Quadcopter attitude and position estimation using UWB technology and low-cost IMU module

Matteo Albi
dept. Industrial Engineering
University of Trento
Trento, Italy
matteo.albi@studenti.unitn.it

***Abstract*—This paper presents a possible approach to estimate the position and attitude of quadcopter drones for indoor applications.**

The estimation algorithm is composed of two Extended Kalman Filters (EKFs). The first filter estimates the attitude and the external accelerations the drone is subject to. The second filter uses the information of the first one and the UWB module to estimate the position.

The paper presents both simulated and experimental results. In the first case, proper data generation is performed, starting from the ground truth and then applying the noise to which the sensors are subject. In the second case, a motion capture system is used as ground truth.

***Index Terms*—EKF, UWB, IMU, drone estimation, indoor positioning**

I. Introduction

Unmanned Aerial Vehicles (UAVs), also known as drones, are increasingly being used in a wide range of industries and applications, including indoor environments such as manufacturing facilities and warehouses [1], [2]. One of the key benefits of UAVs is their ability to operate in areas that are difficult or dangerous for humans to access. This makes them well-suited for tasks such as inspection, surveying, and monitoring in manufacturing and industrial settings.

For these applications, a precise and real-time positioning system is necessary. Many UAVs mount a Global Positioning System (GPS) to estimate their location [1], [2]. However, this is not feasible for indoor applications because GPS signals can't penetrate solid objects like walls and ceilings. Thus, Ultra-Wide-Band (UWB) communication ia a promising solution for the indoor-positioning problem [3], [4].

UWB (Ultra-Wideband) technology is a wireless communication protocol that uses a wide frequency band to transmit data over short distances at very high speeds while consuming very low power. By using this communication protocol, it is possible accurately measure the time-of-flight (TOF) of signals, and use it for ranging and positioning applications.

Nevertheless, to achieve a robust estimation, an Inertial Measurement Unit (IMU) is used to retrieve the attitude of the drone and the external accelerations it is subject to. Using this information, it is possible to perform a dead-reckoning positioning estimation, which is then fused with the UWB data to reach higher precision. To fuse the data coming from the sensor, an Extended Kalman Filter (EKF) has been designed. EKF is a model-based estimation algorithm, derived from the Kalman Filter (KF), used for non-linear systems, which aims to minimize the estimation uncertainty using proper uncertainty propagation and information from several sources.

The paper is organised as follows. All previous knowledge required to understand the proposed solution is described in section II: quaternions, dynamics of quadcopters, Decentralised Time Difference of Arrival (DTDoA) positioning algorithm based on UWB technology, KF and EKF. In section III the problem formulation and the proposed solution are presented, followed by the simulation and experimental results in section V, where an A-type uncertainty analysis is performed.

II. Previous knowledge

A. Quaternions

Any rotation in the 3D space can be represented by the vector around which the rotation is performed, and the rotation angle. These two elements are summarized in quaternions, formulated by William Rowan Hamilton in 1843. They use four values to represent rotations in the 3D space: the first term represents the scalar part, corresponding to the angle of the performed rotation, and the other three terms correspond to the three components of the rotation vector. The relation between rotation vector and rotation angle are explained in eq. 1.

$$q = \begin{bmatrix} cos(\theta/2) \\ u \cdot sin(\theta/2) \\ v \cdot sin(\theta/2) \\ w \cdot sin(\theta/2) \end{bmatrix} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{1}$$

where θ is the rotation angle, and $[u, v, w]^T$ is the unit vector representing the rotation. Note that using this definition, the quaternion has unitary norm.

Given a unit quaternion, the corresponding rotation matrix is defined by eq. 2.

$$[R](q) = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2-q_3q_0) & 2(q_1q_3+q_2q_0) \\ 2(q_1q_2+q_3q_0) & 1-2(q_1^2+q_3^2) & 2(q_2q_3-q_1q_0) \\ 2(q_1q_3-q_2q_0) & 2(q_2q_3+q_1q_0) & 1-2(q_1^2+q_2^2) \end{bmatrix} \tag{2}$$

In this paper, Euler angles are referred to using the Roll, Pitch, Yaw (RPY: Φ, Θ, Ψ) convention, corresponding to rotations around the X, Y, and Z axis. When using Euler angles to describe the attitude, mainly two problems raises: the discontinuity of the angles (between $\pm\pi$) and the presence of singular configurations known as Gimbal Lock.

The advantage of using quaternions to describe the attitude of a rigid body is that they are a continuous function of the body attitude and they don't present any singular configuration. However, it is important to remember that there is an underdetermination problem, given that $q = -q$ (opposite quaternions describe the same rotation).

B. Dynamics of quadcopters

Considering the quadcopter as a rigid body, the Newton-Euler equations describing its dynamics are reported in eq. 3 and 4. Equations are reported with all vectors projected in the global reference frame, according to the North-East-Down (NED) convention.

$$\begin{cases} \dot{p} = v \\ m\dot{v} = mge_3 - [R](q)[0,0,T]^T + f_{EXT} \end{cases} \tag{3}$$

$$\begin{cases} \dot{q} = \frac{1}{2}\begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} q = \frac{1}{2}S(\omega)q \\ J\dot{\omega} = -\omega \times J\omega + \tau_{EXT} \end{cases} \tag{4}$$

Where p, q, v and $\omega = [\omega_x, \omega_y, \omega_z]^T$ are the position, the quaternion attitude, the velocity and the angular velocity of the drone, respectively. m and J are the quadcopter mass and inertia tensor respectively, g is the gravity acceleration, $e_3 = [0,0,1]^T$ is the vertical versor in the global reference frame. T is the motors vertical trust, f_{EXT} and τ_{EXT} represent respectively all contributions from external forces and torques.

It is important to notice that the attitude dynamic is independent, while the position dynamic depends on the attitude ($[R](q)$ term).

C. DTDoA algorithm based on UWB technology

UWB technology is often used in Real-Time Location Systems (RTLS) to precisely track the location of people or objects in real-time. RTLS typically consists of two main components: tags and anchors. Tags are small devices attached to the objects or people being tracked, while anchors are fixed points in the environment.

As explained in [3], the DTDoA offers a scalable infrastructure able to retrieve the position of ideally an infinite number of tags. The idea relies on the anchors being the active systems sending time-stamped messages, and the tags receiving them and computing their position. Thus, the tags will know their position in the environment, while the anchors will not register the position of the tags.

