Università di Pisa
Scuola di Ingegneria
Master's Degree in
Robotics and Automation Engineering

Academic Year
2020/2021

GUIDANCE AND NAVIGATION SYSTEMS

PROJECT WORK:

SENSOR FUSION OF AHRS, GPS, UWB, COMPUTER VISION, BAROMETER AND ALTIMETER FOR AN OUTDOOR-INDOOR UAV

Student: Davide De Benedittis

Professor: Lorenzo Pollini

Table of Contents

Case Study

Introduction

Vehicle

Low-speed quadrocopter for packages transportation

Environment

- Confined and partially unstructured environment
- Both indoor and outdoor

Fused sensors

1. MARG
2. GPS
3. UWB
4. Computer vision (with ArUco Markers)
5. Barometer
6. Laser altimeter

Problems and Difficulties

Integration of multiple sensors with:

- Different frames:
  1. rotated sensor reference frames
  2. sensor offset relative to body frame
  3. altimeter measurement dependent on ground height

- Unknown frames:
  - UWB beacons can have unknown positions
  - ArUco markers can have unknown positions

Problems and Difficulties

The sensors are affected by:

1. Measurement outliers (NLOS errors of GPS and UWB, computer vision errors)
2. Outages of unknown duration (GPS, UWB, vision)
3. Variable sensor noises
4. Environmental disturbances (magnetometer)
5. Variable measurement rates

Filter Design Choices

Error-State (Indirect) Quaternion-Based Adaptively-Robust Extended Sequential Kalman Filter (I-QB-AR-ESKF)

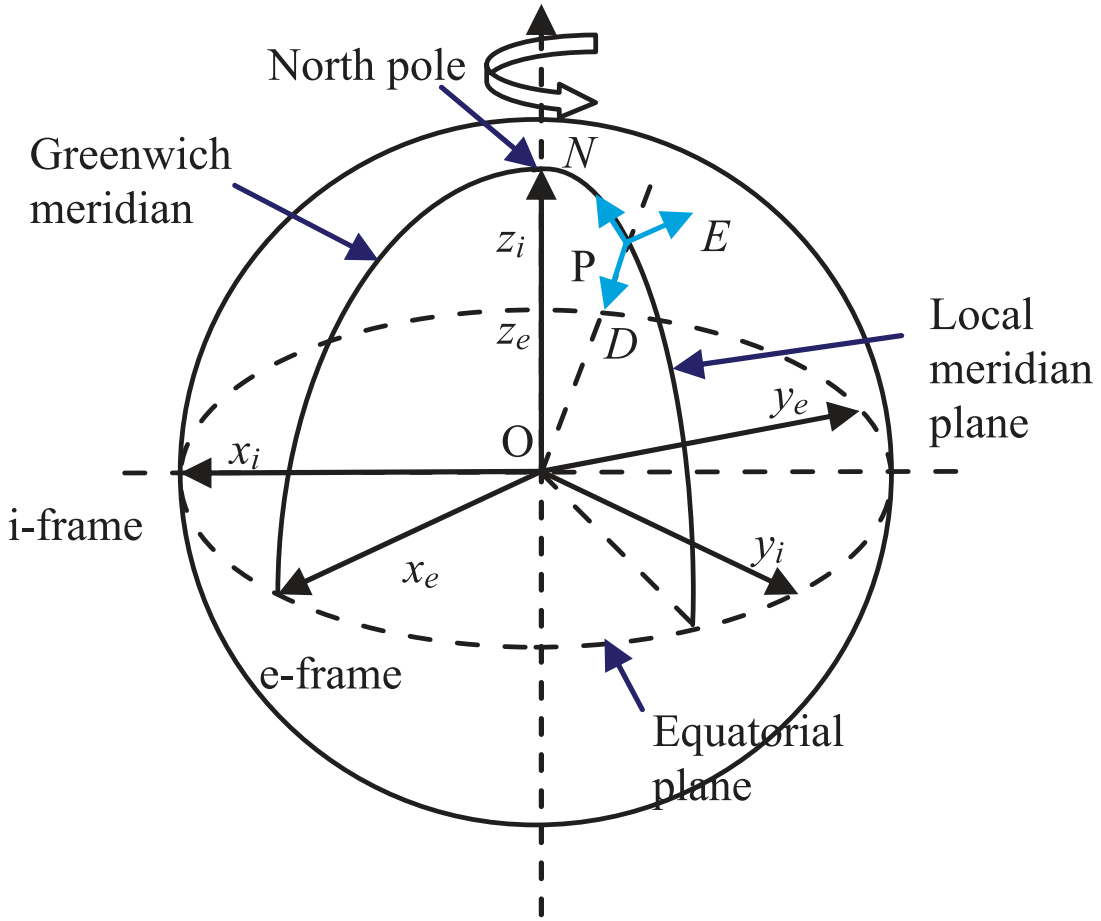
- **ES** avoids the numerical problems and the divergence of the Direct KF
- **QB** guarantees robustness to the (extremely unlikely) singularities of the Euler Angles and a lower computational cost
- **AR** increased robustness to outliers, disturbances and sensor errors in addition to adaptiveness
- **EKF** industry standard, enormous availability of literature, appropriate trade-off between accuracy, computational cost and simplicity
- **S** suitable for estimation with sensors with different measurement rates, reduced computational cost

Mathematical Background

Reference Frames

Used reference frames (RF):

1) Navigation Frame (n)
2) Body Frame (b)
3) i-th Sensor Frame (s_i)
4) Camera Frame (c)



Reference Frames

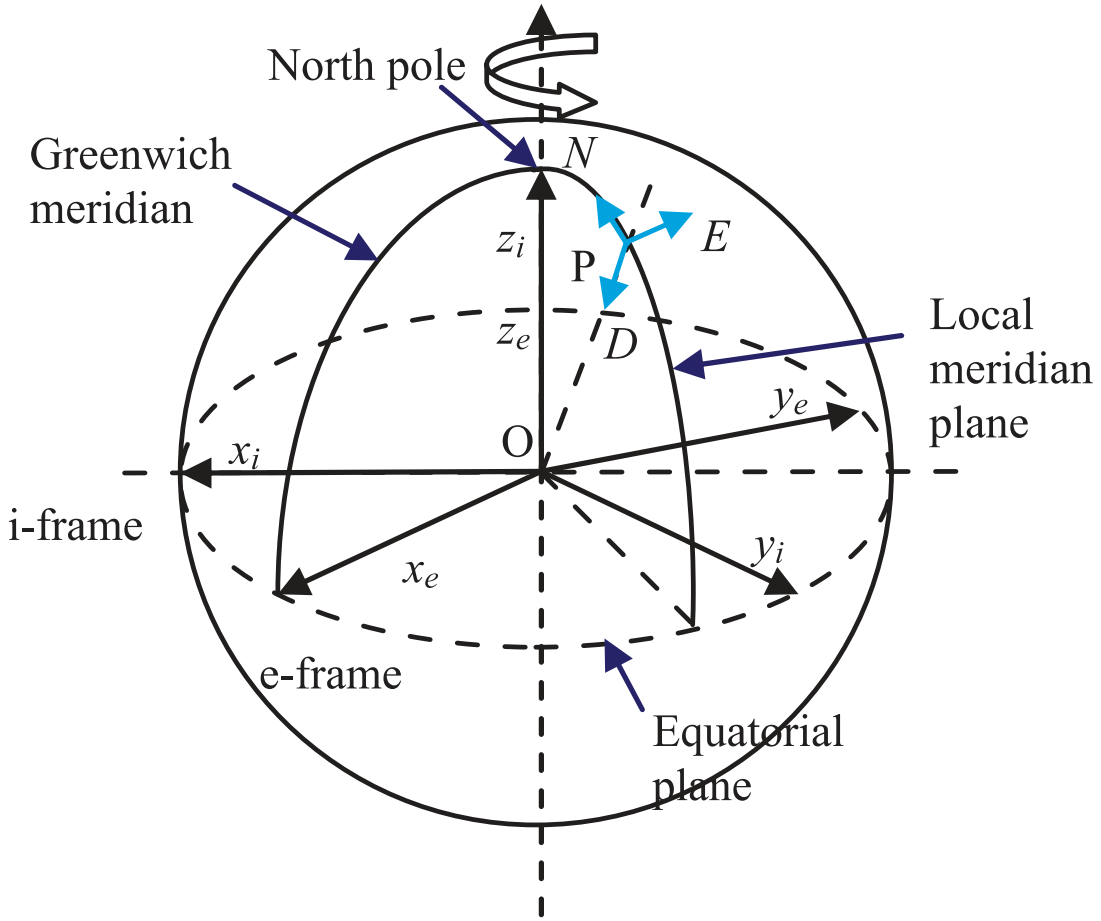
Used navigation frame

North-East-Down (NED) Local Geographic Frame

Output coordinate system

More efficient and meaningful to provide the output coordinates directly in NED frame, not in the geodetic frame.

(The UAV needs to operate only in a fixed and confined environment)



Quaternions and Rotations

Definition:

$$\boldsymbol{q} = q_0 + q_1\boldsymbol{i} + q_2\,\boldsymbol{j} + q_3\boldsymbol{k} = \begin{bmatrix} q_0 \\ \boldsymbol{q}_v \end{bmatrix}$$

Multiplication:

$$\boldsymbol{q} \otimes \boldsymbol{p} = \begin{bmatrix} q_0 p_0 - \boldsymbol{q}_v{}^T \boldsymbol{p}_v \\ q_0\boldsymbol{p}_v + p_0\boldsymbol{q}_v - \boldsymbol{q}_v \times \boldsymbol{p}_v \end{bmatrix}$$

Rotation:

$$\begin{bmatrix} 0 \\ \boldsymbol{p}^n \end{bmatrix} = \boldsymbol{q}_{nb} \begin{bmatrix} 0 \\ \boldsymbol{p}^b \end{bmatrix} \boldsymbol{q}_{nb}^*$$

Derivative:

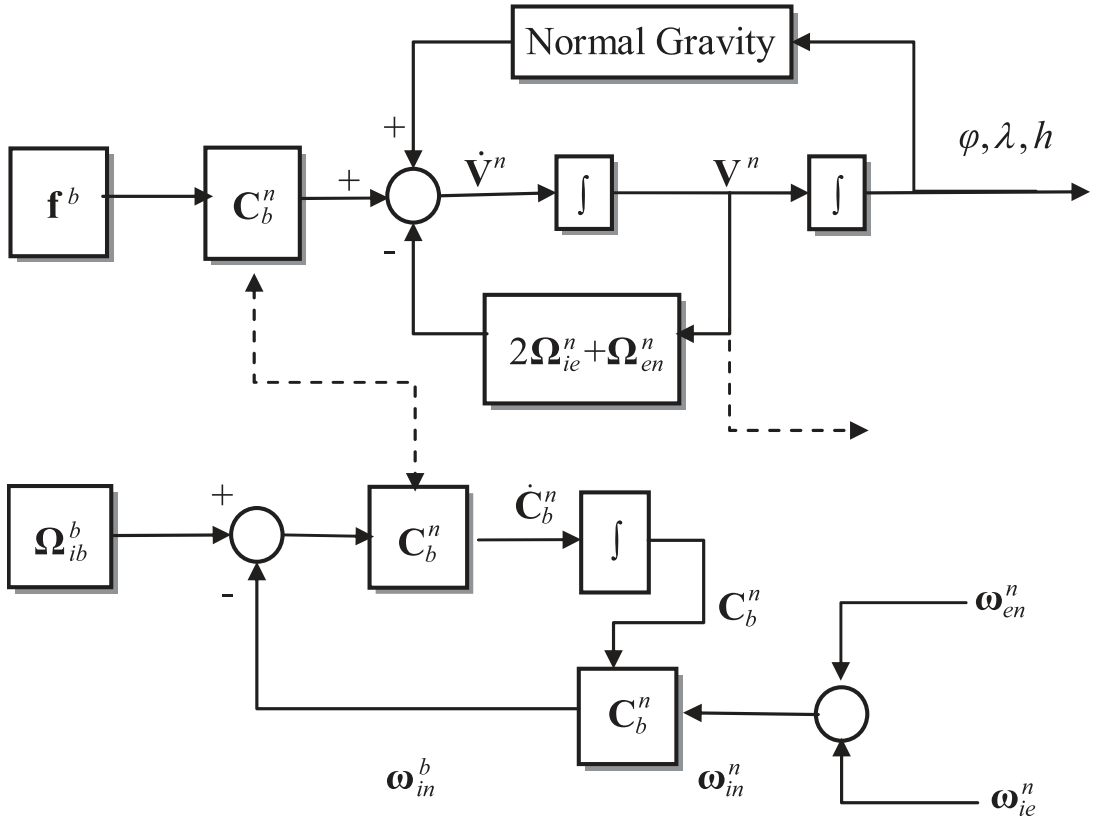
$$\frac{\mathrm{d}\boldsymbol{q}}{dt} = \frac{1}{2}\delta t \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix}$$

First order DT quat. integrator :

$$\boldsymbol{q}(k) = \boldsymbol{q}(k-1) \cdot \left[\exp\left(\frac{1}{2} \begin{bmatrix} 0 \\ \bar{\boldsymbol{\omega}} \end{bmatrix} (\Delta t) \right) + \frac{\Delta \mathrm{t}^2}{24} \begin{bmatrix} 0 \\ \boldsymbol{\omega}_{k-1} \times \boldsymbol{\omega}_k \end{bmatrix} \right]$$

with $\bar{\boldsymbol{\omega}} = \frac{1}{2}(\boldsymbol{\omega}_k + \boldsymbol{\omega}_{k-1})$

StrapDown Platform



- Low-cost IMU is mounted solidal to the UAV body: StrapDown Platform
- Simplifications:
  - Neglected Earth rotation (n is inertial)
  - Earth is considered as flat (Cartesian CS)
- The equations used are the following:

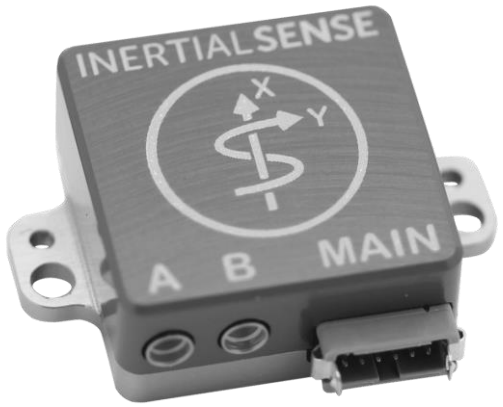
$$\boldsymbol{\omega}_{bn}^n = -C_b^n \boldsymbol{\omega}_{ib}^b$$

$$\dot{\boldsymbol{v}}^n = C_b^n \boldsymbol{f}^b + \boldsymbol{g}^n$$

$$\dot{C}_b^n = -[\boldsymbol{\omega}_{nb}^n \times] C_b^n$$

Sensor Models

IMU



Inertial Measurement Unit

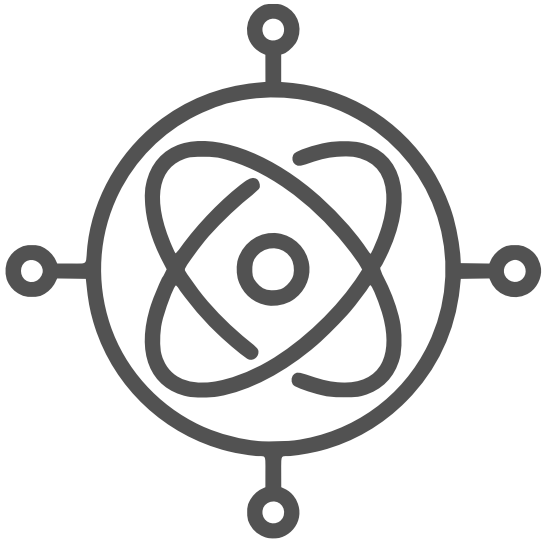
Acceleromer + Gyroscope + Magnetometer

Simulated using the MATLAB function `imuSensor` present in the Sensor Fusion toolbox.

