DOI: 10.48011/asba.v2i1.1656

Attitude and Position Estimation in UAVs using Artificial Landmarks and MEMS Sensors in a Virtual Environment

Abstract: This work discusses the development of a hybrid estimation algorithm based on computer vision and microelectromechanical system sensors. A mathematical enviroment was developed to simulate the dynamics of the quadrotor and its sensors, a 3D simulation software was also developed, simulating a on-board camera. The results obtained were compared to a TRIAD/MEMS attitude and position estimation technique. A fourty times increase in precision was shown, at the cost of five times additional computational processing time.

Keywords: Artificial landmarks; Attitude estimation; Position Estimation; Computer vision; Microelectromechanical systems sensors; UAV.

1. INTRODUCTION

Precise estimation of attitude and position is an unavoidable step for an efficient control in quadrotors applications. Every controller action depends on, both, the system's states and the desired target. If sensor information is not reliable, the controller will not be able to operate on its full potential.

According to Gulmammadov (2009), inertial sensors have been considered an essential navigational tool, especially in the aerospace and automotive industries and, furthermore, in robotics, Mainly due to its relative low-price and its small size. Also, according to Perlmutter and Robin (2012), for medium to high volume applications, MEMS accelerometers and gyroscopes will be dominating the market for a few decades.

Generally speaking, a quadrotor employs, at least, two MEMS sensors (Nwe et al., 2008). The first sensor is an accelerometer which is used to measure the body's acceleration and, consequently, to estimate the gravity vector. The second sensor is a gyroscope which is used to measure the body's angular velocity. Another optional sensor is a magnetometer (Chan et al., 2011) which is used to measure the earth's magnetic field at the quadrotor's position. All these sensors measurements are taken with respect to the body frame of reference.

Despite its importance, even the best MEMS sensors are liable to certain types of errors that interfere its measurements. The most common errors are white (or Gaussian) noise and bias drift, being the latter the most problematic (Gulmammadov, 2009). Some techniques may reduce the estimation error, such as Kalman filters, which is an estimation technique that might reduce white noise dependent error, however it relies on the knowledge of the sensor's variance and on consecutive measurements (Li et al., 2015).

A way to reduce the measurements degradation produced by these intrinsic errors could consider some prior knowledge of the landscape through the use of natural or artificial landmarks, i.e. a recognizable feature whose position is already known. In many mobile robot navigation problems, methods based on artificial landmarks are used to estimate position, exploiting computer vision techniques, which are capable of recognizing the landmarks and extract useful features from them (Mata et al., 2001).

Artificial landmark position estimation is useful in these scenarios as it provides absolute positioning, in contrast to MEMS sensors, which are based on relative positioning. According to Liu and Pang (2001), relative positioning determination relies on previous position measurements, while absolute positioning does not have such dependence.

2. OBJECTIVES

This work aims to propose an hybrid sensor system based on artificial landmarks identification and on-board MEMs sensors to estimate the quadrotor's attitude and position. The methodology described here employs the Python language and OpenCV library - an open-source project focused on computer vision and machine learning applications (Bradski and Kaehler, 2000) - to detect and extract useful features from an artificial landmark within a virtual environment created specifically for this purpose - which is based on a Python based engine called Panda3D (Panda3D, 2019). The estimation algorithm proposed is composed of a lower frequency computer vision algorithm, which, using the artificial landmarks, estimates the attitude and the position, and a higher frequency conventional MEMS sensing algorithm. The additional power consumption of coupling these two sensing algorithms is well compensated by a considerable reduction in state estimation error.

3. METHODOLOGY

The most important aspects of the proposed sensing system are its viability and its effectiveness. A viable option was to simulate the entire system in a computational environment, avoiding unnecessary equipment acquisition and unexpected catastrophic failures. This simulation software was developed-in house and will be briefly discussed in Section 3.4. In the following, the main aspects related to the dynamics of the quadrotor are presented.

3.1 Dynamics Simulation

The mathematical equations governing the quadrotor's dynamic are based in (de Oliveira et al., 2019). These equations include the following dynamic aspects:

- Translational and rotational drag (Eqs. (2) and (5));
- Position and velocity in the inertial frame of reference (Eq. (4));
- Propeller gyroscopic effect (Eq. (7));
- Quaternion representation for attitude (Eqs. (11)).

$$\begin{bmatrix} f_{thrust} \\ m_{thrust,x} \\ m_{thrust,y} \\ m_{thrust,z} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} f_{drag,x} \\ f_{drag,y} \\ f_{drag,z} \end{bmatrix} = -\rho C_d l D \begin{bmatrix} |\dot{x}_b|\dot{x}_b \\ |\dot{y}_b|\dot{y}_b \\ 2|\dot{z}_b|\dot{z}_b \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} f_{xb} \\ f_{yb} \\ f_{zb} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ f_{thrust} \end{bmatrix} + \begin{bmatrix} f_{drag,x} \\ f_{drag,y} \\ f_{drag,z} \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} \ddot{X} \\ \ddot{Y} \\ \ddot{Z} \end{bmatrix} = \frac{R_b^I}{M} \begin{bmatrix} f_{xb} \\ f_{yb} \\ f_{zb} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ G \end{bmatrix} \tag{4}$$

$$\begin{bmatrix} m_{drag,x} \\ m_{drag,y} \\ m_{drag,z} \end{bmatrix} = -\frac{1}{3}\rho D^4 l C_d \begin{bmatrix} |w_x|w_x \\ |w_y|w_y \\ 2|w_z|w_z \end{bmatrix} \tag{5}$$