D. KF

The Kalman Filter is a model-based estimation algorithm which, under the assumptions of linear system subject to zero-mean, white and gaussian noise, is proved to be an optimal estimator for that system.

Considering a generic discrete linear system (eq. 5):

$$\bar{x}_{k+1} = Ax_k + Bu_k + G\nu_k \tag{5}$$

Where x_k is the state of the system. u_k is the system input the estimator is able to retrieve, which uncertainty is described by the covariance matrix Q. ν_k is the noise which the system is subject to, modelled as $\sim N(0, R_\nu)$. A, B, G are the state, input and noise matrices, respectively. All variables are evaluated at step k. $\bar{x}_{k+1}$ is the state of the system at step $k+1$ subject to uncertainty. In this discussion, x_k will identify the real state of the system and $\hat{x}_k$ its estimated state, which uncertainty is described by the covariance matrix P_k.

By knowing u_k, under the assumption of zero-mean noise, the filter predicts the next state x^-_{k+1} and the respective covariance matrix P^-_{k+1} as reported in eq. 6:

$$\begin{cases} \hat{x}^-_{k+1} = A\hat{x}_k + Bu_k \\ P^-_{k+1} = AP_kA^T + BQB^T + GR_\nu G^T \end{cases} \tag{6}$$

This is called *Prediction step* and exploits the model of the system to compute a first guess of its next state.

Now let's consider a generic linear measurement system as described in eq. 7:

$$\bar{z}_k = Hx_k + \epsilon_k \tag{7}$$

Where $\bar{z_k}$ is the measurement value subject to noise, H_k is the measurement matrix and ϵ_k is the noise acting on the measurement system, modelled as $\sim N(0, R_\epsilon)$.

Using this information, the KF perform an *Update step* as reported by eq. 8:

$$\begin{aligned} &Inn_{k+1} = z_{k+1} - H\hat{x}^-_{k+1} \\ &S_{k+1} = HP^-_{k+1}H^T + R_\epsilon \\ &W_{k+1} = P^-_{k+1}H^TS^{-1}_{k+1} \\ &\begin{cases} \hat{x}_{k+1} = \hat{x}^-_{k+1} + W_{k+1}Inn_{k+1} \\ P_{k+1} = (I - W_{k+1}H)P^-_{k+1} \end{cases} \end{aligned} \tag{8}$$

Where Inn_{k+1} is the innovation, which is the difference between real measurement and expected measurement given the predicted state (mapped into measurement space by matrix H). S_{k+1} is the innovation covariance matrix. Then the algorithm computes the *Kalman gain* W_{k+1} combining the prediction uncertainty and the innovation uncertainty. This matrix fuses the data from prediction and measurements as shown in the last two equations, minimizing the estimation uncertainty.

The update step can be seen as a weighted sum between prediction and measurement. Indeed, assuming state and measurement have unitary dimension, if $P^-_{k+1} \gg S_{k+1}$, then W_{k+1} is close to one, and the updated state is equal to the measurement (mapped in state space); vice-versa, if $P^-_{k+1} \ll S_{k+1}$, then W_{k+1} is close to zero, and the updated state is equal to the predicted state.

E. EKF

As mentioned in sec. II-D, the KF assumes that the system is represented by a linear model. However, this is often not the case; indeed drones have a non-linear dynamic, thus the KF is not directly applicable. Let's examine a generic discrete, non-linear system as described in eq. 9:

$$\begin{aligned} \bar{x}_{k+1} &= f(x_k, u_k, \nu_k) \\ \bar{z}_k &= h(x_k) + \epsilon_k \end{aligned} \tag{9}$$

Where f denotes the non-linear state function and h is the non-linear map from state space to measurement space.

To compute the mean value of x_k and z_k, f and h can be directly used as these values represent single points in the state and measurement spaces. However, to propagate their covariance, a linear mapping is mandatory, otherwise the Gaussian representing x_k and z_k would not be correctly propagated (a linear map always maps a Gaussian probability density function (PDF) into another Gaussian PDF). One possible solution is linearizing the functions f and h in order to obtain their approximated linear version. Adopting this idea, and under the hypothesis of zero-mean, Gaussian white noises, the resulting algorithm is the EKF, which is described by eq. 10 to 13:

- Prediction step:

$$\begin{cases} \hat{x}^-_{k+1} = f(\hat{x}_k, u_k, 0) \\ P^-_{k+1} = A_k P_k A_k^T + B_k Q B_k^T + G_k R_\nu G_k^T \end{cases} \tag{10}$$

  where:

$$\begin{aligned} A_k &= \frac{\delta f}{\delta x_k}\Big|_{\substack{u_k \\ x_k=\hat{x}_k \\ \nu_k=0}} \\ B_k &= \frac{\delta f}{\delta u_k}\Big|_{\substack{u_k \\ x_k=\hat{x}_k \\ \nu_k=0}} \\ G_k &= \frac{\delta f}{\delta \nu_k}\Big|_{\substack{u_k \\ x_k=\hat{x}_k \\ \nu_k=0}} \end{aligned} \tag{11}$$

- Update step:

$$\begin{aligned} Inn_{k+1} &= z_{k+1} - h(\hat{x}^-_{k+1}, 0) \\ S_{k+1} &= H_{k+1} P^-_{k+1} H^T_{k+1} + R_\epsilon \\ W_{k+1} &= P^-_{k+1} H^T_{k+1} / S_{k+1} \\ &\begin{cases} \hat{x}_{k+1} = \hat{x}^-_{k+1} + W_{k+1} Inn_{k+1} \\ P_{k+1} = (I - W_{k+1} H_{k+1}) P^-_{k+1} \end{cases} \end{aligned} \tag{12}$$

  where:

$$H_k = \frac{\delta h}{\delta x_k}\Big|_{\substack{x_k=\hat{x}_k \\ \epsilon_k=0}} \tag{13}$$

III. Problem formulation and proposed solution

A. Problem formulation

To develop the estimation system, measurements from an accelerometer, magnetometer and gyroscope sensors (assumed to be mounted approximately on the centre of mass of the quadcopter), denoted acc_k, mag_k, ω_k respectively, are available. Furthermore, a UWB infrastructure made of six anchors and one tag is already set up and programmed, with the tag device able to receive the messages from the anchors and save the related receiving instant timestamp. The tag is mounted on the drone as well.