The considered errors are white noise, bias and quantization.

It is supposed that other sources of errors (such as axis misalignments and magnetometers soft-iron distortions) have been compensated with a previously performed sensor calibration.

Gyroscope



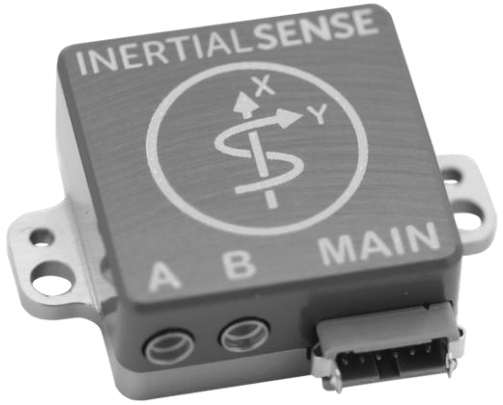
$$\boldsymbol{y}_{G,k} = \boldsymbol{\omega}_k + \boldsymbol{\omega}_{b,k} + \boldsymbol{v}_{G,k}$$

$$\boldsymbol{\omega}_{b,k} = \boldsymbol{\omega}_{b,k-1} + \boldsymbol{w}_{\omega_b,k} \qquad \text{(First-order RW)}$$

With:

1) $\boldsymbol{y}_{G,k}$ gyroscope sensor measurement
2) $\boldsymbol{\omega}_k$ true angular velocity
3) $\boldsymbol{\omega}_{b,k}$ zero-rate gyro offset
4) $\boldsymbol{v}_{G,k}$ gyro zero-mean (ZM) additive Gaussian white noise (AGWN)
5) $\boldsymbol{w}_{\omega_b,k}$ gyro bias zero-mean additive Gaussian white noise

Accelero-meter



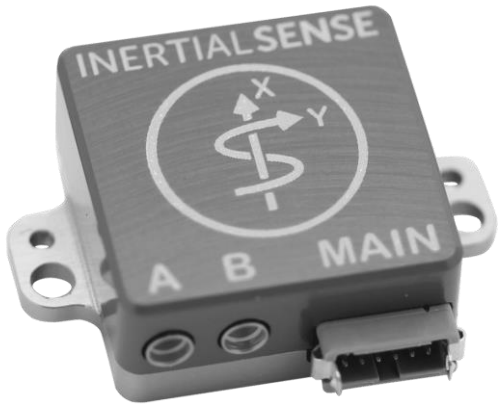
$$\boldsymbol{y}_{A,k} = -\boldsymbol{a}_k^{s_a} + \boldsymbol{g}_k^{s_a} + \boldsymbol{a}_{b,k}^{s_a} + \boldsymbol{v}_{A,k}$$

$$\boldsymbol{a}_{b,k}^{s_a} = \boldsymbol{a}_{b,k-1}^{s_a} + \boldsymbol{w}_{b_a,k} \qquad \text{(First-order RW)}$$

With:

1) $\boldsymbol{y}_{A,k}$ accelerometer sensor measurement
2) $\boldsymbol{a}_k^{s_a}$ true linear acceleration in acceleromer sensor RF
3) $\boldsymbol{g}_k^{s_a}$ gravity vector
4) $\boldsymbol{a}_{b,k}^{s_a}$ accelerometer bias
5) $\boldsymbol{v}_{A,k}$ accelerometer zero-mean AGWN
6) $\boldsymbol{w}_{a,k}$ linear acceleration zero-mean AGWN
7) $\boldsymbol{w}_{a_b,k}$ accelerometer bias zero-mean AGWN

Magneto-meter



$$\boldsymbol{y}_{M,k} = \boldsymbol{m}_k^{Sm} + \boldsymbol{d}_k^{Sm} + \boldsymbol{v}_{M,k}$$

$$\boldsymbol{d}_k^{Sm} = c_d\, \boldsymbol{d}_{k-1}^{Sm} + \boldsymbol{w}_{d,k} \qquad \text{(Low-pass filtered WN)}$$

With:

1) $\boldsymbol{y}_{M,k}$ magnetometer sensor measurement
2) $\boldsymbol{m}_k^{Sm}$ true Earth magnetic vector
3) $\boldsymbol{d}_k^{Sm}$ magnetic disturbance
4) $\boldsymbol{v}_{M,k}$ magnetometer zero-mean AGWN
5) $\boldsymbol{w}_{d,k}$ magnetometer bias zero-mean AGWN

GPS Position

Global Positioning System

Both the position and the velocity measurements are accessible by the navigation system.

Simulated using the MATLAB function `gpsSensor` present in the Sensor Fusion toolbox, considering white noises (different on lat-long and on altitude) and bias errors.

Loosely-coupled integration:

- PRO: easier implementation, lower computational cost
- CONS: worse accuracy compared to Tightly-coupled and GPS-RTS integration, problems with low satellites count

GPS Position

$$\boldsymbol{y}_{pGPS,k} = \boldsymbol{p}_k^g + \boldsymbol{b}_{GPS,k}^g + \boldsymbol{v}_{pGPS,k}$$

$$\boldsymbol{b}_{GPS,k}^g = \boldsymbol{b}_{GPS,k-1}^g + \boldsymbol{w}_{b_pGPS,k} \qquad \text{(First-order RW)}$$

With:

1) $\boldsymbol{y}_{pGPS,k}$ GPS position sensor measurement (as latitude, longitude and altitude)
2) $\boldsymbol{p}_k^g$ true position vector
3) $\boldsymbol{b}_{GPS,k}^g$ GPS position bias
4) $\boldsymbol{v}_{pGPS,k}$ GPS position zero-mean AGWN
5) $\boldsymbol{w}_{b_pGPS,k}$ GPS position bias bias zero-mean AGWN

GPS Velocity

$$\boldsymbol{y}_{vGPS,k} = \boldsymbol{v}_k^n + \boldsymbol{v}_{vGPS,k}$$

With:

1) $\boldsymbol{y}_{vGPS,k}$ GPS velocity sensor measurement
2) $\boldsymbol{v}_k^n$ true position vector
3) $\boldsymbol{v}_{vGPS,k}$ GPS zero-mean AGWN

UWB

Ultra WideBand

Simulated using the custom system object `uwb_sensor`.

It is possible to set the beacons positions and ids (as vectors), the beacons maximum and minimum range, the measurement noise and the RNG seed.

Function call:

```
[meas_dist, id] = uwb(drone_pos, unobstructed_ids)
```

Where the second input is used to take into account the fact that walls may obstruct some beacons signal.

No NLOS errors, only WGN and quantization errors are considered.

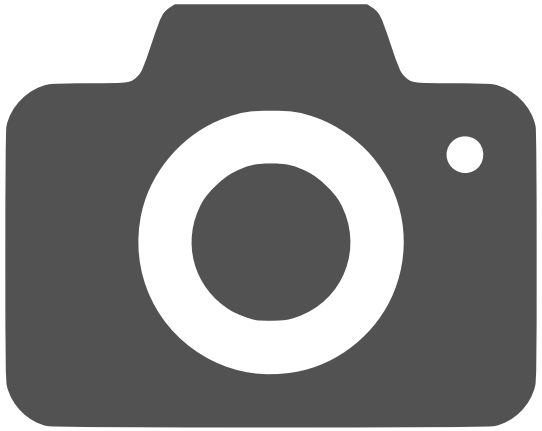
UWB

$$\boldsymbol{y}_{UWB,i,k} = \left\| \boldsymbol{p}_k^n - \boldsymbol{p}_{beacon,i}^n \right\|_2 + \boldsymbol{v}_{UWB,k} + \boldsymbol{v}_{UWB,NLOS,k}$$

With:

1) $\boldsymbol{y}_{UWB,i,k}$ UWB i-th distance sensor measurement
2) $\boldsymbol{p}_{beacon,i}^n$ true i-th anchor position
3) $\boldsymbol{v}_{UWB,k}$ UWB zero-mean AGWN
4) $\boldsymbol{v}_{UWB,NLOS,k}$ eventual UWB NLOS error (not ZM, nor AWGN) added “manually” in some simulations

Computer Vision



Computer Vision: ArUco markers

Simulated using the custom system object `aruco_vision_sensor`.

It is possible to set the marker positions and ids (vectors), the vision system maximum range, the measurement noise and the RNG seed.

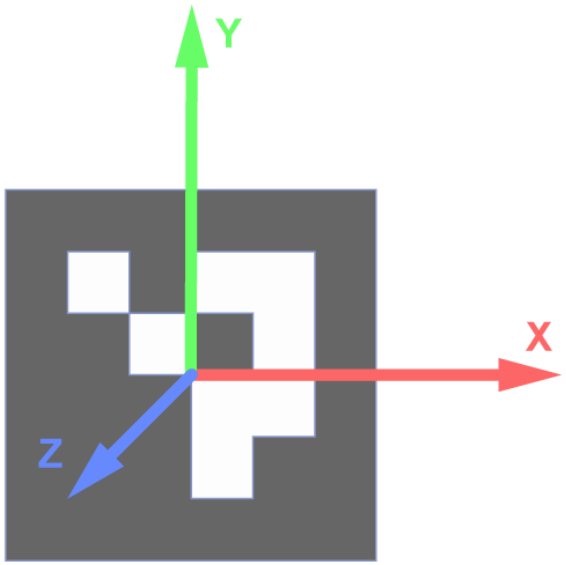
It is also possible to set the camera (optical center) position and orientation relative to the body frame.

Function call:

```
[Tcm, id] = aruco(drone_orientaion, drone_position)
```

The output is a homogeneous roto-traslation matrix.

Computer Vision



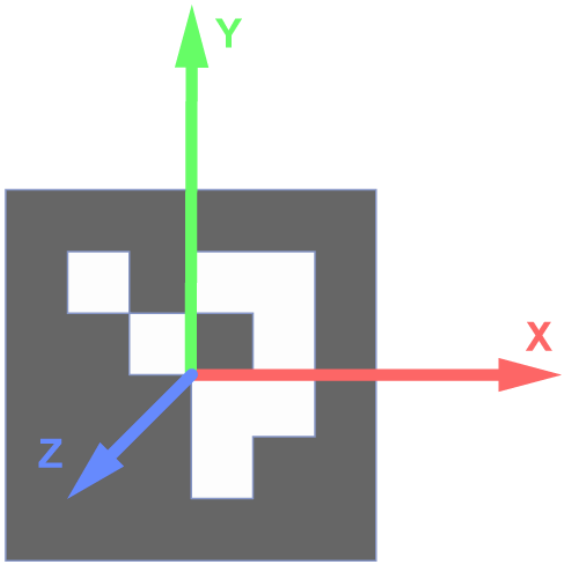
Errors

Some considerations have been done regarding the principles of the computer vision to model its errors.

It has been determined that the errors increases when the camera-marker distance increases; in addition they are not the same on all directions.

It is not considered the case where the marker plane is parallel to the line of sight, which would increase the errors. The z axis thus is similar to the line of sight direction.

Computer Vision

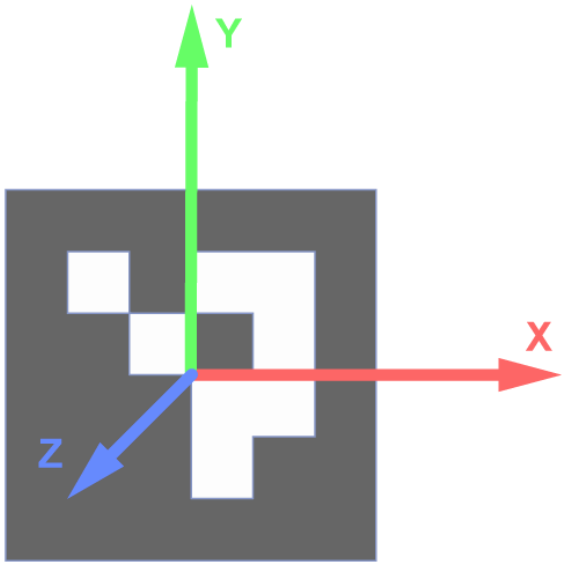


Errors

Orientation error: multiplicative error matrix with (small) Euler angles. These Euler angles have a statistical distribution which is the sum of a ZM-GWN and another ZM_GWN multiplied by the camera-marker distance. The errors on the yaw angle are smaller than the ones on the roll and pitch angles.

Position error: additive errors. The error is again distributed as the sum of a ZM-GWN and another ZM_GWN multiplied by the camera-marker distance. This time the error along the z-axis is greater than the one on the other axes (considering that the marker plane is roughly perpendicular to the LOS).