$$w_r = -w_1 + w_2 - w_3 + w_4 \tag{6}$$

$$\begin{bmatrix} m_{gyro,x} \\ m_{gyro,y} \\ m_{gyro,z} \end{bmatrix} = I_R \omega_r \begin{bmatrix} -w_x \\ w_y \\ 0 \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} = \begin{bmatrix} m_{thrust,x} \\ m_{thrust,y} \\ m_{thrust,z} \end{bmatrix} + \begin{bmatrix} m_{gyro,x} \\ m_{gyro,y} \\ m_{gyro,z} \end{bmatrix} + \begin{bmatrix} w_{drag,x} \\ w_{drag,y} \\ w_{drag,z} \end{bmatrix} \tag{8}$$

$$\begin{bmatrix} \dot{w}_x \\ \dot{w}_y \\ \dot{w}_z \end{bmatrix} = J^{-1} \left(\begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} - \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} \times J \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} \right) \tag{9}$$

$$\tag{10}$$

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -w_x & -w_y & -w_z \\ w_x & 0 & w_z & -w_y \\ w_y & -w_z & 0 & w_x \\ w_z & w_y & -w_x & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \frac{1}{2} \bar{W} \vec{q} \tag{11}$$

where u_i and w_i are the i-th propeller thrust and angular velocity, respectively; $\dot{x}_b$, $\dot{y}_b$ and $\dot{z}_b$ are the components of the quadrotor translational velocity and w_x, w_y and w_z, the components of its angular velocity, both taken in the body frame; $\ddot{X}$, $\ddot{Y}$, $\ddot{Z}$ being the quadrotor acceleration in the inertial frame; ρ, C_d, l, D and R_b^I are, respectively, the air density, the drag coefficient, the quadrotor arm thickness, the propeller center to body center of gravity horizontal distance and the body to inertial rotation matrix (see Chou (1992) for further information on how to calculate this matrix). Knowing the quadrotor physical constants, it is possible to numerically integrate (4), (9) and (11), in order to obtain all the current states of the system.

3.2 MEMS Sensors Simulation and Conventional Estimation Algorithms

In the proposed simulation environment (Section 3.4), it is necessary to simulate the main intrinsic characteristics of MEMS sensors, i.e. the inertial measurements alongside white noise and bias drift. Those main characteristics should resemble real sensors. The proposed method is a simplified stochastic approach, defined by a perfect Gaussian noise with zero mean and a predetermined standard deviation added to the measurements. The bias drift is defined at the first iteration of the algorithm as a zero mean random variable bounded by a given limit, and added to each measurement.

From Figure 1 it is shown schematically a conventional estimation algorithm. The real (by 'real' in the virtual environment, meaning the simulated inertial measurements) angular and position states w_k and $\ddot{X}_k$ are observed by the sensors in the observed states $\dot{q}_{k,obs}$ and $\ddot{X}_{k,obs}$. Those observed states are also degraded by the noise introduced by the sensors.

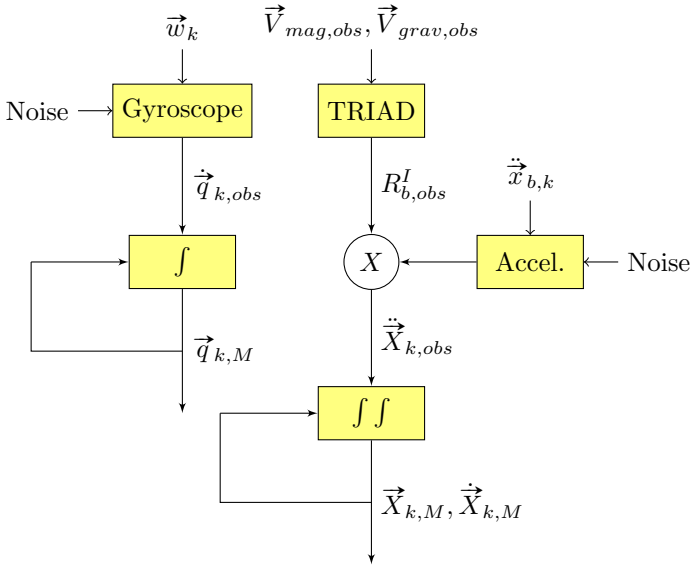

Figure 1. Conventional attitude and position estimation algorithm.

Note that this traditional algorithm is also dependent on the previously estimated states, that happens due to the numerical integration used to calculate the accelerometer and gyroscope measurements in order to obtain the current states. Any numerical integration is heavily dependent on previous results.

If the on-board sensor were capable to measure perfectly the state of the system, ideally none error would be introduced in the estimation process. However that is not the case as some noise is introduced by the sensor in the measures. As already discussed, there are normally two kinds of noise, the white noise and the bias drift. White noise, per definition, will cancel itself out over some time, as its mean is zero, or close to it. On the other hand, bias drift noise is not easily predictable, its mean is not zero and, moreover, it will affect the estimation cumulatively, as the algorithm progresses through time. In each iteration

of estimation, some bias is introduced by the sensor, being added by the integration block, causing the estimated states to drift away from the real ones.