The project goal is to develop an algorithm that can estimate the UAV's position and attitude in real-time, with an uncertainty sufficiently small.

B. Algorithm mathematical description

As explained in sec. II-B, the quadcopter attitude dynamic is independent, thus the first step in the solution will be to create an estimator for the attitude of the drone. An EKF has been developed for this purpose. After that, another EKF will estimate the position of the quadcopter by combining the information from the attitude and the sensor readings.

1) Accelerations decoupling: One main concern when approaching this problem is the accelerometer readings, because the sensor will measure both gravity acceleration and other external accelerations. The first is needed to determine the attitude of the drone, the latter must be integrated to estimate its position. Thus, it is necessary to decouple the two components.

One possible approach can be derived by investigating the quadcopter dynamic. The accelerometer measures the accelerations the quadcopter is subject to in the drone reference frame. Assuming that in indoor environments external forces are negligible, the accelerometer values at instant k (denoted as $acc_k = [acc_x, acc_y, acc_z]$) can be described by rewriting eq. 3 in the mobile reference frame, obtaining eq. 14. It can be observed that gravity acceleration is the only contribution having components on the x and y axis. Thus, it is possible to compute the gravity acceleration contribution as described in eq. 15.

$$\dot{v}_b = acc_k = g[R]^T(q)e_3 - [0, 0, T/m]^T \tag{14}$$

$$acc_g = \begin{bmatrix} acc_x/g \\ acc_y/g \\ \sqrt{g - acc_x^2 - acc_y^2}/g \end{bmatrix} \tag{15}$$

It is also important to mention that many studies tried to estimate the external accelerations using a model-free algorithm. This allows the creation of a general-purpose algorithm, applicable to any system. One of the most recent papers proposing this solution is [5]. In this research, the external accelerations are modelled as a first-order low-pass filtered White noise process. The researchers propose to correct the accelerometer readings with the estimated external accelerations and to properly propagate and estimate the uncertainties of the resulting values. Then, they perform the update step and use the new computed attitude to decouple the accelerations and update the external accelerations estimation.

2) Gyroscope bias estimation: Gyroscope drift is a prevalent problem occurring when integrating the gyroscope to predict the attitude of a body [6]. MEMs gyroscopes are subject to a variable variance due to several conditions (mainly the

temperature), which makes it challenging to model. However, by enlarging the state vector of the system (denoted as $\hat{X}_{att}$) by adding three more variables representing the gyroscope bias on each axis, the EKF is able to estimate it. Hence, it can be subtracted from the gyroscope readings to avoid drift. Using this method, the attitude of the quadcopter is represented by the seven-elements vector $\hat{X}_{att} = [\hat{q}, \hat{g}_b]$, with $\hat{q}$ being a four-elements vector denoting the drone attitude, and $\hat{g}_b$ being a three-elements vector denoting the gyroscope bias. Thus, the attitude uncertainty (denoted as P_{att}) is a seven-by-seven matrix.

In the prediction phase, the gyroscope bias is modelled as a zero-bias random walk, thus the estimated value remains unchanged. Then, when propagating the uncertainties (as described in section III-B4), P_{att} will register a correlation between $\hat{q}$ and $\hat{g}_b$. This correlation will affect the update step, allowing the EKF to correct the $\hat{g}_b$ estimation.

3) Attitude prediction: The prediction step in eq. 16 is obtained by discretizing eq. 4, with ω being measured by the gyroscope.

$$\hat{q}_{k+1}^{-} = \left(\frac{T_S}{2} S(\omega_k - \hat{g}_b) + I_4\right) \hat{q}_k = A(\omega_k - \hat{g}_b)\hat{q}_k \quad (16)$$

Where T_S is the step period of the algorithm and I_4 is the identity matrix of dimensions 4×4. Eq. 16 is not linear, thus the uncertainty propagation must be performed as described in eq. 10 and 11. Moreover, being $\hat{g}_b$ modelled as a zero-bias random walk (as described in sec. III-B2), it is necessary to increase the last three diagonal elements of P_{att} (representing the uncertainty of $\hat{g}_b$).

4) Attitude update: For the update step, it is necessary to define a set of measurements and the associated map h from the state space to the measurement space. The measured normalized quantities are described in eq. 17.

$$\begin{aligned} a_b &= [R](q)^T a_w \\ m_b &= [R](q)^T m_w \end{aligned} \quad (17)$$

Where a_b, m_b are respectively the gravity acceleration (decoupled from external acceleration as described in sec. III-B1) and magnetometer readings in the drone reference frame (or mobile frame); a_w, m_w are the corresponding vectors projected in the world (or global) reference frame. $[R](q)$ is the rotation matrix as described in eq. 2.

Considering the NED reference system, m_w can be decomposed as $[m_N, 0, m_D]^T$, and a_w is equal to $[0, 0, 1]^T$. This description is obtained by considering that the North is aligned with the magnetic vector, and the m_D component is due to the magnetometer *inclination*, which is a deviation of the magnetic vector towards the earth's centre. a_w measures only the gravity acceleration, which after normalization is equal to one. The components m_N, m_D can be computed as described by eq. 18.

$$\begin{aligned} a_b^T m_b &= a_w^T R R^T m_w = [0, 0, 1] m_w = m_D \\ m_N &= \sqrt{1 - m_D^2} \end{aligned} \quad (18)$$

Assuming a_b and m_b are not parallel (except for boundary cases, e.g. the earth poles), the columns of the matrix $B = [a_b | m_b | a_b \times m_b]$ are all linearly independent, thus B represent a basis for the vectorial space $\Re^3$. Being $[R](q)$ an isometry, it does not change the vectors' norm, nor their cross product. Therefore, eq. 19 holds.

$$\begin{aligned} B =& [a_b | m_b | a_b \times m_b] = [R]^T(q)[a_w | m_w | a_w \times m_w] = \\ =& [R]^T(q) W \end{aligned} \quad (19)$$

By using equation 19, it is possible to obtain $[R](q)$ as a function of the measurements. Considering that $det(WW^T) = m_N^2$, eq. 20 can be used to determine [R](q).

$$\begin{aligned} & BW^T = [R]^T(q)(WW^T) \\ & \Rightarrow [R]^T(q) = BW^T(WW^T)^{-1} = \\ & = \begin{bmatrix} \frac{m_x - m D a_x}{m_N} & \frac{a_y m_z - a_z m_y}{m_N} & a_x \\ \frac{m_y - m D a_y}{m_N} & \frac{a_z m_x - a_x m_z}{m_N} & a_x \\ \frac{m_z - m D a_z}{m_N} & \frac{a_x m_y - a_y m_x}{m_N} & a_x \end{bmatrix} \end{aligned} \quad (20)$$