Computer Vision



Errors

Orientation error:

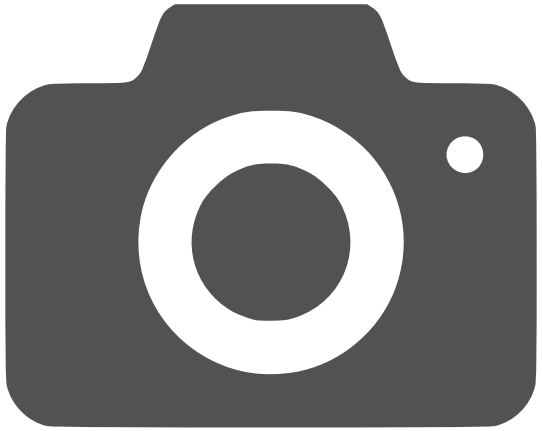
$$R_{\text{error}} = R_z(\varphi_{\text{err}})R_y(\theta_{\text{err}})R_x(\phi_{\text{err}})$$

$$R_{\text{measured}} = R_{\text{error}} \cdot R_{\text{true}}$$

Position error:

$$\boldsymbol{p}_{\text{measured}} = R_{\text{error}} \cdot \boldsymbol{p}_{\text{true}} + \boldsymbol{p}_{\text{error}}$$

Computer Vision



$$\boldsymbol{y}_{pVis,i,k} = -\boldsymbol{p}^{\mathrm{m_i}}_{m_i c,k} + \boldsymbol{v}_{pVis,k}$$

$$\boldsymbol{y}_{oVis,i,k} = V_{oVis,k} \cdot C_{m_i c}$$

With:

1) $\boldsymbol{y}_{pVis,i,k}$ vision position sensor measurement
2) $\boldsymbol{p}^{m_i}_{m_i c,k}$ true position of the m_i marker with respect to the camera, in its own RF
3) $\boldsymbol{v}_{pGPS,k}$ vision position zero-mean AGWN
4) $\boldsymbol{y}_{oVis,i,k}$ vision orientation sensor measurement
5) $C_{m_i c}$ true rotation matrix between the i-th marker frame and the camera frame
6) $V_{oVis,k}$ suitable error matrix (zero-mean AGWN on the euler angles)

Barometer

Simulated using the custom system object: `barometer_sensor`

It is supposed that the sensor itself performs the conversion between pressure and altitude, and the output provided is the altitude.

The displacement of the sensor w.r.t. the drone COM is neglected.

The sources of errors are AWGN, bias instability and quantization.

Altimeter

Simulated using the custom system object: `altimeter_sensor`

Both the position of the sensor on the drone and the drone's pose are considered.

In addition, the ground height is not supposed constant.

The sources of errors are AWGN, bias instability and quantization.

Barometer

$$y_{\text{bar},k} = h_k^{s_b} + v_{\text{bar},k}$$

With:

1) $y_{\text{bar},k}$ barometer sensor output
2) $h_k^{s_b}$ real height in s_b
3) $v_{\text{bar},k}$ ZM AWGN

Altimeter

$$y_{\text{alt},k} = \frac{h_k^n - h_{\text{sensor},k}^n - h_{g,k}^n}{\cos(\phi_{\text{vertical}})} + v_{\text{alt},k}$$

With:

1) $y_{\text{alt},k}$ altimeter sensor output
2) h_k^n real altitude (from ground) (NED)
3) $h_{\text{sensor},k}^n$ height of the sensor w.r.t. the drone COM
4) $h_{g,k}^n$ ground height
5) ϕ_{vertical} angle with the vertical
6) $v_{\text{alt},k}$ ZM AWGN

Kalman Filter

Kalman filters

Two different Kalman filters have been developed:

1. **my_navigation_filter_adaptive**: the main filter used for the navigation of the drone. It is composed of two separates modules: one main module used exclusively for the navigation of the drone. The other module is used to deal with the unknown beacons position estimation, it has thus a state of variable dimension (it depends on the number of different anchors detected) that includes the coordinates of the unknown anchors and the square sum of the coordinates (4 scalars per anchor).
2. **detect_marker**: this smaller Kalman filter is used to estimate the position of the unknown markers representing the mission objective(s). This is described more in detail in the "Unstructured Environment Detection Section". It has as inputs the vision sensor measurements and the drone pose; its outputs are the marker position, orientation and velocity.

The Kalman filter description will start with the description of **my_navigation_filter_adaptive**.

Nominal State Vector

Nominal state vector

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{q}_k \\ \boldsymbol{\omega}_{b,k} \\ \boldsymbol{a}^b_{b,k} \\ \boldsymbol{a}^b_k \\ \boldsymbol{d}^b_k \\ \boldsymbol{p}^n_k \\ \boldsymbol{v}^n_k \\ h_{g,k} \end{bmatrix}$$

With:

$\boldsymbol{q}_k$	orientation in quaternion form	$\boldsymbol{\omega}_{b,k}$	gyroscope offset vector
$\boldsymbol{a}^n_{b,k}$	accelerometer bias vector	$\boldsymbol{a}^b_k$	linear acceleration vector
$\boldsymbol{d}^b_k$	magnetic disturbance vector	$\boldsymbol{p}^n_k$	position vector
$\boldsymbol{v}^n_k$	velocity vector	$h_{g,k}$	ground height

Nominal State Vector

Nominal state vector

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{q}_k \\ \boldsymbol{\omega}_{b,k} \\ \boldsymbol{a}^b_{b,k} \\ \boldsymbol{a}^b_k \\ \boldsymbol{d}^b_k \\ \boldsymbol{p}^n_k \\ \boldsymbol{v}^n_k \\ h_{g,k} \end{bmatrix}$$

With:

$\boldsymbol{q}_k$	orientation in quaternion form	$\boldsymbol{\omega}_{b,k}$	gyroscope offset vector
$\boldsymbol{a}^n_{b,k}$	accelerometer bias vector	$\boldsymbol{a}^b_k$	linear acceleration vector
$\boldsymbol{d}^b_k$	magnetic disturbance vector	$\boldsymbol{p}^n_k$	position vector
$\boldsymbol{v}^n_k$	velocity vector	$h_{g,k}$	ground height

State Vector

State vector

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{q}_k \\ \boldsymbol{\omega}_{b,k} \\ \boldsymbol{a}^b_{b,k} \\ \boldsymbol{a}^b_k \\ \boldsymbol{d}^b_k \\ \boldsymbol{p}^n_k \\ \boldsymbol{v}^n_k \\ h_{g,k} \end{bmatrix}$$

A high number of states (23) guarantees a good trade-off between the accuracy of the filter and the computational cost

- $\boldsymbol{q}_k, \boldsymbol{\omega}_{b,k}, \boldsymbol{p}^n_k, \boldsymbol{v}^n_k, \boldsymbol{a}^b_{b,k}$: common choice, necessary to estimate the state and correct IMU systematic errors
- $\boldsymbol{a}^b_k$ and $\boldsymbol{d}^b_k$: less common, increased robustness of the orientation estimate to external accelerations and magnetometer disturbances [15].
- $h_{g,k}$: used to fuse the altimeter measurements and to be resistent to obstacles and ground height changes [2].

Quaternion Approach

The orientation is described using quaternions because of:

- no singularities in the notation
- decreased computational cost

Only a *multiplicative* error is suitable to be used with quaternions (to avoid violating the condition on the unit norm).

The $\oplus$ operator is used with the following meaning when applied to quaternions (and is a simple $+$ otherwise):

$$\mathbf{q}_\mathrm{k} \oplus \boldsymbol{\theta}_\mathrm{k} = \mathbf{q}_\mathrm{k} \otimes \exp\left(\boldsymbol{\theta}_{q,k}\right)^{-1}$$

Where $\boldsymbol{\theta}_{q,k} = \mathrm{quaternion}(0, \boldsymbol{\theta}_k)$, and $\exp(\cdot)$ is the exponential of a quaternion.

Additionally, the quaternion is re-normalized after every prediction step with:

$$\boldsymbol{q}_k = \frac{\boldsymbol{q}_k}{\|\boldsymbol{q}_k\|_2}$$

State Update

The system's discrete time equations are:

$$\boldsymbol{x}_{k+1}^{-} = \begin{bmatrix} \boldsymbol{q}_{k+1}^{-} \\ \boldsymbol{\omega}_{b,k+1}^{-} \\ \boldsymbol{a}_{k+1}^{-} \\ \boldsymbol{d}_{k+1}^{-} \\ \boldsymbol{p}_{k+1}^{-} \\ \boldsymbol{v}_{k+1}^{-} \\ \boldsymbol{a}_{b,k+1}^{-} \\ h_{g,k+1}^{-} \end{bmatrix} = f(\boldsymbol{x}_k^{+}, \boldsymbol{u}_k) = \begin{bmatrix} \boldsymbol{q}_k^{+} \otimes \left[exp\left(\frac{1}{2} \boldsymbol{\omega}_{q,k} \Delta t \right) + \frac{\Delta t^2}{24} \begin{bmatrix} 0 \\ \boldsymbol{\omega}_{k-1} \times \boldsymbol{\omega}_k \end{bmatrix} \right] \\ \boldsymbol{\omega}_{b,k}^{+} \\ c_a \, \boldsymbol{a}_k^{+} \\ c_d \, \boldsymbol{d}_k^{+} \\ \boldsymbol{p}_k^{+} + \boldsymbol{v}_k^{+} \Delta t + \frac{1}{2} \left[C_b^n \left(-\boldsymbol{y}_{A,k} + \boldsymbol{a}_{b,k}^{+} \right) - \boldsymbol{g}^n \right] \Delta t^2 \\ \boldsymbol{v}_k^{+} + \left[C_b^n \left(-\boldsymbol{y}_{A,k} + \boldsymbol{a}_{b,k}^{+} \right) - \boldsymbol{g}^n \right] \Delta t \\ \boldsymbol{a}_{b,k}^{+} \\ h_{g,k+1}^{+} \end{bmatrix} \quad c_a, c_d \in [0; 1]$$

These equations are easily obtained by discretizing the continuous time system equations.

Error State

$$\boldsymbol{x}_{true,k} = \boldsymbol{x}_k \oplus \boldsymbol{x}_{\varepsilon,k}; \qquad \boldsymbol{x}_{\varepsilon,k} = \begin{bmatrix} \boldsymbol{\theta}_{\varepsilon,k} \\ \boldsymbol{\omega}_{b,\varepsilon,k} \\ \boldsymbol{a}^b_{\varepsilon,k} \\ \boldsymbol{d}^b_{\varepsilon,k} \\ \boldsymbol{p}^n_{\varepsilon,k} \\ \boldsymbol{v}^n_{\varepsilon,k} \\ \boldsymbol{a}^b_{b,\varepsilon,k} \\ h_{g,\varepsilon,k+1} \end{bmatrix} = f_\varepsilon\left(\boldsymbol{x}_{\varepsilon,k-1}\right) + \boldsymbol{w}_k$$

nominal state

With:

$\boldsymbol{\theta}_{\varepsilon,k}$	orientation error vector	$\boldsymbol{\omega}_{b,\varepsilon,k}$	gyro zero-rate offset vector
$\boldsymbol{a}^b_{\varepsilon,k}$	linear acceleration error	$\boldsymbol{d}^b_{\varepsilon,k}$	magnetic disturbance error vector
$\boldsymbol{p}_{\varepsilon,k}$	position error vector	$\boldsymbol{v}_{\varepsilon,k}$	velocity error vector
$\boldsymbol{a}_{b,\varepsilon,k}$	accelerometer bias error vector	$\mathrm{h_{g,\varepsilon,k}}$	ground height error

Error Process

$$\left.\frac{\partial f_\varepsilon}{\partial \boldsymbol{x}_\varepsilon}\right|_{\hat{\boldsymbol{x}}_{\varepsilon_{k-1|k-1}},\boldsymbol{u}_k} = F_{\varepsilon,k} =$$

$$= \begin{bmatrix} I_3 - \Delta t(\boldsymbol{\omega}_k)_\times & -I_3\Delta t & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 & 0_{3\times1} \\ 0_3 & I_3 & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 & 0_{3\times1} \\ 0_3 & 0_3 & (1-\lambda\Delta t)I_3 & 0_3 & 0_3 & 0_3 & 0_3 & 0_{3\times1} \\ 0_3 & 0_3 & 0_3 & (1-\sigma\Delta t)I_3 & 0_3 & 0_3 & 0_3 & 0_{3\times1} \\ -\frac{1}{2}C_{s_a}^n\left(-y_A^b + a_{b,k}^b\right)\Delta t^2 & 0_3 & 0_3 & 0_3 & I_3 & I_3\Delta t & +\frac{1}{2}C_{s_a}^n\Delta t^2 & 0_{3\times1} \\ -C_{s_a}^n\left(-y_A^b + a_{b,k}^b\right)\Delta t & 0_3 & 0_3 & 0_3 & 0_3 & I_3 & +C_{s_a}^n\Delta t & 0_{3\times1} \\ 0_3 & 0_3 & 0_3 & 0_3 & 0_3 & 0_3 & I_3 & 0_{3\times1} \\ 0_{1\times3} & 0_{1\times3} & 0_{1\times3} & 0_{1\times3} & 0_{1\times3} & 0_{1\times3} & 0_{1\times3} & 1 \end{bmatrix}$$

The components of $\hat{\boldsymbol{x}}_{\varepsilon,k}^+$ are used to update the state vector at every filter step. Thus, the error state is reset to zero at the end of each correction step, and $\hat{\boldsymbol{x}}_{k+1}^- = 0$.

Kalman Filter Equations

Prediction

Nominal state prediction:

$$\hat{\boldsymbol{x}}_k^- = f(\hat{\boldsymbol{x}}_{k-1}^+, \boldsymbol{u}_k)$$

Error state prediction:

$$\boldsymbol{x}_{\varepsilon,k+1}^- = \mathbf{0}$$

Error state covariance matrix prediction:

$$P_k^- = F_{\epsilon,k}\, P_{k-1}^+ F_{\epsilon,k}^T + Q_k$$

Error computation:

$$\boldsymbol{z}_{\epsilon,k} = \boldsymbol{y} - \boldsymbol{h}(\hat{\boldsymbol{x}}_k^-) = \boldsymbol{h}_\varepsilon(\hat{\boldsymbol{x}}_k^-)$$

Correction

Kalman gain computation:

$$K_k = P_k^- H_k^T \left(H_k P_k^- H_k^T + R_k\right)^{-1}$$

Error state correction:

$$\hat{\boldsymbol{x}}_{\varepsilon,k}^+ = \cancel{\hat{\boldsymbol{x}}_{\varepsilon,k}^-} + K_k\left(\boldsymbol{z}_{\varepsilon,k} - \cancel{H_k \hat{\boldsymbol{x}}_{\varepsilon,k}^-}\right) = K_k\, \boldsymbol{z}_{\varepsilon,k}$$

Error covariance matrix correction:

$$P_k^+ = (I - K_k H_k) P_k^-$$

State correction and error state update:

$$\hat{\boldsymbol{x}}_k^+ = \hat{\boldsymbol{x}}_k^- \oplus \hat{\boldsymbol{x}}_{\varepsilon,k}^+; \qquad \hat{\boldsymbol{x}}_{\varepsilon,k}^+ = 0$$

Sequential Kalman Filter Equations

Initialization:

$$\hat{\boldsymbol{x}}_{k,0}^{+} = \hat{\boldsymbol{x}}_{k}^{-}$$

$$P_{k,0}^{+} = P_{k}^{-}$$

Cycle over $i \in \{1, \dots, D\}$ (where D is the number of sequential steps, or of independent measurements):

$$K_{k,i} = P_{k,i-1}^{+} H_{k,i}^{T} \left(H_{k,i} P_{k,i-1}^{+} H_{k,i}^{T} + R_{k,i} \right)^{-1}$$

$$\hat{\boldsymbol{x}}_{\varepsilon,k,i}^{+} = \cancel{\hat{\boldsymbol{x}}_{\varepsilon,k,i-1}^{+}} + K_{k,i} \left(\boldsymbol{z}_{\varepsilon,k,i} - \cancel{H_{k,i} \hat{\boldsymbol{x}}_{\varepsilon,k,i-1}^{+}} \right) = K_{k,i}\, \boldsymbol{z}_{\varepsilon,k,i}$$

$$\hat{\boldsymbol{x}}_{k,i}^{+} = \hat{\boldsymbol{x}}_{k,i-1}^{+} \oplus \hat{\boldsymbol{x}}_{\varepsilon,k,i}^{+}$$

$$P_{k,i}^{+} = \left(I - K_{k,i} H_{k,i} \right) P_{k,i-1}^{+}$$

Process covariance matrix

The process covariance matrix used in the Kalman filter is diagonal.

