation flight. In *American Helicopter Society 64th Annual Forum*.

Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. In *IEEE International Conference on Computer Vision* (pp. 2564–2571).

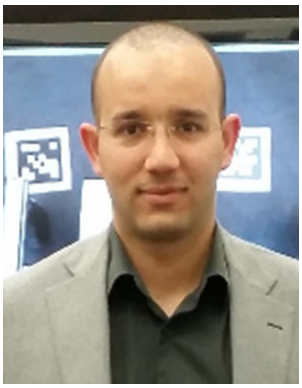
Shakernia, O., Koo, T., & Sastry, S. (1999). Landing an unmanned air vehicle: Vision based motion estimation and nonlinear control. *Asian Journal of Control*, *1*, 128–145.

Shen, S., Michael, M., & Kumar, V. (2011). Autonomous multi-floor indoor navigation with a computationally constrained MAV. In *IEEE International Conference on Robotics and Automation* (pp. 20–25).

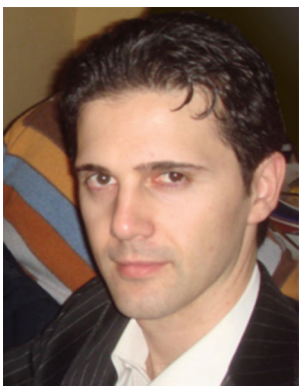
Steger, C. (1996). On the calculation of arbitrary moments of polygons. *Technical Report FGBV-96-05, Forschungsgruppe Bildverstehen (FG BV) Informatik IX*, Technische Universität München.

Weiss, S., Achtelik, M., Lynen, S., Chli, M., & Siegwart, R. (2012). Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments. In *IEEE International Conference on Robotics and Automation* (pp. 957–964).

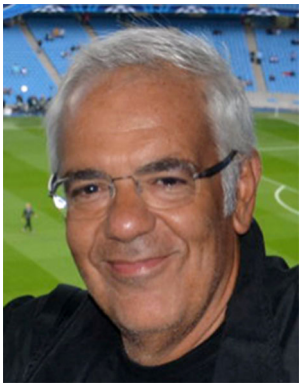
Zhao, S., Lin, F., Peng, K., Dong, X., Chen, B. M., & Lee, T. H. (2015). Vision-aided estimation of attitude, velocity, and inertial measurement bias for UAV stabilization. *Journal of Intelligent and Robotic Systems, 81*, 531–549.



Rafik Mebarki was born in Algiers, Algeria, on April 29, 1983. He received the engineering degree from the National Polytechnique School of Algeria, in 2005, the M.S. degree in automatic systems from University Paul Sabatier, Toulouse, France, in 2006, and the Ph.D. degree from University of Rennes 1, Rennes, France, in 2010. He carried out his Ph.D. works with Lagadic Team at IRISA/INRIA Rennes, on the topic of visual servoing from ultrasound images for medical robotics. From 2012 he has been a Research Fellow with PRISMA Laboratory, University of Naples Federico II, Napoli, Italy, where he worked on vision-based control and sensing for aerial robotics. His current research interests include aerial robotics, visual servoing, nonlinear control, and data fusion.



Vincenzo Lippiello was born in Naples, Italy, on June 19, 1975. He received the Laurea degree in Electronic Engineering and the Ph.D. degree in Information Engineering from the University of Naples, in 2000 and 2004, respectively. He is an Associate Professor of Automatic Control in the Department of Electrical Engineering and Information Technology, University of Naples Federico II. His research interests include visual servoing of robot manipulators, hybrid visual/force control, adaptive control, grasping and manipulation, aerial robotics, robotic ball catching, and visual object tracking and reconstruction. He has co-authored more than 90 journal and conference papers and book chapters. He is member of the IFAC Technical Committee on Robotics.



Bruno Siciliano was born in Naples, Italy, on October 27, 1959. He received the Laurea degree and the Ph.D. degree in Electronic Engineering from the University of Naples in 1982 and 1987, respectively. He is Professor of Control and Robotics, and Director of the PRISMA Lab in the Department of Electrical Engineering and Information Technology at University of Naples Federico II. His research interests include force and visual control, human-robot interaction, aerial and service robotics. He has co-authored 13 books, more than 80 journal papers, more than 250 conference papers and book chapters. He has delivered 130 invited lectures and seminars at institutions worldwide, and he has been the recipient of several awards. He is a Fellow of IEEE, ASME and IFAC. He has served on the editorial boards of several peer-reviewed journals and has been chair of program and organizing committees of several international conferences. He is Co-Editor of the Springer Tracts in Advanced Robotics, and of the Springer Handbook of Robotics, which received the PROSE Award for Excellence in Physical Sciences & Mathematics and was also the winner in the category Engineering & Technology. His group has been granted 15 European projects, including an Advanced Grant from the European Research Council. Professor Siciliano is the Past-President of the IEEE Robotics and Automation Society.