By comparing eq.2 and 20, a map from the state space to the measurement space can be defined for at most nine different values. However, given we have only four state variables to determine, it is convenient to choose only four elements, otherwise the measurement vector will present dependent components within it, generating instability when inverting S_{k+1} (eq 12). The chosen values for the proposed solution are shown in eq. 21, along with the corresponding state-space map.

$$v_{meas} = \begin{bmatrix} a_x \\ a_y \\ a_z \\ \frac{a_y m_z - a_z m_y}{m_N} \end{bmatrix} = \begin{bmatrix} 2(q_1 q_3 - q_2 q_0) \\ 2(q_2 q_3 + q_1 q_0) \\ 1 - 2(q_1^2 + q_2^2) \\ 2(q_2 q_1 + q_3 q_0) \end{bmatrix} \quad (21)$$

Given the map from the state space to the measurement space, the update step can be performed as described in eq. 12 and 13.

5) Position prediction: The state for the second EKF comprehends (x,y) position and velocity, and is denoted as X_{pos}. Its uncertainty is marked as P_{pos}. The z component is ignored because the UWB infrastructure used to develop this system is projected for 2D positioning (being the anchor arrangement defective on the vertical axis). Given the external accelerations to which the drone is subject (see sec. III-B1), the position prediction is performed by integration. It results in a linear

system, whose state matrix and input matrix are described in eq. 22, and the input is the external acceleration.

$$A = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$B = \begin{bmatrix} \frac{dt^2}{2} & 0 & 0 \\ 0 & \frac{dt^2}{2} & 0 \\ dt & 0 & 0 \\ 0 & dt & 0 \end{bmatrix} \quad (22)$$

Where dt is the integration step. Being the system linear, P_{pos} can be computed as described in eq. 6. The input uncertainty can be computed by traditional uncertainty propagation when computing the external accelerations by using the accelerometer readings and the current attitude.

6) Position update: The UWB tag mounted on the quadcopter is considered a sensor that returns the current system position with a known level of uncertainty. In order to sense the quadcopter speed, every two different readings are used to compute the velocity by discrete first-order differentiation. This means that the position can be updated every time receiving new data from the sensor, but the speed only for every pair of readings.

By using this approach, the map from the state space to the measurement space is linear, represented by an identity matrix. However, during implementation, one must properly manage the reading pairs, their time interval, and the different update step when using only the position or the speed as well.

IV. Implementation and development steps

A. MATLAB simulation

Several filters have been implemented and tested via simulation on MATLAB. The development mainly focused on the EKF to estimate the attitude and the external accelerations, because the position estimation is trivial. It has been carried out incrementally, first under the hypothesis of no external accelerations and gyroscope bias, then considering the gyro drift, and finally adding external accelerations.

1) Gyroscope bias estimation: For the gyroscope bias estimation, the method described in sec. III-B2 has been implemented. As can be observed in fig. 1, the filter is perfectly able to compensate the drift, with a maximum error in the bias estimation lower than 0.05 [rad/s].

To assess the performances of the EKF, I implemented a Madgwick filter as a comparison. In fig. 2a is shown the estimation error of the EKF. The Roll and Pitch angles error are lower than $2°$, but the Yaw angle error riches up to $10°$. This is due to the simulation data, where the magnetometer has been set with higher noise than the accelerometer. The error spikes are caused by the limited range of $\pm 180°$ of the Euler angles. In fig. 2b is shown the estimation error of the Madgwick filter. The results are similar to the EKF, with a slightly lower error on the Yaw angle.

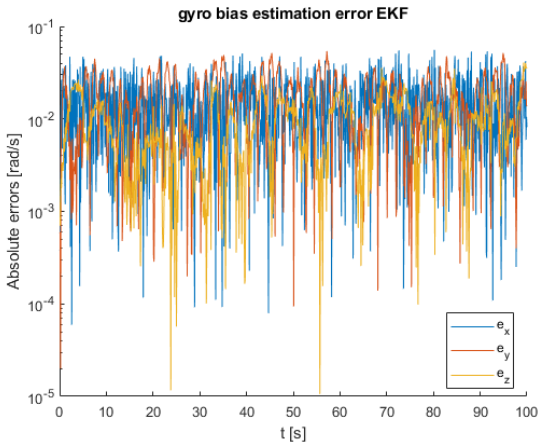
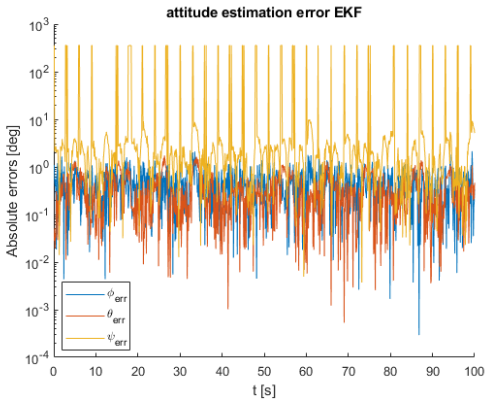
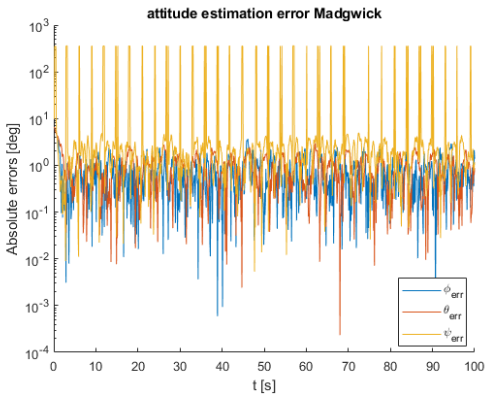


Fig. 1: Bias estimation error of the EKF, log scale.



(a)



(b)

Fig. 2: Attitude estimation error of (a) the EKF and (b) the Madgwick filter, log scale.

However, for the Madgwick filter to be able to properly compensate for the gyroscope bias variations, accurate information about its model are necessary. The Madgwick filter considers the gyroscope bias as a time-linear function with a given increasing factor γ. If the filter is provided with a wrong guess of γ, it is not able to compensate for the gyroscope drift.