In addition, its terms are a function of time (linear or quadratic).

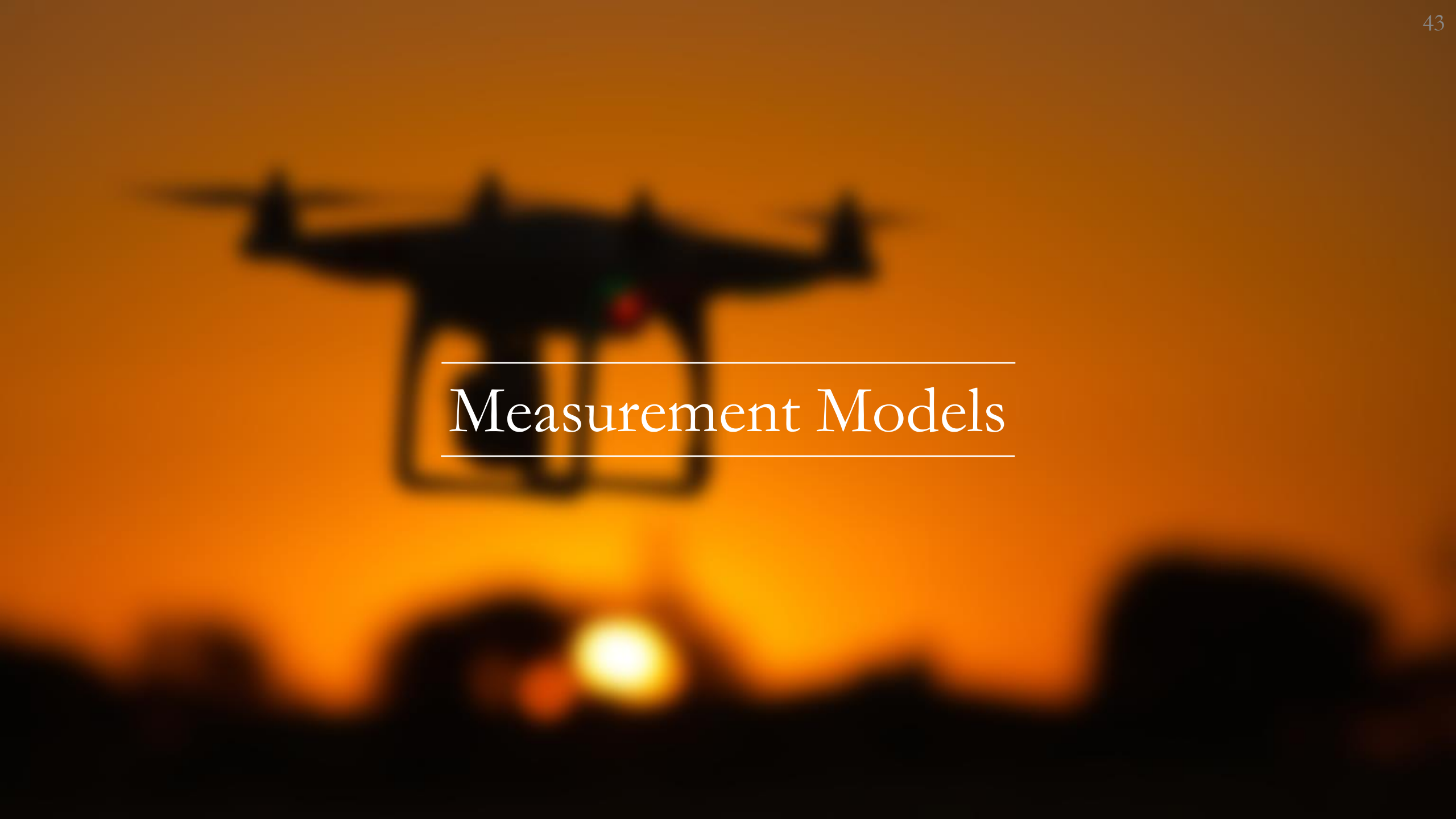
$$Q = \text{diag}\left(\begin{bmatrix} n_{\text{gyro}} \cdot \Delta t^2 \cdot I_{3,1} \\ n_{\text{gyro bias}} \cdot \Delta t^2 \cdot I_{3,1} \\ n_{\text{acc bias}} \cdot \Delta t \cdot I_{3,1} \\ n_{\text{lin acc}} \cdot \Delta t \cdot I_{3,1} \\ n_{\text{magn dist}} \cdot \Delta t \cdot I_{3,1} \\ n_{\text{pos}} \cdot \Delta t^2 \cdot I_{3,1} \\ n_{\text{vel}} \cdot \Delta t \cdot I_{3,1} \\ n_{\text{ground height}} \cdot \Delta t \end{bmatrix}\right)$$

Measurement Covariance Matrix

The measurement covariance matrix used in the Kalman filter is diagonal.

This choice reflects the fact that the measurements are assumed to be *uncorrelated*.

$$R = \text{diag}\left(\begin{bmatrix} n_{\text{acc}} \cdot I_{3,1} \\ n_{\text{mag}} \cdot I_{3,1} \\ n_{\text{GPS, pos}} \cdot I_{3,1} \\ n_{\text{GPS, vel}} \cdot I_{3,1} \\ n_{\text{UWB}} \\ n_{\text{vis, pos}} \cdot I_{3,1} \\ n_{\text{vis, orient}} \cdot I_{3,1} \\ n_{\text{bar}} \\ n_{alt} \end{bmatrix}\right)$$



Measurement Models

Measurement Models

Due to the equations considered, it can be demonstrated that some measurements do not affect some states (i.e.: the magnetometer measurements do not influence the ground-height state, nor should they influence it).

Therefore, only a subset of the state and of the state covariance matrix is considered for each sensor used. This makes the filter less computationally expensive, without any loss in tracking performance.

Measurement Models

In general, the Jacobian of the measurement is calculated as follows:

$$\boldsymbol{H}_k = \left.\frac{\partial \boldsymbol{z}_k}{\partial \boldsymbol{x}_\varepsilon}\right|_{\boldsymbol{x}} = \left.\frac{\partial \boldsymbol{h}_\varepsilon}{\partial \boldsymbol{x}_\varepsilon}\right|_{\boldsymbol{x}} = \left.\frac{\partial \boldsymbol{h}_\varepsilon}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}} \left.\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{x}_\varepsilon}\right|_{\boldsymbol{x}} = \boldsymbol{H}_x \boldsymbol{X}_{\delta x}$$

With:

$$\boldsymbol{X}_{\delta x} = \begin{bmatrix} \boldsymbol{Q}_{\delta\boldsymbol{\theta}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}$$

$$\boldsymbol{Q}_{\delta\boldsymbol{\theta}} = \frac{\partial(\boldsymbol{q} \otimes \delta\boldsymbol{q})}{\partial \delta\boldsymbol{\theta}} = \left.\frac{\partial([\boldsymbol{q}]_L \delta\boldsymbol{q})}{\partial \delta\boldsymbol{q}}\right|_{\boldsymbol{q}} \cdot \left.\frac{\partial \begin{bmatrix} 1 \\ \frac{1}{2}\delta\boldsymbol{\theta} \end{bmatrix}}{\partial \delta\theta}\right|_{\delta\,\hat{\boldsymbol{\theta}}=0} = \begin{bmatrix} q_0 & -\boldsymbol{q}_v \\ \boldsymbol{q}_v^T & q_0 I_3 + (\boldsymbol{q}_v)_\times \end{bmatrix} \cdot \frac{1}{2} \begin{bmatrix} 0_{1,3} \\ \boldsymbol{I}_3 \end{bmatrix}$$

Measurement Models

Different errors have been considered for the various sensors.

For most sensors, the error considered for the i-th sensor sensor at time k ($\boldsymbol{z}_{i,k}$) is computed as:

$$\boldsymbol{z}_{i,k} = \text{i}^{\text{th}} \text{ sensor measurement } - \text{predicted i}^{\text{th}} \text{ sensor measurement}$$

Measurement Models

Accelerometers

Considered error states:

$$\boldsymbol{x}_{\varepsilon,k} = [\boldsymbol{\theta}_{\varepsilon,k} \quad \boldsymbol{\omega}_{b,\varepsilon,k} \quad \boldsymbol{a}_{b,\varepsilon,k} \quad \boldsymbol{a}_{\varepsilon,k} \quad \boldsymbol{p}_{\varepsilon,k} \quad \boldsymbol{v}_{\varepsilon,k}]^T$$

Observation:

$$\boldsymbol{z}_{A,k} = \boldsymbol{y}^s_{\mathrm{A,k}} - \left(-\hat{\boldsymbol{a}}^b_k + \hat{\boldsymbol{a}}^b_{b,k} + \hat{\boldsymbol{g}}^s\right)$$

Observation Jacobian matrix:

$$H_{\mathrm{x,A}} = \left[\frac{\partial(\boldsymbol{R}^b_n(q)\,\boldsymbol{g}^n)}{\partial \boldsymbol{q}} \quad 0_3 \quad I_3 \quad -I_3 \quad 0_3 \quad 0_3\right]^T$$

Magnetometer

Considered error states:

$$\boldsymbol{x}_{\varepsilon,k} = [\boldsymbol{\theta}_{\varepsilon,k} \quad \boldsymbol{\omega}_{b,\varepsilon,k} \quad \boldsymbol{a}_{b,\varepsilon,k} \quad \boldsymbol{d}_{m,\varepsilon,k} \quad \boldsymbol{p}_{\varepsilon,k} \quad \boldsymbol{v}_{\varepsilon,k}]^T$$

Observation:

$$\boldsymbol{z}_{m,k} = \boldsymbol{y}_{\mathrm{M}} - \left(\hat{\boldsymbol{m}}^s_k + \hat{\boldsymbol{d}}_{m,k}\right)$$

Observation Jacobian matrix:

$$H_{\mathrm{x},M} = \left[\frac{\partial(\boldsymbol{R}^b_n(q)\,\boldsymbol{m}^n)}{\partial \boldsymbol{q}} \quad 0_3 \quad 0_3 \quad I_3 \quad 0_3 \quad 0_3\right]^T$$

Measurement Models

GPS

Considered error states:

$$\boldsymbol{x}_{\varepsilon,k} = [\boldsymbol{\theta}_{\varepsilon,k} \quad \boldsymbol{\omega}_{b,\varepsilon,k} \quad \boldsymbol{a}_{b,\varepsilon,k} \quad \boldsymbol{p}_{\varepsilon,k} \quad \boldsymbol{v}_{\varepsilon,k}]^T$$

Observation:

$$\boldsymbol{z}_{\text{pGPS},k} = \boldsymbol{p}_{\text{GPS},k} - \hat{\boldsymbol{p}}_k$$

$$\boldsymbol{z}_{\text{vGPS},k} = \boldsymbol{v}_{\text{GPS},k} - \hat{\boldsymbol{v}}_k$$

Observation Jacobian matrix:

$$H_{x,\text{GPS}} = \begin{bmatrix} 0_3 & 0_3 & 0_3 & I_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & I_3 \end{bmatrix}$$

UWB (known beacon position)

Considered error states:

$$\boldsymbol{x}_{\varepsilon,k} = [\boldsymbol{\theta}_{\varepsilon,k} \quad \boldsymbol{\omega}_{b,\varepsilon,k} \quad \boldsymbol{a}_{b,\varepsilon,k} \quad \boldsymbol{p}_{\varepsilon,k} \quad \boldsymbol{v}_{\varepsilon,k}]^T$$

Observation:

Vector of distances measured by the anchors

$$\boldsymbol{z}_{UWB,a_i,k} = \boldsymbol{d}_{UWB,\boldsymbol{a}_i,k} - \hat{\boldsymbol{d}}_{\boldsymbol{a}_i,k}$$

Vector of distances predicted with the nominal state

Observation Jacobian matrix:

$$H_{x,\text{UWB}} =$$

$$\begin{bmatrix} 0_{d,3} & 0_{d,3} & 0_{d,3} & \begin{bmatrix} \frac{x_k - a_{x,1}}{\boldsymbol{d}_{UWB,a_1,k}} & \frac{y_k - a_{y,1}}{\boldsymbol{d}_{UWB,a_1,k}} & \frac{z_k - a_{z,1}}{\boldsymbol{d}_{UWB,a_1,k}} \\ \vdots & \vdots & \vdots \end{bmatrix} & 0_{d,3} \end{bmatrix}$$

$a_{x,i}$ = x component of the i-th anchor (known)

Measurement Models

Unknown UWB

Two different methods are used when dealing with UWB beacons with unknown position:

1. `fuse_unkn_uwb`: the Kalman filter estimates both the drone states and sensor biases AND the beacon position, simultaneously. In this case, global stability of the algorithm is not guaranteed and is heavily influenced by the initial accuracy of the relative position of the UWB anchor w.r.t. the drone at the end of the initialization phase.
2. `detect_unkn_uwb`: the Kalman filter only estimates the unknown UWB beacon position, without influencing the drone states. A total of four states are considered: the anchor position coordinates and the sum of the squares of the coordinates. This fourth state is used in order to implement a spherical constraint in the filtering algorithm, ensuring global stability.