Here, the sensor noise is defined as a stochastic process, governed by Eqs. (12)-(14). BD_i stands for Bias Drift of the i-th sensor, $D_i(k)$, and $WN_i(k)$ stands for drift and white noise of the i-th sensor, on timestep k, respectively.

$$\overrightarrow{BD}_i = \vec{\mathcal{U}}(-\sigma_{BD,i}, +\sigma_{BD,i}) \tag{12}$$

$$\vec{D}_i(k) = \vec{D}_i(k-1) + \overrightarrow{BD}_i \Delta t \tag{13}$$

$$\overrightarrow{WN}_i(k) = \vec{\mathcal{N}}(-\sigma_{WN,i}, +\sigma_{WN,i}) \tag{14}$$

The drift noise simulation process considered here consists of randomizing sensors' bias drift in the first time step of estimation, by a uniform distribution bounded by $\sigma_{BD,i}$, and adding it over the process of estimation. The white noise simulation consists of a normal distributed random variable with zero mean and a given variance of $\sigma_{WN,i}$, for each time step of estimation. Both noises are added to the real state of the system.

The TRIAD algorithm (Black, 1964) is used to estimate the attitude of a given system. This algorithm can obtain a rotation matrix of the system $R^I_{b,obs}$, from the body frame to the inertial frame, given four known vectors: two in the body frame, and two in the inertial frame. These two first vectors are provided by on-board sensors, accelerometer, and magnetometer, and the latter two vectors are known, normally the magnetic field in the area and the gravity vector, both in NEU (North, East, Up) coordinates.

With the rotation matrix $R^I_{b,obs}$ and sensors' measurements $\vec{A}_{obs}$ and $\vec{w}_{obs}$, it is possible to estimate the state of the system, Eqs. (15)-(18).

$$\ddot{\vec{X}}_{k,obs} = R^I_{b,obs}\vec{A}_{obs} \tag{15}$$

$$\dot{\vec{X}}_{k,obs} = \dot{\vec{X}}_{k-1,obs} + \ddot{\vec{X}}_{k,obs} \cdot \Delta t \tag{16}$$

$$\vec{X}_{k,obs} = \vec{X}_{k-1,obs} + \dot{\vec{X}}_{k,obs} \cdot \Delta t \tag{17}$$

$$\vec{q}_{\,k,obs} = \vec{q}_{\,k-1,obs} + \frac{1}{2}(\bar{W}_{k,obs}\,\vec{q}_{\,k,obs})\Delta t \tag{18}$$

Note that the sensors' measurements may be written as a function of the measurement noise (19)-(20).

$$\vec{A}_{obs} = \vec{A} + \overrightarrow{WN}_a(k) + \vec{D}_a(k-1) + \overrightarrow{BD}_a \cdot \Delta t \tag{19}$$

$$\vec{w}_{obs} = \vec{w} + \overrightarrow{WN}_g(k) + \vec{D}_g(k-1) + \overrightarrow{BD}_g \cdot \Delta t \tag{20}$$

The estimation drift can be obtained replacing Eqs. (19)-(20) in (15)-(18).

3.3 Quadrotor Controller

Attitude and position control in any quadrotor is not an easy task. It is a six degrees of freedom system with twelve relevant states, it is also unstable, under actuated, and nonlinear (Gupte et al., 2012). A multitude of controllers were developed to cope with those characteristics. Early attempts used PID controllers for near-linear regions of the problem (Erginer and Altug, 2007). More recently, nonlinear controllers were successful to control the system, e.g. SDRE control (de Oliveira et al., 2019) and back-stepping control (de Avellar Frederico et al., 2016). Although the controller developed for the system is not within the scope of this work, it will be mentioned briefly.

The quadrotor considered here is controlled by a neural network and its actuation limits are bounded by its propeller and electric motor pair characteristics. Given an arbitrary state S_0, the task of the controller is to conduct the system to the zero state, which means, to take the vehicle to the origin with null velocity and a neutral angular position.

The neural network controller has 20,293 variables and was trained by a deep learning algorithm known as PPO (Proximal Policy Optimization) (Schulman et al., 2017). This technique is relatively new, bringing some advantages over its predecessor, the TRPO (Trust Region Policy Optimization) algorithm (Schulman et al., 2015).

The controller training took about six hours in a computer with an Intel i5-4460 processor and 16 GB of RAM. In each training episode, the system was initialized in a random state, governed by the Eqs. (21)-(24).

$$\vec{X} = \vec{\mathcal{U}}(-2.5, 2.5)\ m \tag{21}$$

$$\dot{\vec{X}} = \vec{\mathcal{U}}(-2.5, 2.5)\ m \cdot s^{-1} \tag{22}$$

$$\begin{bmatrix}\phi\\ \theta\\ \psi\end{bmatrix} = \vec{\mathcal{U}}(-0.5, 0.5)\ rad \tag{23}$$

$$\vec{w} = \vec{\mathcal{U}}(-0.5, 0.5)\ rad \cdot s^{-1} \tag{24}$$

The goal of the training is to achieve a square sum of all the system states (position, velocity, angular position and velocity), lower than 0.01, in at least 95% of the last 100 training episodes. On Figure 2 it is possible to observe the average absolute position of the controlled quadcopter over a span of 30 episodes, lighter color represents the variance. The initial state was governed by Eqs. (21)-(24), the average steady state error was 0.005 meters, the average 5% settling time was 5.24 seconds.