In fig. 3 is shown the estimation error of the Madgwick filter when provided with opposite values of γ. Thus, if the real model is too different from the one provided to the algorithm, the estimation is completely incorrect.

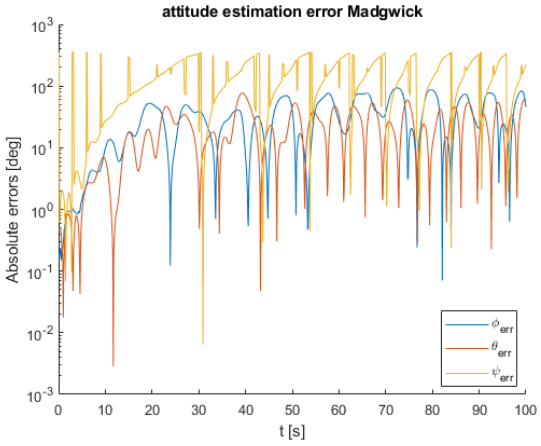


Fig. 3: Attitude estimation error of the Madwick filter provided with wrong parameter γ, log scale.

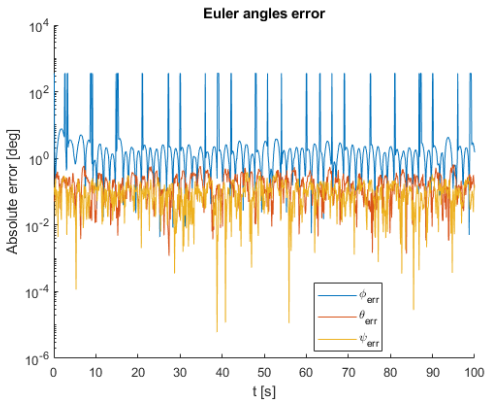
Given that the gyroscope bias model is quite unpredictable, the EKF present a better solution for this problem.

2) Acceleration decoupling: Both approaches presented in sec. III-B1 have been implemented and simulated.

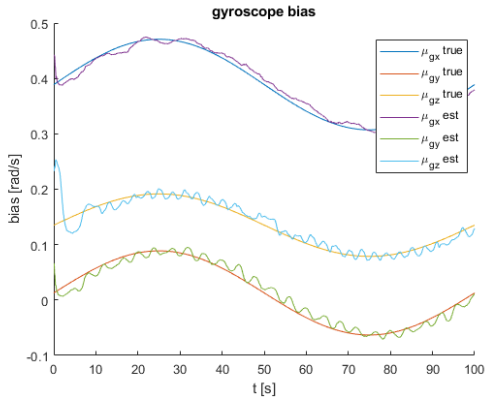
By exploiting the drone dynamics, under the hypothesis of negligible external forces, the algorithm can perfectly decouple the accelerations. In fig. 4a, 4b and 4c are reported the results achieved using this algorithm. As can be observed, the attitude error is equivalent to fig. 2a, the filter is capable of following a sinusoidal varying gyroscope bias, and the maximum error in the position is 10 [cm], probably caused by the error in the Yaw angle estimation.

In [5] the researchers have implemented a classifier to detect the external accelerations, which takes as input a buffer of several elaborated accelerometer samples. Moreover, they use a separate filter to estimate the gyroscope bias. In my implementation, I used a basic acceleration norm threshold detection, and for the gyroscope bias estimation I used the approach presented in sec. III-B2. In the simulation, the external accelerations applied are reported in fig. 5a (X and Y components are superimposed). In fig. 5b are displayed the external acceleration estimated by the filter. It can be observed that the estimation is coarse, even if the detection is precise. The filter is not able to decouple the accelerations contributions, estimating external accelerations also on the Z direction (even if absent).

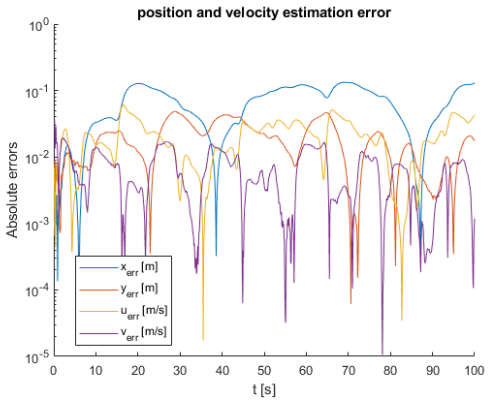
This causes noise in all other estimated quantities. Nevertheless, the filter is able to mitigate them. The gyroscope bias estimation is overall correct, as presented in fig. 6b, and the position error (fig. 6c) has a maximum of around 10 [cm], which may be acceptable depending on the application; the velocity error is a bit higher (over 1 [m/s]). The estimation error



(a) EKF attitude error on the Euler angles, log scale.



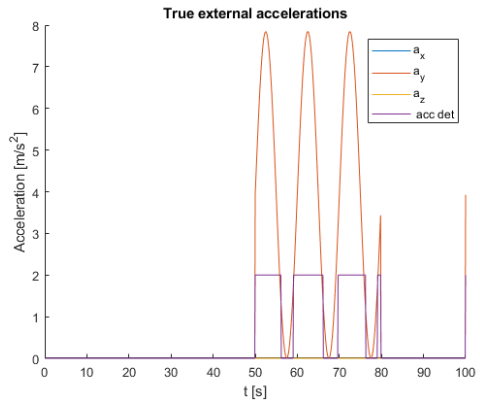
(b) True and estimated gyroscope bias.



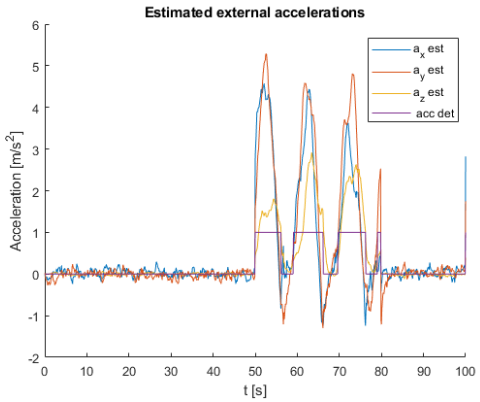
(c) Position and velocity estimation error, log scale.