Measurement Models

Unknown UWB – fuse_unknown_uwb

Considered error states:

Additional states representing the unkn anchor coordinates in nav frame

$$\boldsymbol{x}_{\varepsilon,k} = [\boldsymbol{\theta}_{\varepsilon,k} \quad \boldsymbol{\omega}_{b,\varepsilon,k} \quad \boldsymbol{a}_{b,\varepsilon,k} \quad \boldsymbol{p}_{\varepsilon,k} \quad \boldsymbol{v}_{\varepsilon,k} \quad \boldsymbol{a}_{\text{unkn},i}]^T$$

Observation:

$$\boldsymbol{z}_{UWB,a_i,k} = \boldsymbol{d}_{UWB,\boldsymbol{a}_i,k} - \widehat{\boldsymbol{d}}_{\boldsymbol{a}_i,k}$$

Observation Jacobian matrix:

$$H_{x,\text{ unkn UWB}} =$$

$$\begin{bmatrix} 0_{d,3} & 0_{d,3} & 0_{d,3} & \begin{bmatrix} \frac{x_k - a_{x,1}}{d_{UWB,a_1,k}} & \frac{y_k - a_{y,1}}{d_{UWB,a_1,k}} & \frac{z_k - a_{z,1}}{d_{UWB,a_1,k}} \\ \vdots & \vdots & \vdots \end{bmatrix} & 0_{d,3} & -\begin{bmatrix} \frac{x_k - a_{x,1}}{d_{UWB,a_1,k}} & \frac{y_k - a_{y,1}}{d_{UWB,a_1,k}} & \frac{z_k - a_{z,1}}{d_{UWB,a_1,k}} \\ \vdots & \vdots & \vdots \end{bmatrix} \end{bmatrix}$$

Measurement Models

Unknown UWB – detect_unknown_uwb

Considered error states:

$$\boldsymbol{x}_{\varepsilon,k} = \begin{bmatrix} \left\| \boldsymbol{a}_{\text{unkn},i} \right\|_2^2 & \boldsymbol{a}_{\text{unkn},i} \end{bmatrix}^T$$

Observation:

$$\boldsymbol{z}_{UWB,a_i,k} = d_{UWB,a_{\text{unkn},i},k}^2 - \|\hat{\boldsymbol{p}}_{\boldsymbol{k}}\|_2^2 - \left\| \hat{\boldsymbol{a}}_{\text{unkn},i} \right\|_2^2 + 2 \cdot \hat{\boldsymbol{p}}_{\boldsymbol{k}} \cdot \hat{\boldsymbol{a}}_{\text{unkn},i}$$

Observation Jacobian matrix:

$$H_{x,\,\text{unkn UWB}} = \begin{bmatrix} -1 & +2 \cdot \hat{\boldsymbol{p}}_{\boldsymbol{k}} \end{bmatrix}$$

Measurement Models

Vision (known marker position)

Considered error states:

$$\boldsymbol{x}_{\varepsilon,k} = [\boldsymbol{\theta}_{\varepsilon,k} \quad \boldsymbol{\omega}_{b,\varepsilon,k} \quad \boldsymbol{a}_{b,\varepsilon,k} \quad \boldsymbol{p}_{\varepsilon,k} \quad \boldsymbol{v}_{\varepsilon,k}]^T$$

Observation:

$$\boldsymbol{z}_{p\mathrm{Vis},m_i,k} = \boldsymbol{y}_{\mathrm{vis,m_i},k} - \hat{\boldsymbol{p}}_{\boldsymbol{k}}$$

$$\boldsymbol{z}_{\theta Vis,m_i k} = \boldsymbol{y}^{m}_{c\ vis,\mathrm{m_i},k} - \widehat{\boldsymbol{R}^{m}_{c}}_{k}$$

Observation Jacobian matrix:

$$H_{x,\mathrm{vis,m_i}} = \begin{bmatrix} 0_3 & 0_3 & 0_3 & I_3 & 0_3 \\ I_3 & 0_3 & 0_3 & 0_3 & 0_3 \end{bmatrix}$$

Measurement Models

Barometer

Considered error states:

$$\boldsymbol{x}_{\varepsilon,k} = [\boldsymbol{\theta}_{\varepsilon,k} \quad \boldsymbol{\omega}_{b,\varepsilon,k} \quad \boldsymbol{a}_{b,\varepsilon,k} \quad \boldsymbol{p}_{\varepsilon,k} \quad \boldsymbol{v}_{\varepsilon,k}]^T$$

Observation:

$$z_{\text{bar},k} = -y_{\text{bar},k} - \hat{z}_k$$

Observation Jacobian matrix:

$$H_{\text{x,bar}} = [0_{1,3} \quad 0_{1,3} \quad 0_{1,3} \quad [0 \quad 0 \quad 1] \quad 0_{1,3}]$$

Altimeter

Considered error states:

$$\boldsymbol{x}_{\varepsilon,k} = [\boldsymbol{\theta}_{\varepsilon,k} \quad \boldsymbol{\omega}_{b,\varepsilon,k} \quad \boldsymbol{a}_{b,\varepsilon,k} \quad \boldsymbol{p}_{\varepsilon,k} \quad \boldsymbol{v}_{\varepsilon,k} \quad h_{\varepsilon,g}]^T$$

Observation:

$$z_{\text{alt},k} = y_{\text{alt},k}$$

$$-\left(-\hat{z}_{down,k} - [0 \quad 0 \quad 1]\, \widehat{R}_b^n\, \boldsymbol{p}_{\text{alt sensor}|COM} - \hat{h}_g\right) / \cos(\phi_{\text{vertical}})$$

Observation Jacobian matrix:

$$H_{\text{x,alt}} = \left[\frac{\partial z_{alt,k}}{d\boldsymbol{q}} \quad 0_{1,3} \quad 0_{1,5} \quad \frac{-1}{\cos\phi} \quad 0_{1,5} \quad \frac{-1}{\cos\phi}\right]$$

Adaptive Kalman Filtering

Adaptively-Robust Filter

Adaptive Kalman Filter

The adaptiveness in Kalman filtering is the ability to resist to errors in the process model.

Three main methods are used to achieve adaptiveness in the scientific literature:

- *Multiple Model-based Adaptive Estimation* (MMAE), characterised by an very high computational cost and the necessity to model every condition of interest
- *Innovation-based Adaptive Estimation* (IAE) with moderate computational cost and memory burden.
- *Adaptive factor based*, with a small computational cost.

Adaptively-Robust Filter

Adaptive Kalman Filter Equations

The strategy adopted in the filter developer is the adaptive factor based one [5].

The Kalman equations of the adaptive Kalman filter become:

$$\bar{K}_k = \frac{1}{\alpha_k} P_{k|k-1} H_k^T \left(\frac{1}{\alpha_k} H_k P_{k|k-1} H_k^T + R_k \right)$$

$$x_{k|k} = x_{k|k-1} + \bar{K}_k \left(z_k - H_k x_{k|k-1} \right)$$

$$P_k = P_{k|k-1} - \bar{K}_k H_k P_{k|k-1}$$

Adaptively-Robust Filter

Adaptive Factor

The innovation (predicted residual) is more suitable to express perturbations of the dynamic system. The test statistic based on the predicted residual is $\bar{V}_k$, while the covariance matrix is $P_{\bar{V}_k}$:

$$\bar{V}_k = h\left(x_{k|k-1}\right) - z_k \qquad P_{\bar{V}_k} = H_k P_{k|k-1} H_k^T + R_k$$

The error detection statistic chosen is:

$$\Delta\bar{V}_k = \left(\frac{\bar{V}_k^T \bar{V}_k}{\sqrt{\mathrm{tr}\left(P_{\bar{V}_k}\right)}} \right)^{\frac{1}{2}}$$

Finally, a two-segment adaptive factor is defined by:

$$\alpha_k = \begin{cases} 1 & |\Delta\bar{V}_k| \le c \\ \dfrac{c}{|\Delta\bar{V}_k|} & |\Delta\bar{V}_k| > c \end{cases}$$

with $1.0 \le c \le 1.5$ (value used: $c = 1.3$)

Adaptively-Robust Filter

Adaptively-Robust Filter

The adaptive filter requires reliable observations in order to function correctly. In the presence of faulty measurements/outliers the performance will degrade quickly and get worse than the one of the standard Kalman filter.

A robust strategy is implemented in order to mitigate the effect of measurement outliers and achieve good performance.

The filter gain is modified by introducing the equivalent covariance matrix $\bar{R}_k$:

$$\bar{K}_k = \frac{1}{\alpha_k} P_{k|k-1} H_k^T \left(\frac{1}{\alpha_k} H_k P_{k|k-1} H_k^T + \bar{R}_k \right)$$

Adaptively-Robust Filter

Scaling Factor

The scaling factor λ_i used to compute the equivalent covariance matrix $\bar{R}_k$ is:

$$\lambda_i = \begin{cases} 1 & \left|\bar{V}_{\bar{X}_{k_i}}\right| \leq c \\ \dfrac{\left|\bar{V}_{\bar{X}_{k_i}}\right|}{c} & \left|\bar{V}_{\bar{X}_{k_i}}\right| > c \end{cases}$$

with $1.0 \leq c \leq 1.5$ (value used: $c = 1.3$)

Where $\left|\bar{V}_{\bar{X}_{k_i}}\right|$ is the absolute value of the element of the standardized predicted residuals.

The equivalent covariance matrix is computed by scaling the measurement covariance matrix with:

$$\bar{R}_k = \mathrm{diag}(\lambda_i)\, R_k$$

Adaptively-Robust Filter

Mahalanobis Distance as a Statistical Test

The Mahalanobis distance M_k^2 is :

$$M_k^2 = (z_k - \bar{z}_k)^T \left(P_{\bar{z}_k}\right)^{-1} (z_k - \bar{z}_k)$$

With Gaussian distributed measurements, the square of the Mahalanobis distance should be Chi-square distributed with degree of freedom r (dimension of the observation vector). This fact is used in order to perform a hypothesis testing to determine whether the measurement is an outlier/faulty measurement or not.

An appropriate confidence level α is chosen to compute the α-quantile of the Chi-squared distrubition with r degrees of freedom. If the square of the Mahalanobis distance is bigger than this value, the null hypothesis is rejected and the measurement is regarded as an outlier.

Adaptively-Robust Filter

Pseudo Code

if $M_k^2 \leq \chi_\alpha$

    % The measurement is NOT regarded as an outlier => adaptive & robust filter algorithm

    adaptive filter (innovation -> adaptive factor -> adaptive gain)

    robust filter (scaling factor -> $\bar{R}_k$ -> robust gain)

else $M_k^2 > \chi_\alpha$

    % the measurement is regarded as an outlier => no adaptive factor

    robust filter (scaling factor -> $\bar{R}_k$ -> robust gain)

end

Adaptively-Robust Filter

Algorithm Characteristics

Compared to other techniques (MMAE and IAE), the adaptively-robust method implemented:

- Is considerably less computationally expensive.
- Has smaller memory burden.
- Is more sensitive in reflecting the observational errors in the present epoch (compared to both IAE windowing and forgetting factor methods).
- Provides resistance to both modeling errors and measurements errors

Unstructured Environment Detection

ArUco Markers Detection

In addition to be used to aid the navigation of the drone, ArUco markers can be used to identify and localize the mission objective.

Some markers with unknown coordinates to the navigation system have been introduced. Once they are detected by the camera, a simple Kalman filter is executed to estimate their position and orientation.

The output of the Kalman filter is the estimated position and orientation in the navigation frame. This information can be used to aid the drone in completing its mission.

In the absence of other sensors such as GPS or UWB, the newly identified marker (once the estimate is reliable enough) could also be used to aid the navigation, under the hypothesis that it is fixed in the navigation frame (or that its motion is known).

ArUco Markers Detection

The Kalman filter used to identify the ArUco markers is implemented with the system object `detect_marker`:

```
[p_mn_n, q_mn, v_mn_n] = detect_aruco(Tcm, id, p_bn_n, q_bn)
```

The math behind the filter is similar to the one used to model the vision sensor. The vision sensor output and the pose of the drone in the nav. frame, are used to determine the marker position in nav. frame. This value is then filtered using a Kalman filter.

To consider the fact that the vision sensor is more accurate if the camera-marker distance is small, the measurement noise is adaptively changed and is quadratically dependent on this distance.

ArUco Markers Detection

This situation is interesting in the case that the mission objective (the package to be picked up) is identified by means of an ArUco marker. The position, attitude and (eventually) the velocity of the package is outputted by this separate Kalman filter. It may then be used as an input for the controller, in order to reach and pick up the package.

The velocity of the marker (and thus of the object) is not necessarly supposed to be null because, for example, the package may be located onto a moving conveyor belt. A switch has been developed for the Kalman filter so that if it is known a-priori that the package will not be moving this information is not discarded and the filter is simplified so that it does not estimate the velocity of the package too (which would decrease performance).

ArUco Markers Detection

ArUco Indirect Kalman Filter Matrices

States and error state:

$$x_k = [\boldsymbol{q}_k \quad \boldsymbol{p}_k \quad \boldsymbol{v}_k]^T \qquad\qquad x_{\varepsilon,k} = [\boldsymbol{\theta}_{\varepsilon,k} \quad \boldsymbol{p}_{\varepsilon,k} \quad \boldsymbol{v}_{\varepsilon,k}]^T$$

State transition matrix:

$$F = \begin{bmatrix} I_3 & 0_3 & 0_3 \\ 0_3 & I_3 & \Delta t\, I_3 \\ 0_3 & 0_3 & I_3 \end{bmatrix}$$

Both the process noise covariance matrix and the measurement covariance matrix are diagonal matrices with appropriate elements.

ArUco Markers Detection

ArUco Indirect Kalman Filter Matrices

Observations:

$$\boldsymbol{z}_{pos,k} = \boldsymbol{p}^{n}_{marker} - \boldsymbol{p}^{n}_{IMU}$$

$$\boldsymbol{z}_{orient} = \text{rotvec}(\mathbf{q}_{\text{marker}} \otimes \mathbf{q}^{*}_{\text{IMU}})$$

$$z = \begin{bmatrix} \boldsymbol{z}_{orient} \\ \boldsymbol{z}_{pos,k} \end{bmatrix}$$

Observation matrix:

$$H_k = \begin{bmatrix} I_3 & 0_3 & 0_3 \\ 0_3 & I_3 & 0_3 \end{bmatrix}$$

The Kalman filter equations are the same equations as the standard indirect Kalman filter written previously. Thus, they have not been written again.