3.4 Three Dimensional Environment Simulation

A mathematical environment was developed in-house, aiming to include all the quadrotor and its sensors dynamics. This mathematical environment is able to determine the quadrotor states and sensors readings over the simulation time. On top of this mathematical simulation a 3D representation software was developed, this software is able to simulate a on-board camera of said quadrotor. This 3D representation was implemented using a Python 3D engine called Panda3D, which is commonly used for the development of digital games (Panda3D, 2019). In the 3D environment developed, it is possible to simulate the quadrotor in any given scenario, with all the quadrotor's dynamics inherited directly from its equations of motion.

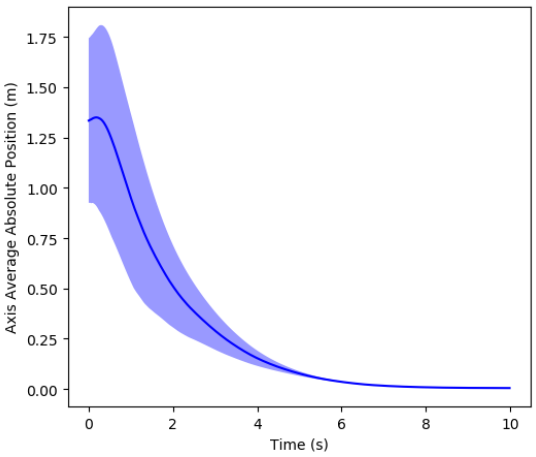

Figure 2. Controlled quadrotor all axis average absolute position.

Figure 3 shows a screen-shoot of the developed 3D environment.

Figure 3. Render of the 3D environment.

It was also necessary to emulate an on-board camera, which was responsible for capturing images of the 3D environment, pointed to the negative z-axis of the quadrotor body frame. One example of these photos is represented in Figure 4. In each simulation time step, a photo is taken and then sent to the OpenCV based algorithm.

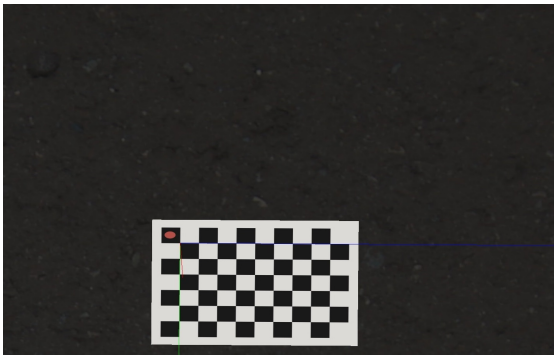

Figure 4. Render of the on-board camera.

3.5 Camera Calibration

According to Weng et al. (1992), the camera calibration "(...) *is crucial for applications that involve quantitative measurements such as dimensional measurements, depth from stereoscope, or motion from images*". Only with a calibrated camera, it is possible to determine various relationships between an object and its picture. The camera calibration process consists mainly in determining the camera intrinsic parameters matrix, Eq. (25), and five lens distortion coefficients, Eq. (26) (Zhang, 2000).

$$mtx_{cam} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{25}$$

$$dist_{cam} = [k_1 \;\; k_2 \;\; p_1 \;\; p_2 \;\; k_3] \tag{26}$$

The calibration algorithm used is part of the OpenCV's standard library, and it is based on (Zhang, 2000). This algorithm uses a given number of distinct photos of a chessboard pattern, all of them in different angles and distances. It is possible to observe one of those poses in Figure 5.

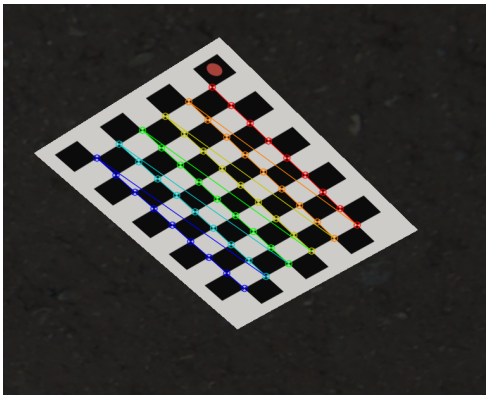

Figure 5. Calibration pose example.

Given the pictures, the algorithm *findChessboardCorners* detects the chessboard vertices in each of the images. All of those obtained vertices are then fed to another algorithm called *calibrateCamera*, which generates the camera intrinsic parameters matrix, based on knowledge of the real size of the chessboard pattern (OpenCV, 2019). The emulated on-board camera is considered perfectly planar, i.e there are no lens distortions, as lens distortion is only possible on a real lens with optical glasses, leading all coefficients of Eq. (26) to zero.

3.6 Attitude and Position Estimation Algorithm Based on Artificial Landmarks

The artificial landmark used in this algorithm is the same chessboard pattern used to calibrate the camera, illustrated in Figure 5. It is important to note that various types of patterns can be used as artificial landmarks, e.g. QR codes. The pattern, that just needs to be recognizable by some algorithm, has a defined size and spacing, and it is advisable not to use an ambiguous pattern, i.e. the pattern must look different when rotated 90° or 180°.

To estimate the quadrotor attitude and position, the algorithm *findChessboardCorners* is used once again, as the vertices of the found pattern are used by OpenCV's *solvepnp* algorithm to determine the pattern rotation axis

and transform vector (OpenCV, 2019). For a given image, if there is a chessboard pattern in it, the first algorithm will return its vertices position. Once the camera is calibrated, the later algorithm will return the pattern position on the camera's coordinate system, and its axis of rotation. This information is then transformed into the appropriate coordinate system.