Fig. 4: Computation results when exploiting the drone dynamics to decouple the acceleration contributions.

reported in fig. 6a, concurrently with the external acceleration, is up to 20 [deg], but right after is immediately corrected. Thus, the error does not last after the external acceleration ends, thanks to the filter always estimating a correct gyroscope bias.

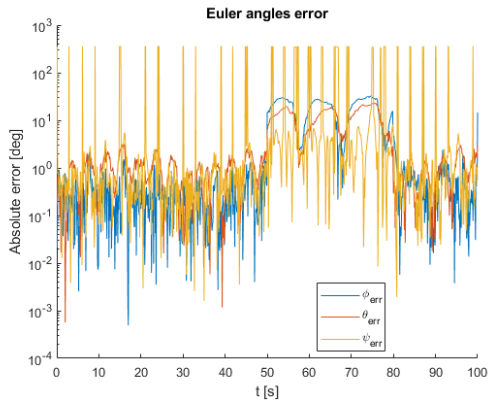


(a) Simulated external accelerations, X and Y components are superimposed.

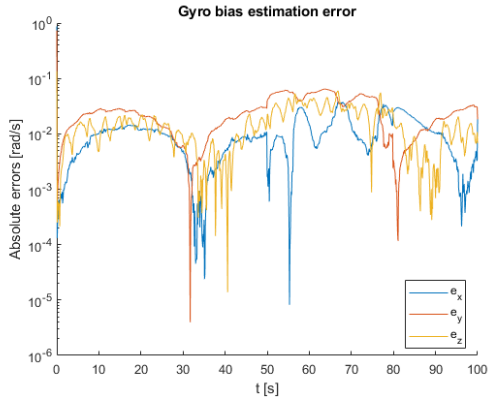


(b) Estimated external accelerations.

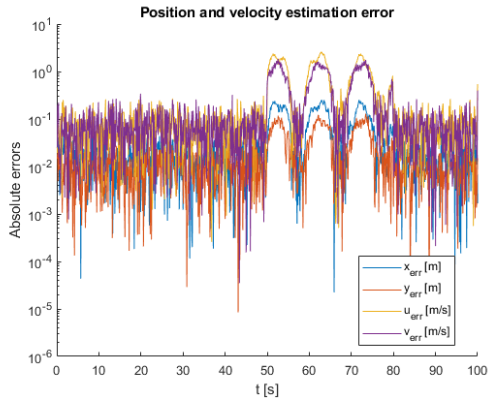
Fig. 5: Computation results when exploiting the drone dynamics to decouple the acceleration contributions.



(a) EKF attitude error on the Euler angles, log scale.



(b) Estimated gyroscope bias error.



(c) Position and velocity estimation error, log scale.

Fig. 6: Computation results implementing the external acceleration compensation method presented in [5].

B. UWB tag

1) DTDoA implementation: For this project, I used the UWB module DWM1001 from Decawave for the UWB tag, the same device used in [3]. The first step in the development of the proposed solution has been the programming of the UWB tag and implement the DTDoA algorithm on board. This required also the implementation of some basic linear algebra computation, such as the pseudo-inverse of a non-square matrix. I decided to manipulate all matrices as flat arrays to allow easier function implementation. In fig. 7 are reported the histograms of the data collected from the tag. The true values have been collected using a motion capture system. The error on the mean values in the X and Y directions are 0.0404 [m] and 0.0141 [m], respectively. Thus, we can conclude that, on average, the DTDoA algorithm provides a reliable measure of the tag position.

2) On-board sensors: After that, the idea was to implement the whole solution on the DWM1001, hence programming the EKF and running all the computations on board. This required first the programming of the sensor. I used the 3-axis accelerometer LIS2DH12 and the 3-axis magnetometer LIS2MDL from STMicroelectronics, but no gyroscope sensor was yet available. In order to calibrate the sensors and compute their uncertainties, heavy computations are needed. Thus, I preferred to run them on a local machine using the programming language Python. I created a library for the UWB tag to manage the communications. Pieces of data were managed using packets, with the Python script acting as master, managing the communication and controlling the flow of the packages.

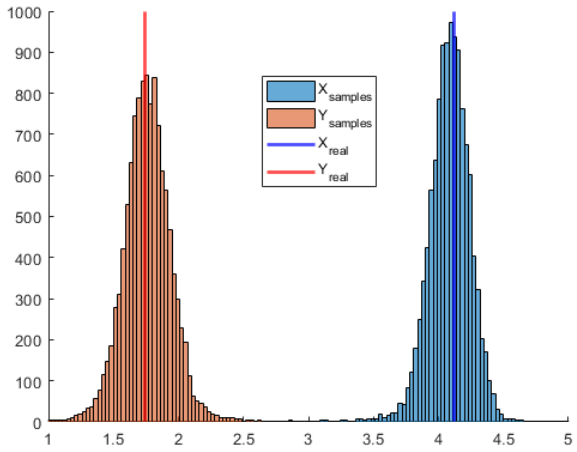


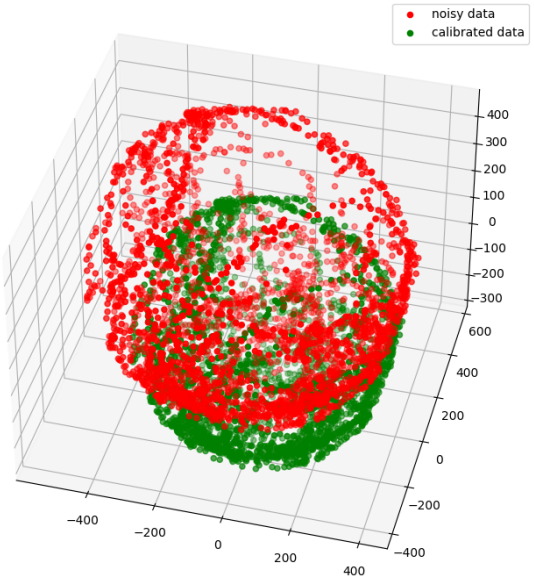
Fig. 7: Histogram of the DTDoA results running on board on the UWB tag.