UWB Beacons Detection

It has been supposed that some of the UWB beacons may have an unknown position in the navigation frame. This situation is interesting (mainly because of the fact that a calibration process is difficult and time consuming and not always feasible) and has already received some attention in the scientific literature.

It is possible to dynamically estimate the position of the beacons and use it to aid the navigation of the drone.

This has been done considering the fact that the current position estimate is always sufficiently accurate.

The estimation is again implemented using a standard Kalman filter.

UWB Beacons Detection

Before using the Kalman filter for the unknown beacon measurements, an initialization process is necessary.

This initialization is performed by acquiring a sufficient number of measurements (theoretically at least four) and solving the multilateration problem. This problem is solved with a *Spherical Least-Squares estimator* [9].

The particularity of this algorithm is that it estimates an additional quantity: the square of the sum of the coordinates. This is done in order to force the spherical constraint onto the LS problem:

$$\begin{bmatrix} 1 & -2x_1 & -2y_1 & -2z_2 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} x^2+y^2+z^2 \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} d_1^2 - x_1^2 - y_1^2 - z_1^2 \\ \vdots \end{bmatrix}$$

$$A \cdot \boldsymbol{x} = \boldsymbol{b} \qquad \Rightarrow \qquad \hat{\boldsymbol{x}} = (A^T A)^{-1} A^T \boldsymbol{b}$$

UWB Beacons Detection

For the initialization, due to the low velocity of the drone, it is recommended that several data points are acquired before finishing the initialization.

A stop criterion on the initialization, used in the developed code, is based on the number of measurements acquired for each unknown beacon. Once a certain choosen threshold is reached, the initialization end. This choice is a result of the trade-off between memory requirements, computational cost, accuracy and speed of the estimation (the threshold used in the simulations is 50).

Another stop criterion could be based on the drone displacement from the first observation of the anchor. Once it has traveled (independently) a "significant" distance in all three directions, the initialization is stopped. This would be useful (to a certain degree) when dealing with "singular" trajectories (where the UWB position is not observable).

UWB Beacons Detection

Two methods with different characteristics have been implemented:

1. `fuse_unknown_uwb`: this algorithm is able to simultaneously estimate the position of the anchors and the state of the drone. This is implemented as seen in the previous sections. The trade-off of this implementation is that the stability is not guaranteed. The stability is highly dependent on the relative anchor position estimated during the initialization phase. Because of this, a stop criterion on the initialization phase based only on the number of data points is not enough, especially in the case of "singular" trajectories (where the UWB position is not observable). In this case, a stop criterion based on the displacement vector (not distance) could be helpful but would require other precautions in order to limit the number of data points. Instead, the second method developed may be an efficient, and iterative, solution.
   This implementation is based on [11].

UWB Beacons Detection

Two methods with different characteristics have been implemented:

2. `detect_unknown_uwb`: the spherical LS estimator equations are used to develop an iterative error-state Kalman filter. The estimated quantities are only the anchors position (no drone states). For every anchor 4 states are necessary in order to use the sperical implementation. This algorithm has the big advantage of being globally stable and able to estimate the anchor position given a non-singular trajectory.
   Because of this, it is the recommended solution when dealing with singular trajectories. After the anchor position has been estimated with a sufficient accuracy, the other algorithm can be used.
   This is an original solution based on the spherical LSE.

MATLAB Implementation

MATLAB

- Software: MATLAB R2020a
- Toolboxes:
  - Sensor Fusion and Tracking Toolbox
  - Mapping Toolbox
- Object-oriented approach: System Objects (SO)

SO & Functions:

- `main`
- `my_navigation_filter`
- `my_navigation_filter_adaptive`
- `uwb_sensor`
- `aruco_vision_sensor`
- `barometer_sensor`
- `altimeter_sensor`

Internal functions:

- `imuSensor`
- `gpsSensor`
- `HelperScrollingPlotter`
- `PoseViewerWithSwitches`

main_1

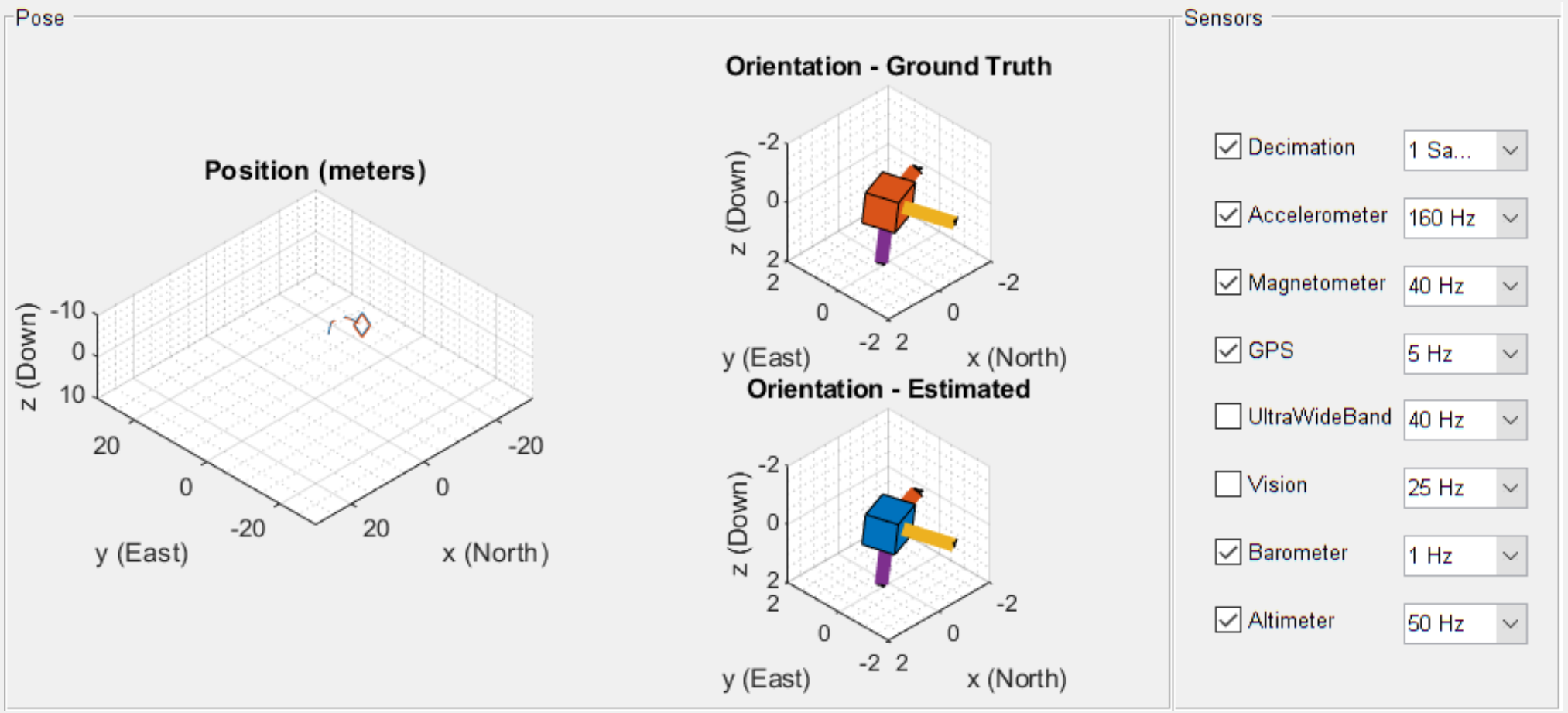
It is used to test the behaviour of the fused sensors with a real(istic) trajectory. The trajectory is loaded from a MATLAB .mat files made available by Mr MATLAB himself.

It visualizes the estimate of the pose of the drone (both position [3D] and orientation) and compares it to the real pose thanks to the `poseViewerWithSwitches` function.

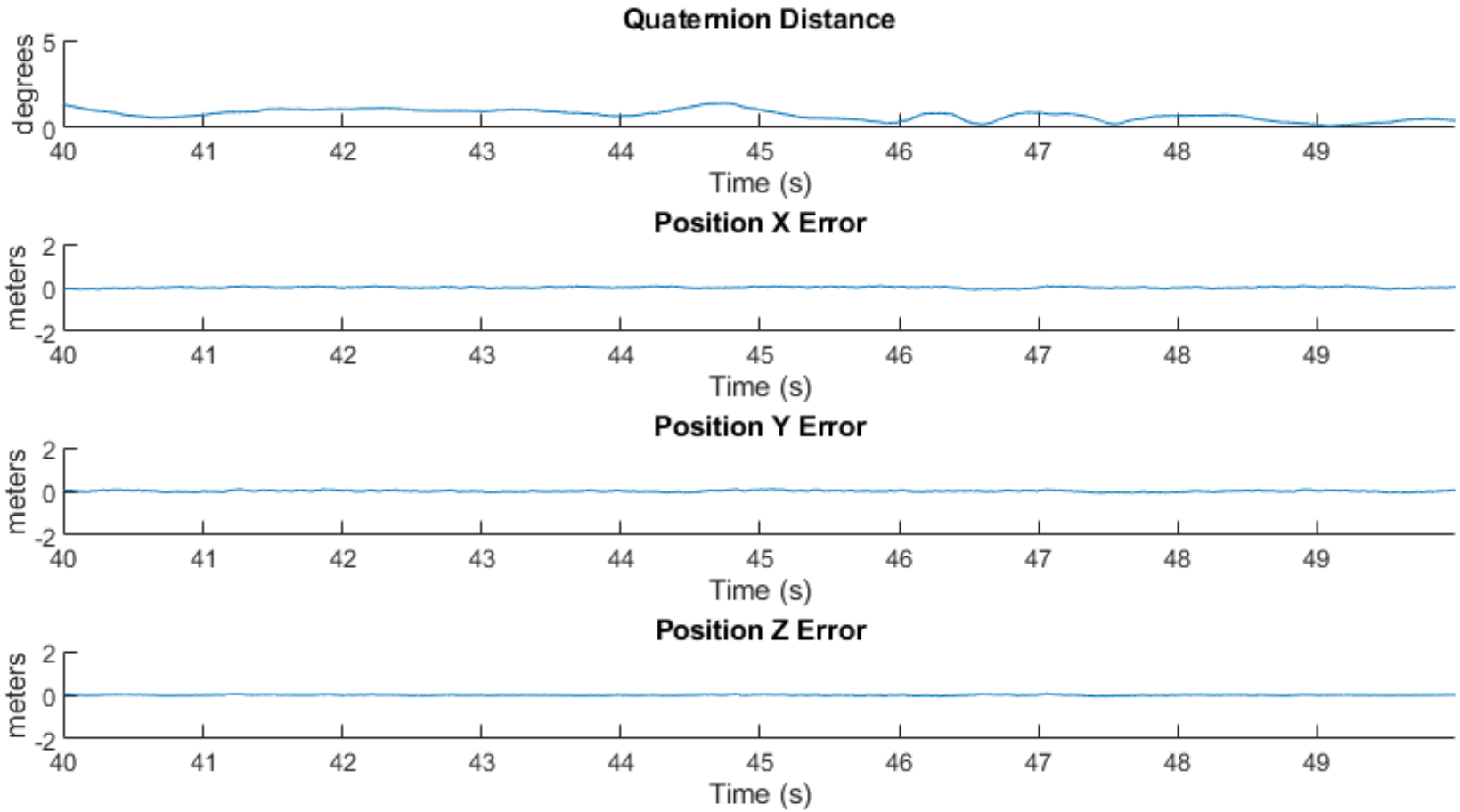
In addition to plotting the estimated and real pose, the mentioned functions enables the user to interact with the simulation, being able to change the sensors frequencies and the currently active sensors.

A scrolling plot of the errors is visualized using the homonym function: `scrolling plotter`.

Interface – Pose Viewer with Switches



Interface – Scrolling Plotter



main_2

This second version of the main script is used to visualize how the drone and the navigation system behave in a partially structured environment.

Some interesting conditions are studied and the navigation system response is analyzed, evaluating its performances.

main_2 - Legend

The most important plot is a bottom view of the navigation environment:

- the real trajectory is plotted with a blue, dashed, thin line
- the estimated trajectory is plotted dynamically with a continuous black line (`animatedline` MATLAB function)
- the walls of the building are represented with thick black lines

- ArUco markers are represented with a square
- known markers are represented in dark blue if outside the camera vision range and in green if inside
- unknown markers are represented in red if outside the camera vision range and in purple if inside it
- the estimate of the unknown marker position is represented in orange

main_2 - Legend

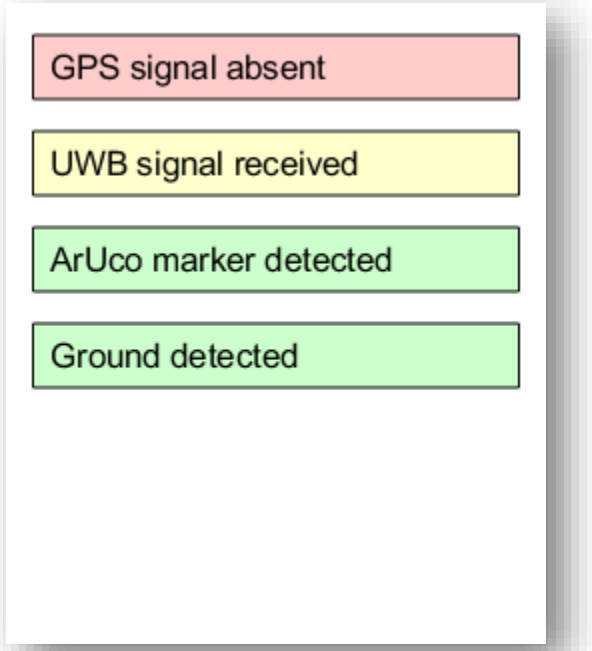
The most important plot is a bottom view of the navigation environment:

- UWB beacons are represented with a circle
- known beacons are represented in dark blue if they are obstructed by walls, in light blue if not obstructed but too far from the drone, and in green if their signal reaches the drone
- unknown beacons are represented in red if the drone does not sense them, and in purple if they reach the drone
- the estimate of the beacon position is represented in orange

main_2 - Interface

Elements:

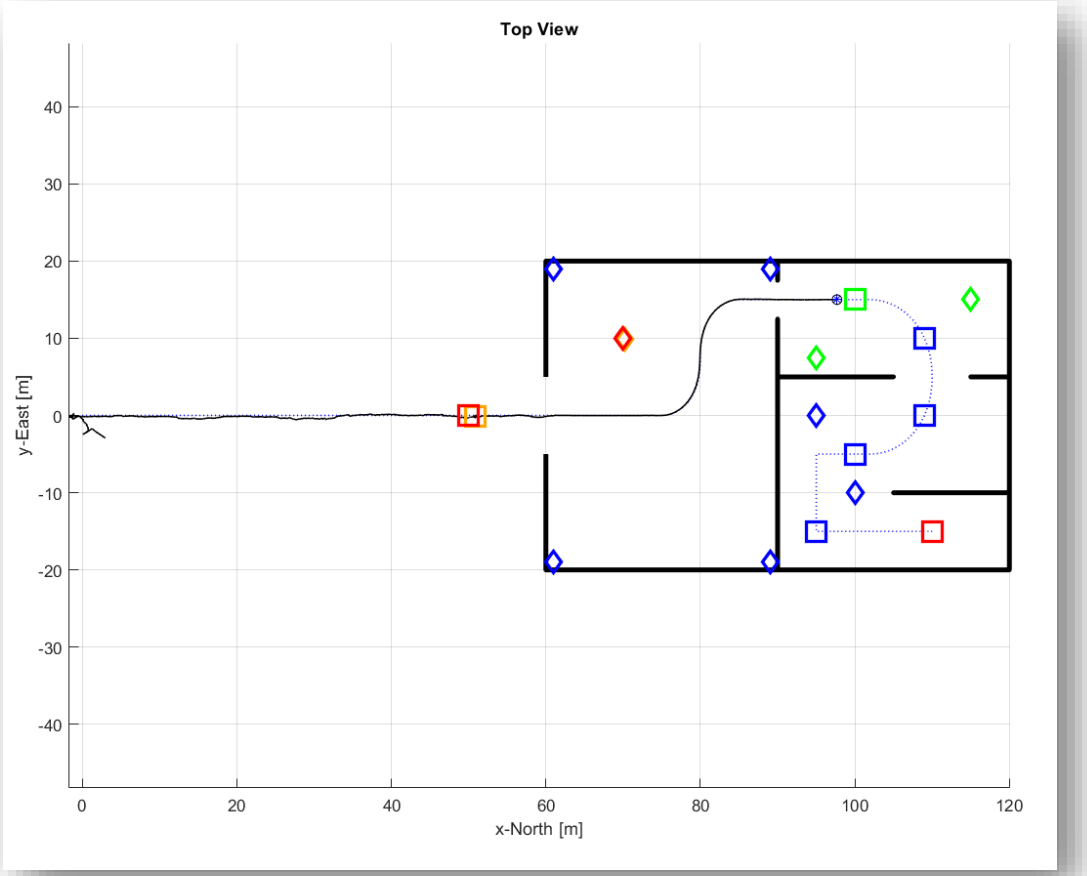
- Pose Viewer with Switches ✓
- Scrolling Plotter ✓
- **Available sensor measurements**
- Top View
- All internal states plot



main_2 - Interface

Elements:

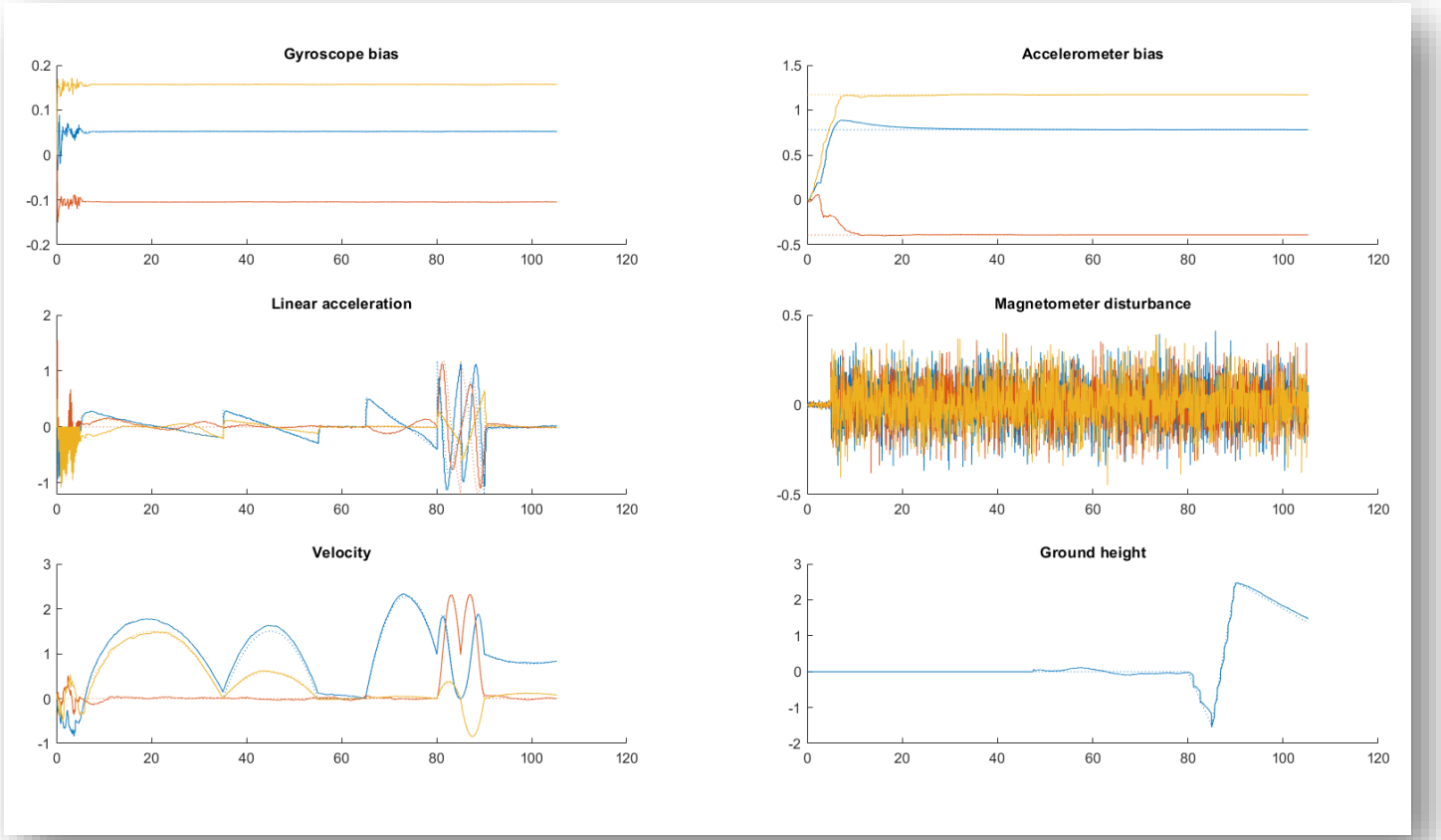
- Pose Viewer with Switches ✓
- Scrolling Plotter ✓
- Available sensor measurements
- **Top View**
- All internal states plot



main_2 - Interface

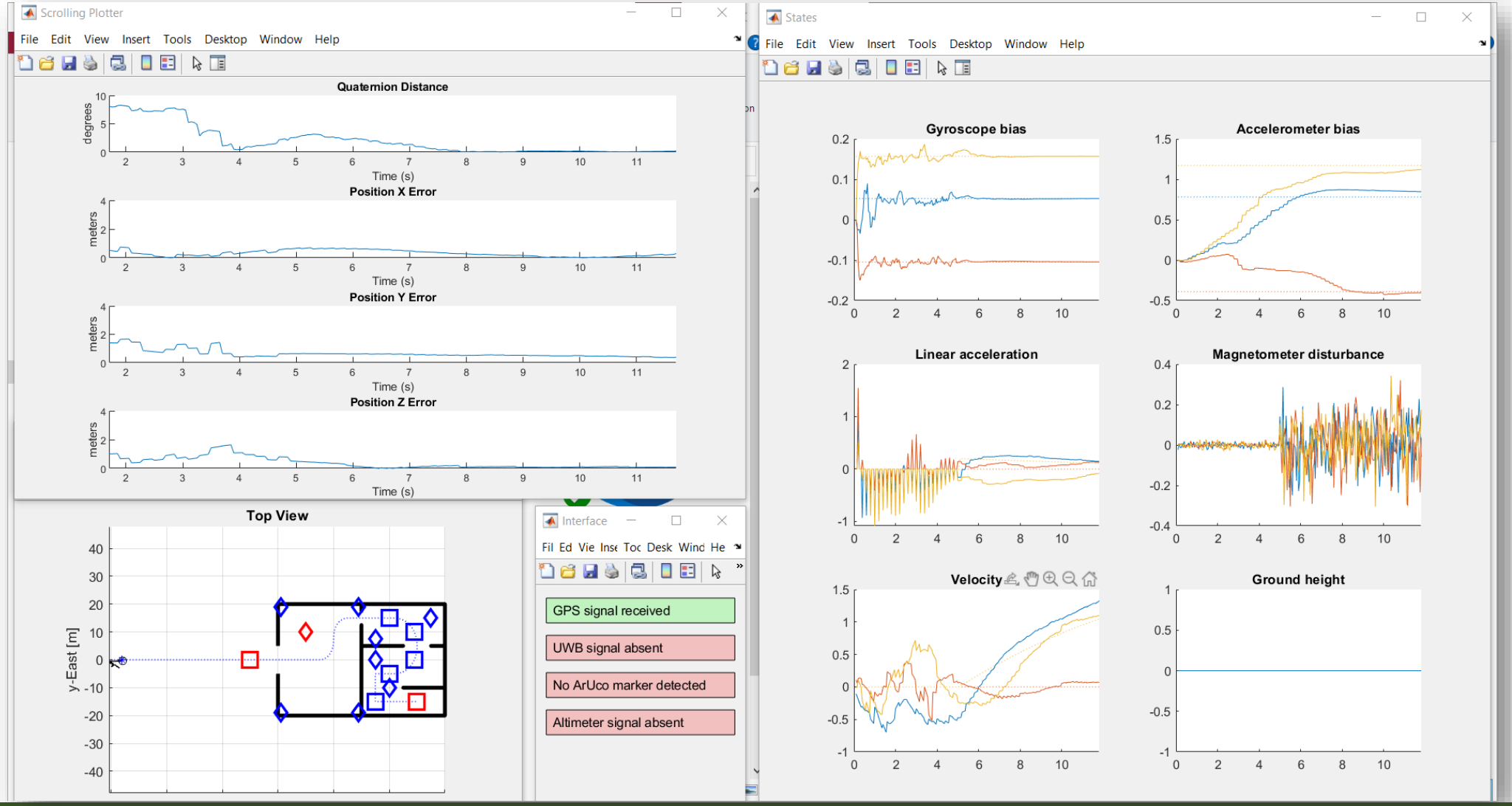
Elements:

- Pose Viewer with Switches ✓
- Scrolling Plotter ✓
- Available sensor measurements
- Top View
- **All internal states plot**



For each plot the three colors (blue, orange and yellow) represents the three components: x, y and z.

main_2 - Interface



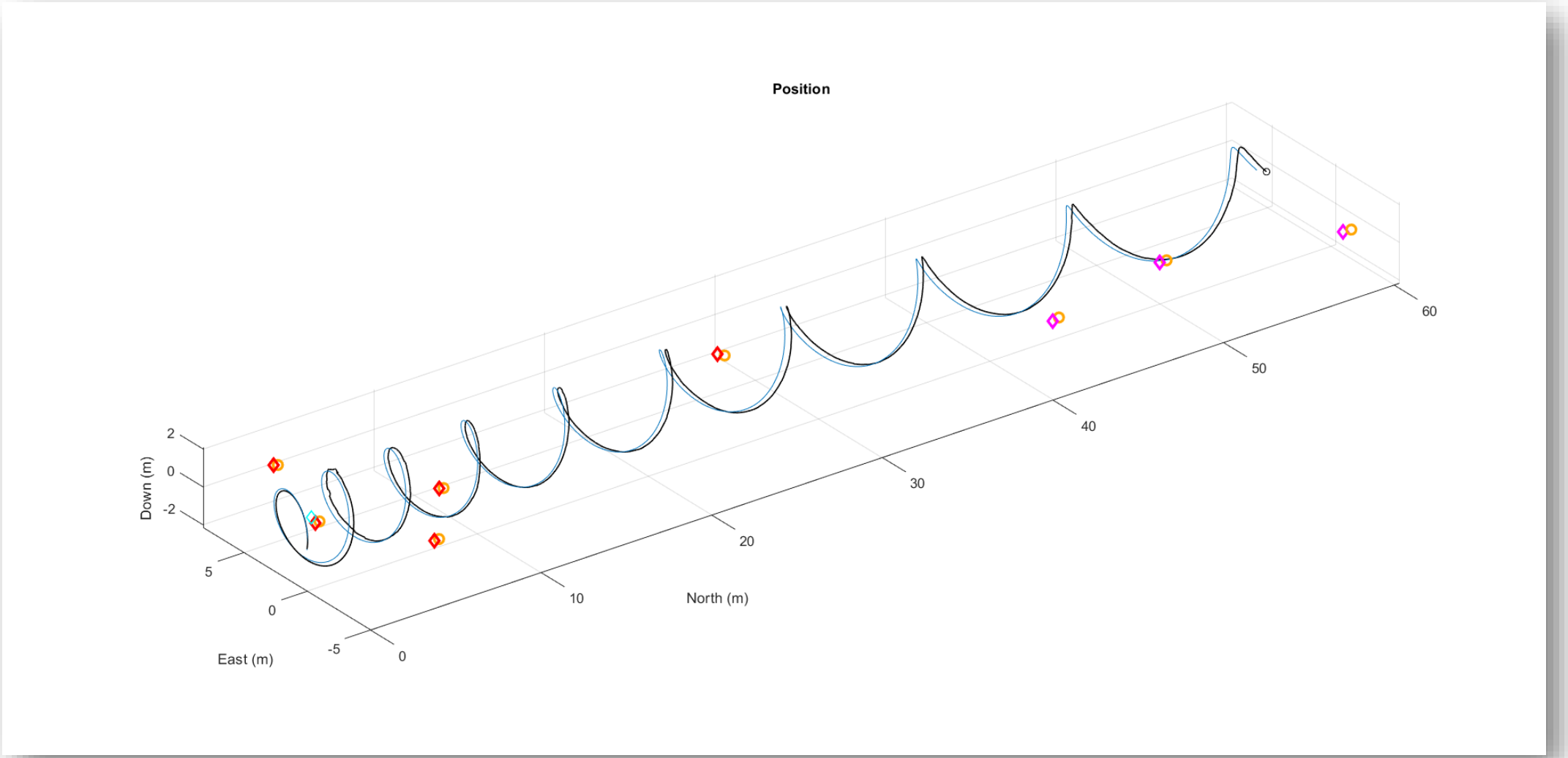
main_3

This version of the main script has been created in order to show the capabilities of the navigation using unknown UWB anchors.