Coordinate system transformations are necessary for several reasons. First of all, the default coordinate system of OpenCV is right-handed, with its origin defined at the top left of the image, x-axis to the right, y-axis down, and z-axis from to plane of the image inward. This coordinate system is also known as east, south, down (ESD), different from the coordinate system used in this work, which is north, east and up (NEU).

A second set of transformations is necessary to obtain the quadrotor position and attitude with respect to the inertial coordinate system, as the OpenCV's algorithm returns the position and attitude of the pattern in relation to the camera. The sequence of transformations needed to rotate the coordinate system from ESD to NEU and from the camera's coordinate system to the inertial coordinate system are defined by Eqs. (27)-(30):

$$\begin{bmatrix}\phi_{cam}\\ \theta_{cam}\\ \psi_{cam}\end{bmatrix} = Euler_{321}(rvecs), \tag{27}$$

$$\begin{bmatrix}\phi_{est}\\ \theta_{est}\\ \psi_{est}\end{bmatrix} = \begin{bmatrix}-\phi_{cam}\\ -\theta_{cam}\\ \psi_{cam}\end{bmatrix}, \tag{28}$$

$$R_c^I = R_{321}\left(\psi_{cam}, \theta_{cam}, \phi_{cam}\right), \tag{29}$$

$$\begin{bmatrix}x_{est}\\ y_{est}\\ z_{est}\end{bmatrix} = \begin{bmatrix}-1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & -1\end{bmatrix} R_c^I \begin{bmatrix}tvecs_x\\ tvecs_y\\ tvecs_z\end{bmatrix}, \tag{30}$$

where ϕ_{cam}, θ_{cam} and ψ_{cam} are the Euler angles of the object, in the camera's coordinate system; $Euler_{321}$ is an Euler function transformation from a rotation vector to Euler angles; ϕ_{est}, θ_{est} and ψ_{est} are the estimated Euler angles of the quadrotor in the artificial landmark/inertial coordinate system; R_c^I is the rotation matrix from camera to pattern coordinate system and x_{est}, y_{est} and z_{est} are the estimated positions of the camera in the pattern's coordinate system.

One characteristic of this algorithm is its heavy computational cost. Determining if a given pattern is present in any image and finding its vertices is a pixel by pixel effort. Due to that, even with the power of modern computers, the computational task may result in slower computational time than the controller time step. Another characteristic of this task is its complete dependency on the artificial landmark being present in the photo taken by the on-board camera.

Because of these two characteristics, it is not possible to state that this algorithm may completely replace all on-board sensors of a quadrotor, it is rather possible to state that this technique should be coupled with other estimation techniques, aiming to improve the state estimation without compromising the total computational cost.

3.7 Hybrid Sensing System

A hybrid sensing system, using both, on-board MEMS sensors and the estimation algorithm based on artificial landmarks, is the ideal trade-off between computational power and reliability. This system is significantly faster than a conventional artificial landmark algorithm because it runs at a slower frequency, requiring fewer calculations per second. It is also reliable since if any absent detection occurs, e.g., the pattern not being in the frame, or being poorly lit by a passing shadow, the conventional sensing system will be still running normally and flying the quadrotor stably. Moreover, as discussed previously, MEMS sensors tend to generate drift over time, however these type of sensor are precise for short periods. Due to this fact, the hybrid sensing system also maintains almost the same precision of the artificial landmark algorithm.

The proposed system acts directly on Eqs.(15)-(17), dampening the previous MEMS estimated state with the estimated state obtained by the artificial landmark algorithm, as defined by Eqs. (31) and (33).

$$\vec{X}_{k,obs} = \left[(1-\sigma_h)\vec{X}_{obs} + \sigma_h\vec{X}_{img}\right]_{k-1} + \ddot{\vec{X}}_{k,obs}\Delta t \tag{31}$$

$$\vec{X}_{k,obs} = \left[(1-\sigma_h)\vec{X}_{obs} + \sigma_h\vec{X}_{img}\right]_{k-1} + \dot{\vec{X}}_{k,obs}\Delta t \tag{32}$$

$$\vec{q}_{k,obs} = \left[(1-\sigma_h)\vec{q}_{obs} + \sigma_h\vec{q}_{img}\right]_{k-1} + \dot{\vec{q}}_{k,obs}\Delta t \tag{33}$$

where σ_h is a given dampening factor, $\dot{\vec{X}}_{img}$, $\vec{X}_{img}$, and $\vec{q}_{img}$ the computer vision estimated velocity, position and angular position, respectively.

Figures 6 and 7 show the block diagrams of the proposed sensing system. Both diagrams are only valid in the iteration on which occurs the artificial landmark estimation, all other iterations in between those, i.e all the iterations out of phase relative to the computer vision algorithm, occurs solely based on MEMS sensors, governed by the block diagram depicted in Figure 1.

3.8 Model Parameters

The given parameters used in the simulation can be observed from Tables 1 and 2. The on-board camera was setup to a sensor size of 36 by 24 millimeters, and a focal length of 45 millimeters. The 3D scenario was carefully created to be represented in real scale, compared to the quadrotor size. Each inner edge of the chessboard pattern has 10 centimeters, the total area of the chessboard pattern represents the same area of an A2 size sheet of paper.