To calibrate the magnetometer, I tried two different ellipsoid-fitting algorithms. The first algorithm is from Nicolas Liaudat (Git repository), which, given the magnetic vector norm, fits the ellipsoid using nine parameters (offset, axes distortion and 3D rotation). The second is from the function *magcal* from MathWorks MATLAB, which implements a three-step ellipsoid-fitting algorithm, first with four parameters (magnetic vector norm and offset), second with seven parameters (magnetic vector norm, offset and axes distortion) and last with ten parameters (magnetic vector norm, offset, axes distortion and 3D rotation), checking at each step if the new fitted model is more accurate than the previous one. By python script, it is possible to start and interrupt the data sampling, during which the sensor must be rotated to map a sphere of samples sufficiently dense. In fig. 8a and 8b the results using the two different algorithms are presented. With the dataset being dense enough, the performances are equal, but probably the algorithm by MathWorks MATLAB is more flexible and adaptive in the case of more sparse or defective samples.

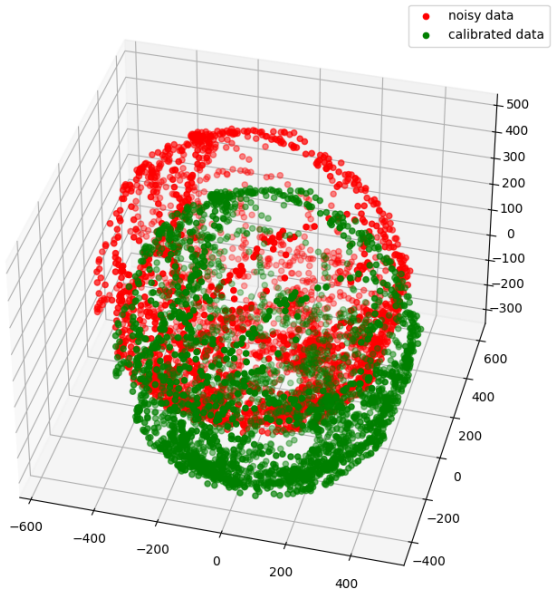
Regarding the accelerometer calibration, it consists in computing a scale and offset for each axis of the sensor in order to map the readings in a desired range. Thus, it is necessary to register the positive and negative gravity acceleration read by the sensor on each axis by manually rotating it. For this purpose, a median filter has been implemented to suppress external acceleration spikes generated by manual rotation.

Moreover, it is necessary to locally store the calibration settings to avoid recalibration every time the tag is rebooted. Thus, I implemented a simple library to write and read the calibration data from the flash memory of the DWM1001.

3) On-board EKF: The next step was implementing the attitude EKF. Given that no gyroscope was available, a zero-bias random walk model has been used. However, even under conditions of no external acceleration, the filter was performing poorly. It was not able to properly estimate the Yaw angle



(a) Algorithm by Nicolas Liaudat, residuals = 0.02146



(b) Algorithm by MathWorks MATLAB, residuals = 0.02146

Fig. 8: Scatter plot of the magnetometer data before and after calibration.

of the body, with the angle having a range half of the full plane (from 0° to 180°). It has been discovered that using only the four measures indicated in 21, in particular the fourth, is not sufficient to map the whole plane for the Yaw angle. Thus, without a gyroscope sensor enforcing the switch during the prediction phase between the two halves of the plane, the update phase was bounding the estimation only in one of them. Hence, it was necessary to use one more measure. In eq. 23 is

displayed the rotation matrix as a function of the Euler angles.

$$\begin{bmatrix} C\psi C\theta & C\psi S\theta S\phi - S\psi C\phi & C\psi S\theta C\phi + S\psi S\phi \\ S\psi C\theta & S\psi S\theta S\phi + C\psi C\phi & S\psi S\theta C\phi - C\psi S\phi \\ -S\theta & C\theta S\phi & C\theta C\phi \end{bmatrix} \tag{23}$$

$$\frac{S\psi C\theta}{C\psi C\theta} = \tan\psi \tag{24}$$

Where C and S denote the cosine and sine functions, respectively. One can observe that the ratio between the elements in positions (2,1) and (1,1) is equal to the tangent of the Yaw angle (as reported in eq. 24). Thus, the idea is to include as the fifth measure the element (2,1), whose map from state space is $2(q1q2 + q3q0)$ (eq. 2), in order to solve the angle indetermination. However, this brings another issue. As stated in sec. III-B4, the equations system to compute the Kalman gain (eq. 8) is overdetermined, leading to instability when inverting the matrix S_{k+1}. To solve this problem, I searched how Mathworks MATLAB implements the right division and implemented the same algorithm. Instead of inverting S_{k+1}, the Kalman gain is computed by solving a linear system where W_{k+1}^T is the unknowns vector, S_{k+1}^T the coefficients matrix and $(P_{k+1}^- H_{k+1}^T)^T$ the known terms. The solution of the linear systems is performed by using the *LU decomposition* on W_{k+1}^T and then solving two systems using forward and backward propagation. These steps are reported in eq. 25 and 26 ($k+1$ has been omitted for simpler notation).

$$\begin{aligned} &W = P^- H^T / S \Rightarrow WS = P^- H^T = Ph \\ &WS = Ph \Rightarrow S^T W^T = Ph^T \\ &S^T = LU \Rightarrow LUW^T = Ph^T \end{aligned} \tag{25}$$

$$\begin{aligned} &\text{solve } LX = Ph^T \text{ forward prop.} \Rightarrow X \\ &\text{solve } UW^T = X \text{ backward prop.} \Rightarrow W^T \end{aligned} \tag{26}$$

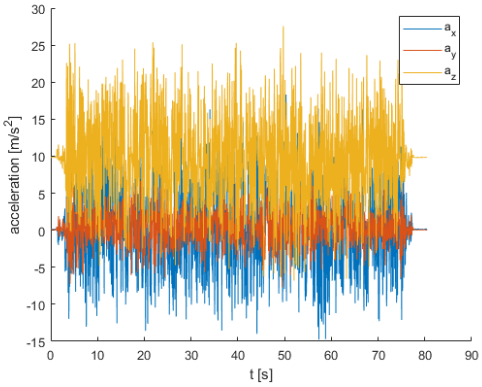
C. Full sensors setup

To achieve better results, especially when trying to compensate for external acceleration, it is preferable to use a gyroscope sensor in order to use a more precise model in the prediction phase. Hence, I started to work with the 9-axis ICM-20948 IMU, using the tag only as a position sensor. At this point, the project has been developed in Python, using a Raspberry zero as control board. The purpose of the code was only to collect data, which have been processed offline later. The data collection and the processing results are presented in sec. V.