The 3D render was done in 1920 by 1080 pixels of resolution, an 16x anisotropic texture filter was set and texture quality configured to slow. This setup was necessary to keep a reliable texture and shape representation of the 3D environment, mainly of the chessboard pattern, as lower resolution or faster filters resulted in blurry images, hampering the ability of the sensing system to detect the vertices of the pattern.

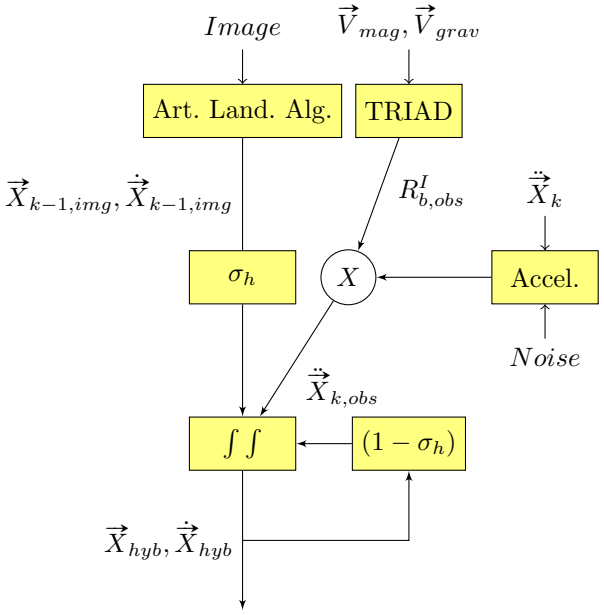

Figure 6. Hybrid sensing system: accelerometer algorithm.

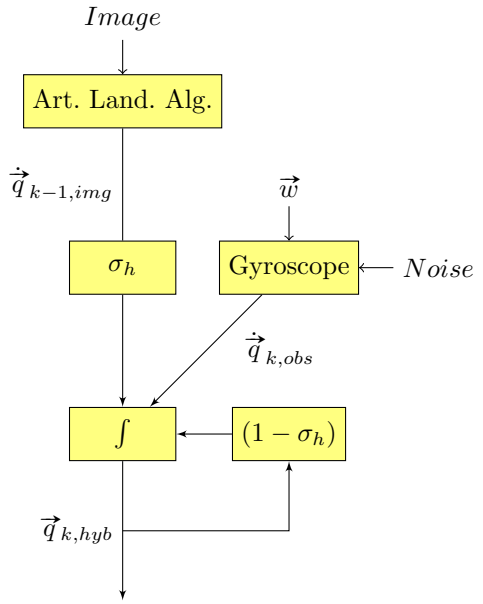

Figure 7. Hybrid sensing system: gyroscope Algorithm

4. RESULTS

From Tables 3-4 it is possible to observe the error behavior of the conventional MEMS sensing system and the proposed hybrid system; where error is defined as the absolute difference between the estimated and simulated state. The average was taken over 120,000 samples, throughout fourty simulation episodes.

In the results presented in the Table 3, the quadrotor was set initially at five meters height and all other states equal zero. The goal of this test was just to keep the quadrotor still in a hovering flight at that height. Table 4 considers that the quadrotor was initially set in a random initial state bounded by Eqs. (21), (23),(24), the initial velocity was bounded by equation $\dot{\vec{x}} = \vec{u}(-1.25, 1.25)\ m \cdot s^{-1}$, the quadrotor goal was to reach five meters height on z axis, with all other states being null.

Table 1. Quadrotor parameters.

System Frequency	f_s	100Hz
Propeller center to CG horizontal distance	D	$0.26m$
Arm Thickness	l	$0.05m$
Mass	m	$1.03kg$
X inertia moment	J_{xx}	$16.83 \cdot 10^{-3} kgm^{-2}$
Y inertia moment	J_{yy}	$16.83 \cdot 10^{-3} kgm^{-2}$
Z inertia moment	J_{zz}	$28.34 \cdot 10^{-3} kgm^{-2}$
Propeller inertia moment	I_r	$5 \cdot 10^{-5} kgm^{-2}$
Propeller Thrust Coeff.	K_f	$1.4 \cdot 10^{-5}\ Ns^2rad^{-2}$
Propeller Torque Coeff.	K_m	$2.4 \cdot 10^{-7} Nms^2rad^{-2}$
Body Drag Coefficient	C_d	1.1
MEMS Sensing Frequency	f_{MEMS}	100Hz
Hybrid Sensing Frequency	f_h	10Hz
Hybrid Sensing Damping	σ_h	0.2

Table 2. Sensors parameters.

Sensor	**Model Parameters**	
	Standard Deviation	Max. Bias Drift
Accelerometer	$0.1m/s^2$	$0.0005m/s^2$
Gyroscope	$35mrad/s$	$0.15mrad/s$
Magnetometer	$15mG$	$0.075mG$

Table 3. Average absolute estimation error in 30 seconds hover flight.

Sensing	**Average Estimation Error**	
System	**Position (cm)**	**Quaternion (10^{-3})**
Hybrid	1.05 ± 0.01	2.45 ± 0.01
MEMS	43.0 ± 31.2	9.99 ± 0.17

Table 4. Average absolute estimation error in random initial state flight.

Sensing	**Average Estimation Error**	
System	**Position (cm)**	**Quaternion (10^{-3})**
Hybrid	1.39 ± 0.03	2.43 ± 0.01
MEMS	47.5 ± 33.2	9.72 ± 0.16

Figures 8-9 show the average position error over time. This average was taken over 40 episodes and the lighter curve represents its variance. From Figures 10 and 11 it is possible to observe the quadrotor average position over time, taken over 40 episodes, the lighter curve being its variance.