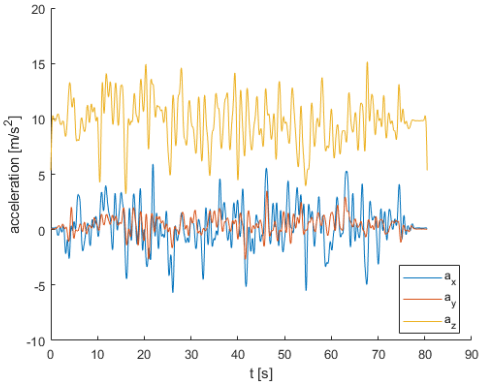
V. Experimental results

For the data collection, the control board, the UWB tag and the IMU have been mounted on a drone. Then, the drone has been moved on a rectangular trajectory. The true position and attitude have been registered using a motion capture system. The main problem during this data collection was the extremely noisy accelerometer data, as displayed in fig. 9a.

This problem is due to the IMU being mounted directly on the quadcopter chassis. This, as studied in [7], causes the IMU to sense the vibrations generated by the propellers. Researchers in [7] states that algorithms can't compensate



(a) Accelerometer data.



(b) Accelerometer data after low-pass filtering.

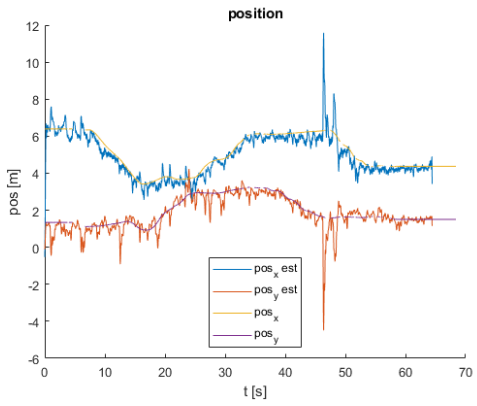
Fig. 9: Collected and elaborated accelerometer data from test 2.

for this large noise, but it is necessary to adopt mechanical solution installing dumpers to reduce vibrations. Moreover, it must be considered that the IMU wasn't installed on the quadcopter centre of mass, generating undesired contributions caused by rotations. However, the filtered accelerometer data, displayed in fig. 9b, show that maybe it is possible to exploit the quadcopter dynamics to decouple the accelerations, which would allow precise estimation. I tried to run that algorithm on data filtered in post-processing with a low pass filter, but it didn't lead to meaningful results.

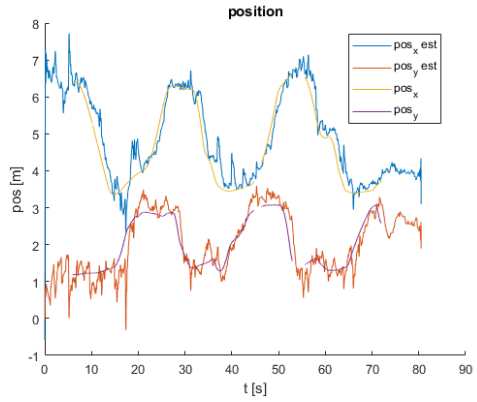
In fig. 10a and 10b are displayed the true and estimated position during the two tests. By tuning the uncertainty on the sensor, the filter is able to follow the trajectory with a coarse estimation. Due to the large noise, the algorithm is not able to estimate the attitude, nor the gyroscope bias.

VI. Conclusions

The UWB technology has proved to be an optimal candidate for indoor positioning systems. The EKF is the current state of the art for estimation, showing promising results in simulation phase. It is capable of correctly estimating the gyro bias in several test cases. It is challenging to decouple the acceleration contribution by using model-free algorithms. However, it



(a)



(b)

Fig. 10: True and estimated position during two different tests.

would allow the development of a general-purpose estimation system, both for position and attitude, that could work on any platform. From the collected data, it is not clear if it is possible to exploit the quadcopter dynamic to decouple the acceleration, thus this hypothesis needs further analysis.

During this project, I developed a full system to program, collect data and calibrate sensors for the DWM1001 SoC. The communication and flash management libraries are easily scalable, allowing for the definition of new packages, data streams and memorized variables. Once the gyroscope will be available, it should be trivial to add it to the current project. The EKF library is already implemented and needs only a few adjustments to integrate the gyroscope sensor readings. All code and data are available on this GitHub repository.

VII. Acknowledgements

I would like to thank Luca Santoro and Matteo Nardello for all the support they gave me during this project development and the patience they showed. The topics I studied during this experience made me develop a keen interest in estimation and embedded systems, which I hope to be able to learn even more about. Generally, the course on Designing methods for unmanned vehicles fascinated me. Indeed, I will continue my studies in this direction by developing a thesis about the use of behaviour trees for autonomous execution of hierarchically structured mission tasks of UAVs, at DLR German Aerospace Center.

References

[1] D. Mariusz, S. S. Mieczysław, V. George-Ch., and M. Gerasimos, "Opportunities and challenges for exploiting drones in agile manufacturing systems," *Procedia Manufacturing*, vol. 51, pp. 527–534, 2020, 30th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2021). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2351978920319314

[2] O. Maghazei and T. Netland, "Drones in manufacturing: exploring opportunities for research and practice," *Journal of Manufacturing Technology Management*, vol. 31, no. 6, pp. 1237–1259, Jan 2020. [Online]. Available: https://doi.org/10.1108/JMTM-03-2019-0099

[3] L. Santoro, M. Nardello, D. Brunelli, and D. Fontanelli, "Scale up to infinity: the UWB indoor global positioning system," in *2021 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*. IEEE, oct 2021. [Online]. Available: https://doi.org/10.1109%2Frose52750.2021.9611770

[4] P. Dabove, V. Di Pietra, M. Piras, A. A. Jabbar, and S. A. Kazim, "Indoor positioning using ultra-wide band (uwb) technologies: Positioning accuracies and sensors' performances," in *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2018, pp. 175–184.

[5] M. A. Javed, M. Tahir, and K. Ali, "Cascaded kalman filtering-based attitude and gyro bias estimation with efficient compensation of external accelerations," *IEEE Access*, vol. 8, pp. 50 022–50 035, 2020.

[6] J. Kaiser, A. Martinelli, F. Fontana, and D. Scaramuzza, "Simultaneous state initialization and gyroscope bias calibration in visual inertial aided navigation," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 18–25, 2017.

[7] A. Johansson and D. Wallén, "Quadcopter sensor and filter evaluation," 2016. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:955490/FULLTEXT01.pdf