The results obtained with the proposed hybrid system were better, comparing to the conventional MEMS sensing approach. The average error was 37 times lower in the hybrid system, with the absolute precision of 1.22 centimeters, while the conventional MEMS algorithm had 45.25 centimeters of absolute precision. The hybrid approach for estimation of angular position was also better, although not as much, netting 4.03 times lower error in comparison to the conventional approach.

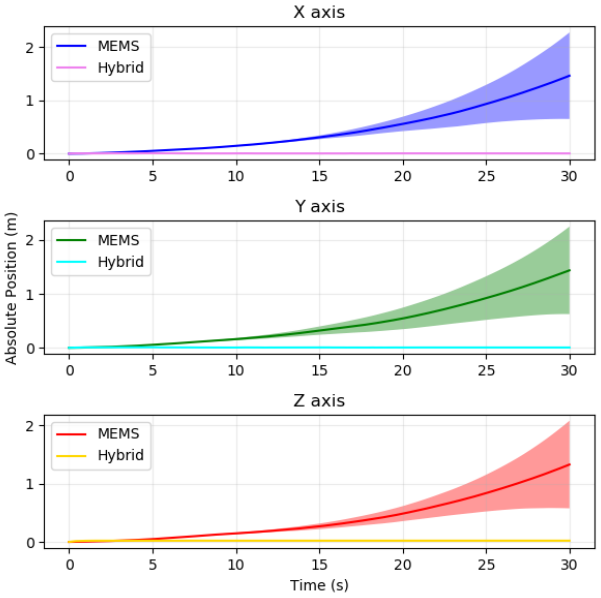

Figure 8. Hover flight average absolute estimation error over time.

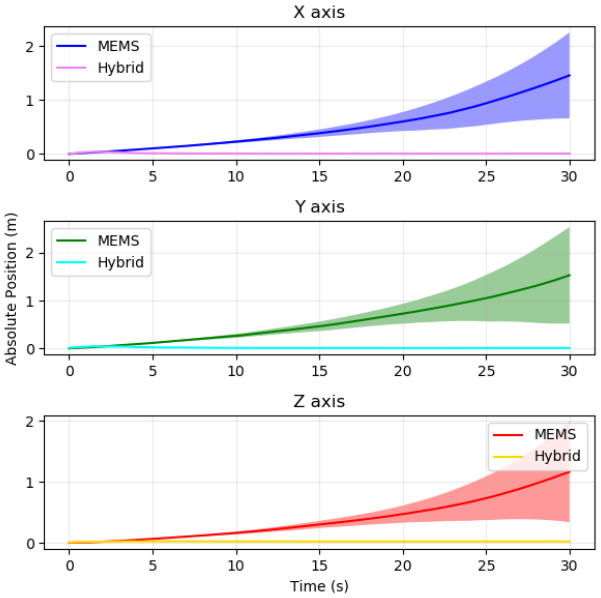

Figure 9. Random initial state flight average absolute estimation error over time.

From Table 5 it is possible to compare the computational time of the conventional and hybrid approach. The hybrid system computational time, on 10,000 samples average, was three times slower on hover flight and seven times slower on the random initial state, in comparison to the conventional approach.

Table 5. Average iteration computational time.

Flight	**Iteration Computational Time**		
Mode	**Art. Land.**	***MEMS***	**Hybrid**
Hover Flight	22.9 ms	1.16 ms	3.45 ms
Random State	69.6 ms	1.16 ms	8.12 ms

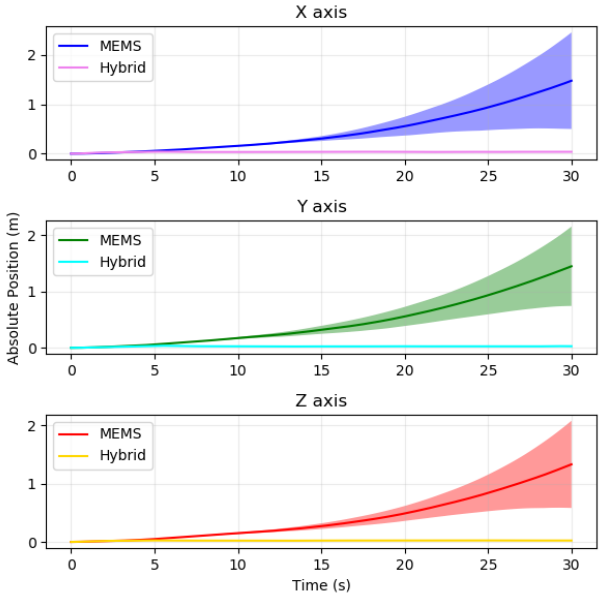

Figure 10. Hover flight average absolute position over time.

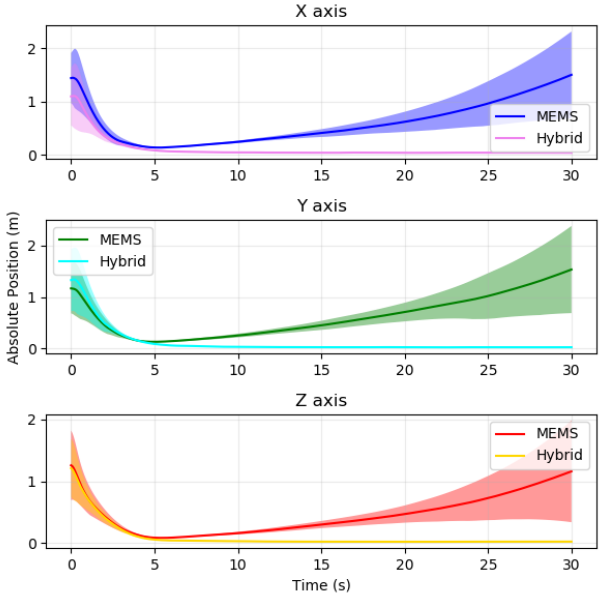

Figure 11. Random initial state flight average absolute position over time.

The main cause of the longer average iteration time in the random initial state flight mode is probably due to the *findChessboardCorners* algorithm because, if there is no artificial landmark present on the image (which frequently occurs in the first moments of the flight with randon initial state), the algorithm checks all the pixels of the image before deciding that there is no artificial landmark in that picture. On the other hand, if there is a landmark on the image, the algorithm only scans until it finds it, as the pattern is normally in the middle of the image, the iteration time is reduced.

The effective time of one iteration of the artificial landmark estimation is, on average, 40 times slower compared to conventional MEMS sensing, but as its frequency is 10

times slower, the hybrid approach iteration time is diluted in the out of phase iterations, i.e the iterations in which only the MEMS estimation occurs, resulting on average 5.8 milliseconds per iteration.

5. CONCLUSIONS

The proposed hybrid sensing system attained good results, providing, approximately, fourty times more accurate on position and four times more accurate on attitude, eliminating the bias drift effect of the conventional MEMS sensing algorithm. Its downside is the computational time required, being, on average, five times slower compared to the conventional algorithm.

An interesting approach that should be investigated later would be to replace the standard chessboard detection algorithm with a faster alternative, possibly based on deep learning and convolutional networks. A faster detection algorithm would probably generate even better results, creating the possibility of on-board and offline use, since quadrotors on-board computers are normally slower in comparison to desktop PCs.

6. ACKNOWLEDGMENTS

This study was financed in part by CAPES.

REFERENCES

Black, H.D. (1964). A passive system for determining the attitude of a satellite. AIAA journal, 2(7), 1350–1351.

Bradski, G. and Kaehler, A. (2000). Opencv. Dr. Dobb's journal of software tools, 3.

Chan, A.L., Tan, S.L., and Kwek, C.L. (2011). Sensor data fusion for attitude stabilization in a low cost quadrotor system. In 2011 IEEE 15th International Symposium on Consumer Electronics (ISCE), 34–39. IEEE.

Chou, J.C. (1992). Quaternion kinematic and dynamic differential equations. IEEE Transactions on robotics and automation, 8(1), 53–64.

de Avellar Frederico, L., da Silva, A.L., and Martins-Filho, L. (2016). Dinâmica e controle do voo de um vant quadrirrotor. Proceeding Series of the Brazilian Society of Computational and Applied Mathematics, 4(1).

de Oliveira, A.C.F., Altuna, J.A.T., and Correa, D.P.F. (2019). Dynamic modelling and control of unmanned aerial vehicle of the quadrotor type. In 25th ABCM International Congress of Mechanical Engineering.

Erginer, B. and Altug, E. (2007). Modeling and pd control of a quadrotor vtol vehicle. In 2007 IEEE Intelligent Vehicles Symposium, 894–899. IEEE.

Gulmammadov, F. (2009). Analysis, modeling and compensation of bias drift in mems inertial sensors. In 2009 4th International Conference on Recent Advances in Space Technologies, 591–596. IEEE.

Gupte, S., Mohandas, P., and Conrad, J. (2012). A survey of quadrotor unmanned aerial vehicles. Proceedings of IEEE Southeastcon.

Li, Q., Li, R., Ji, K., and Dai, W. (2015). Kalman filter and its application. In 2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS), 74–77. IEEE.

Liu, H.H. and Pang, G.K. (2001). Accelerometer for mobile robot positioning. IEEE Transactions on Industry Applications, 37(3), 812–819.

Mata, M., Armingol, J.M., de la Escalera, A., and Salichs, M.A. (2001). A visual landmark recognition system for topological navigation of mobile robots. In Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164), volume 2, 1124–1129. IEEE.

Nwe, T.T., Htike, T., Mon, K.M., Naing, Z.M., and Myint, Y.M. (2008). Application of an inertial navigation system to the quad-rotor uav using mems sensors. Engineering and Technology, 42, 578–582.

OpenCV (2019). Camera Calibration and 3D Reconstruction. URL `https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html`. (accessed March 18, 2020).

Panda3D (2019). Panda3D - A Python 3D Engine. URL `https://docs.panda3d.org/1.10/python/introduction/index`. (accessed March 18, 2020).

Perlmutter, M. and Robin, L. (2012). High-performance, low cost inertial mems: A market in motion! In Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium, 225–229. IEEE.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. arxiv 2017. arXiv preprint arXiv:1707.06347.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In International conference on machine learning, 1889–1897.

Weng, J., Cohen, P., and Herniou, M. (1992). Camera calibration with distortion models and accuracy evaluation. IEEE Transactions on Pattern Analysis & Machine Intelligence, (10), 965–980.

Zhang, Z. (2000). A flexible new technique for camera calibration. IEEE Transactions on pattern analysis and machine intelligence, 22(11), 1330–1334.