An ad-hoc script has been created because a non-“singular” trajectory is necessary in order to use this algorithm capabilities.

A 3D interactive (can be rotated) representation of the trajectory and of the estimated trajectory is plotted in real time, showing the improvements compared to a simple dead-reckoning estimation.

main_3



my_navigation_filter

System object with the following (public) methods:

- `set_initial_state`
- `predict`
- `fuse_accel`
- `fuse_magn`
- `fuse_gps`
- `fuse_uwb`
- `fuse_unknown_uwb`
- `detect_unknown_uwb`
- `fuse_vis`
- `fuse_bar`
- `fuse_alt`
- `private_properties_update`

my_navigation_filter

To reduce the computational burden, considering the high output rate of the imu sensors and the slow dynamics of the drone, a **decimation factor** has been introduced.

It represents the number of accelerometers measurements which are aggregated in a single measurement and used for the correction step.

This guarantees a reduced computational burden (performing a Kalman filter step at the frequency of the accelerometers measurement rate is expensive) and there is no information loss.

This process is done only for the accelerometer since it is the sensor with the highest sample rate. The effect on performance may even be positive for some values of the decimation factor, since this process averages the measurements reducing the noise.

my_navigation_filter

Decimation Factor

The accelerometer measurement used for the correction step of the orientation is obtained as follows:

$$acc_{\text{decimated},k} = \frac{1}{d} \cdot \sum_{i=0}^{\text{d}-1} \text{Rot}(k,i) \cdot acc_{\text{meas},\,k-i}$$

Where $\text{Rot}(k,i)$ represents the rotation matrix from the corrected orientation at the time-step $(k-i)$ to the predicted orientation at the time-step (k).

The practical iterative implementation used is the following (used to reduce memory requirements):

$$acc_{\text{decimated}} = R\left(\left(\boldsymbol{\omega}_{\text{meas},k} - \boldsymbol{\omega}_{\text{bias},k}\right) \cdot \Delta t\right) \cdot acc_{\text{decimated}} + \frac{1}{d} \cdot acc_{\text{meas},\,k}$$

This assumes that d is constant. After the measurement is used in a correction step, it is reset to 0.

my_navigation_filter

Filter Tuning

To improve the performance of the filter, two different parameters set have been used. A first set of parameters is used for the initialization phase of the filter (the initialization time has been set to be 10 seconds and the drone is supposed to stay still during this time), the second set is used for the rest of the functioning time.

The set of parameters used during the initialization phase has been tuned starting from the other set and increasing the noise on the bias terms while reducing the noise on the disturbances terms (the linear acceleration and the magnetometer disturbance). This simple trick greatly speeds up the initialization of some filter states without a loss of generality.

Simulations

Simulations: Scenario 1

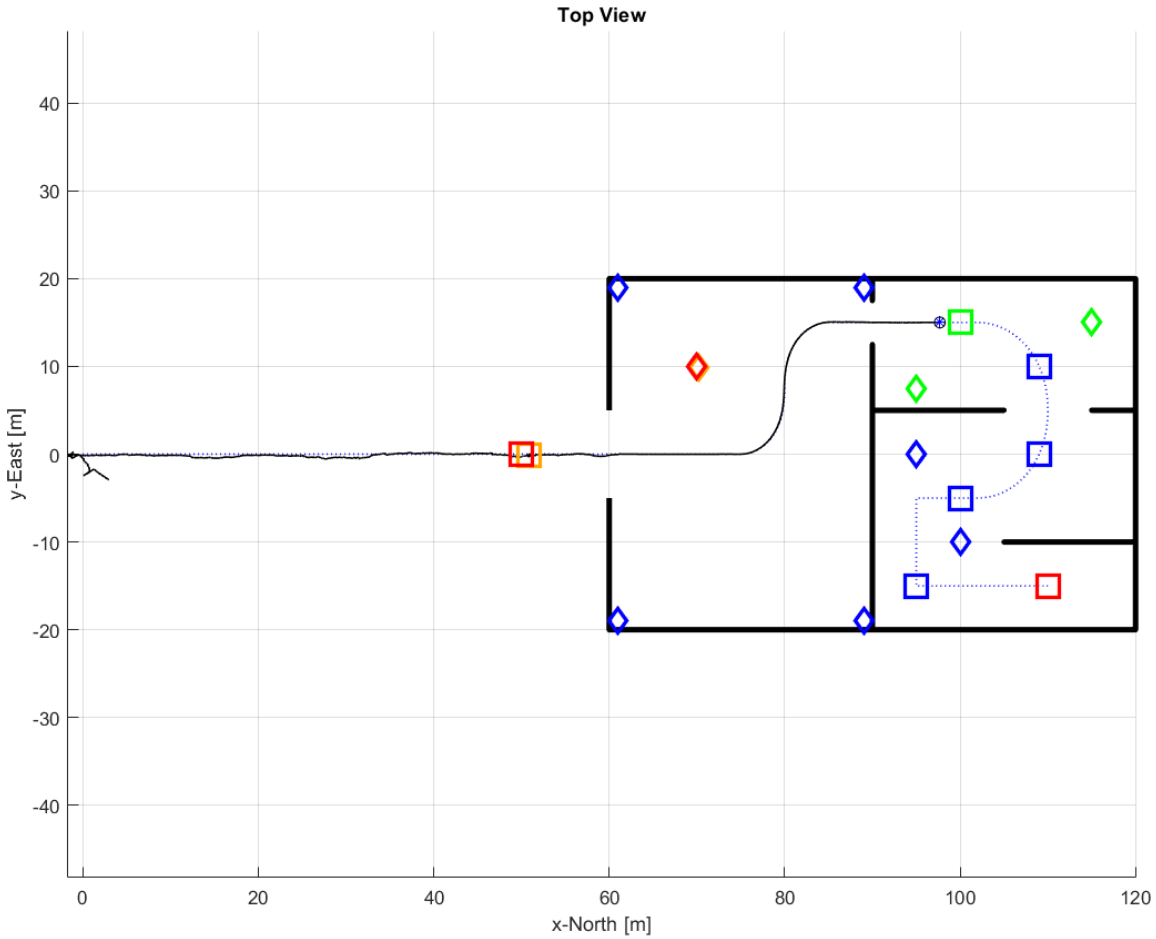
Main Simulated Scenario

All the sensors are simulated: accelerometer, magnetometer, gyroscope, GPS, UWB, vision sensor, barometer and altimeter.

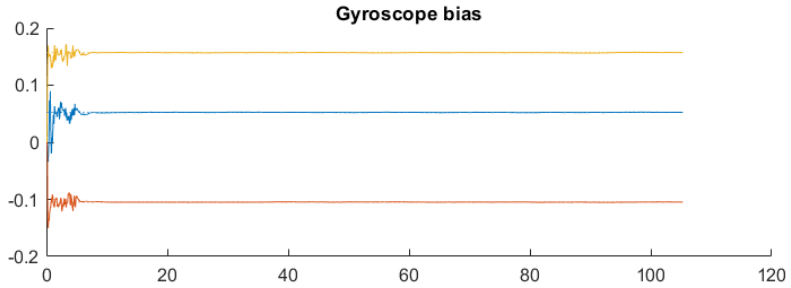
It is observed how they interact and how the filter behaves at the interface between different “zones» (when the GPS shuts down and the UWB signal starts to be received).

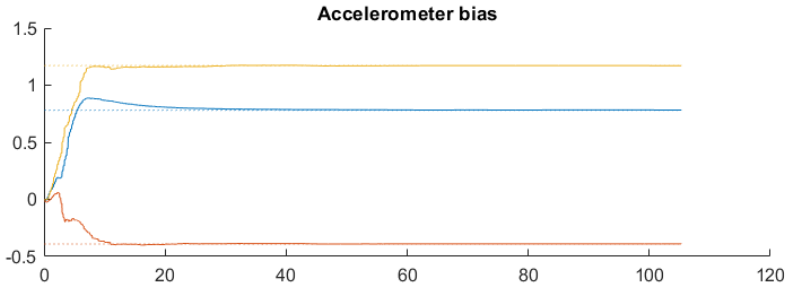
It is also observed the behaviour of the ground height estimation, and the detection and localization of unknown UWB beacons and ArUco markers.

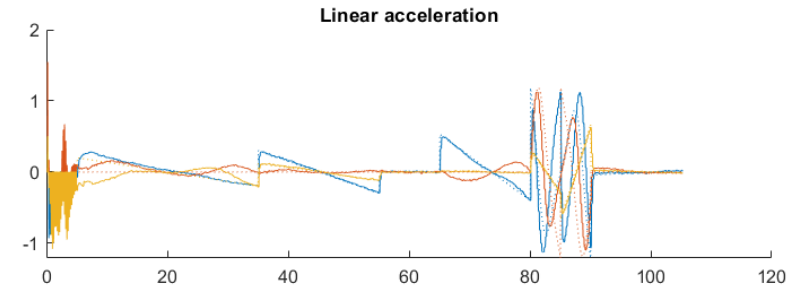
Simulations: Scenario 1

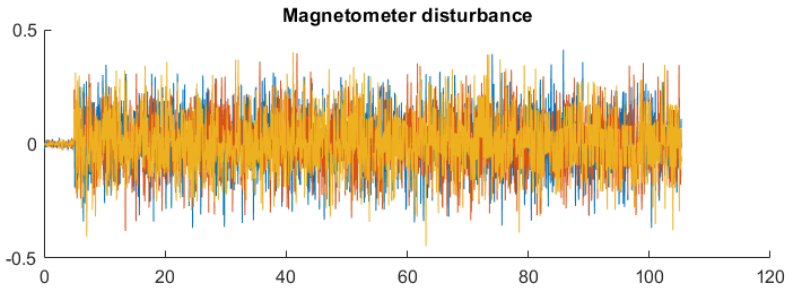


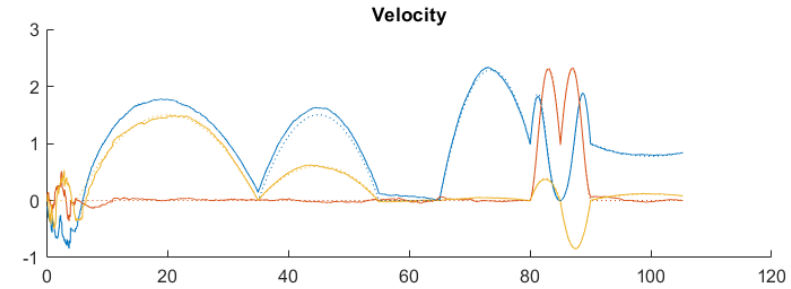
Simulations: Scenario 1

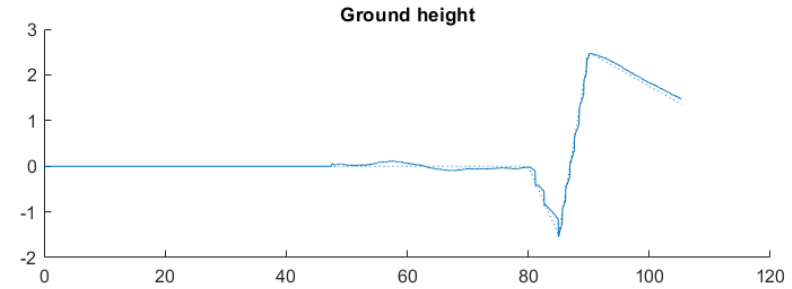




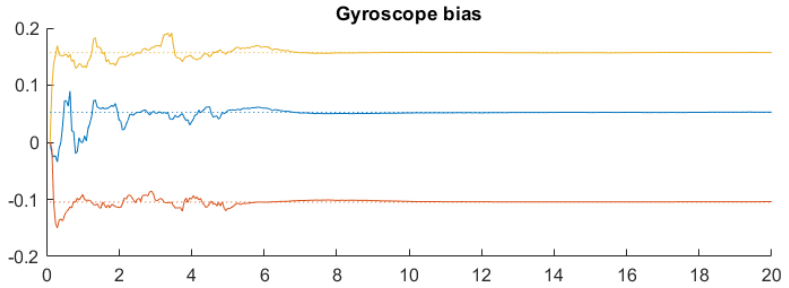


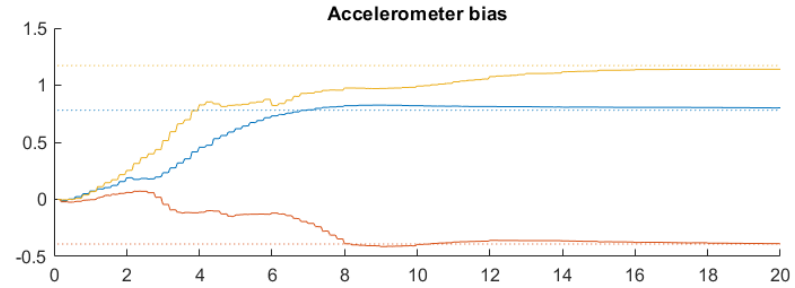


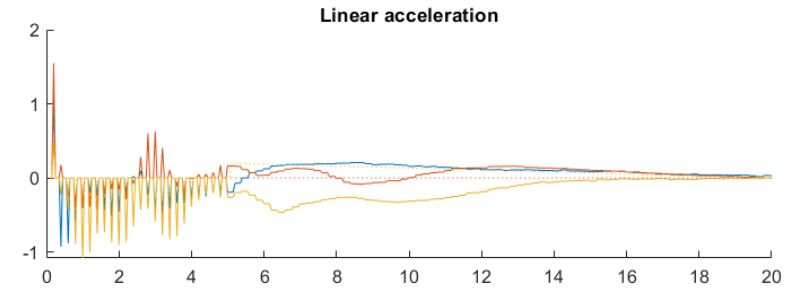


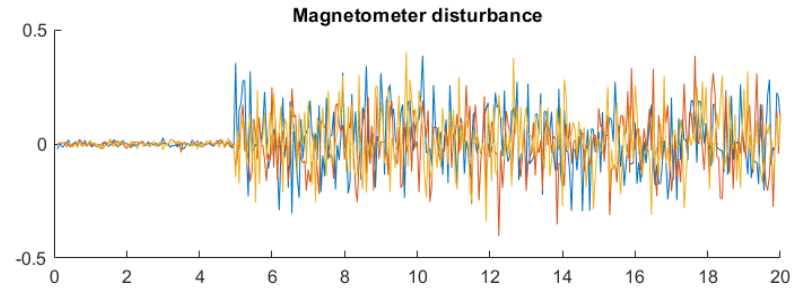


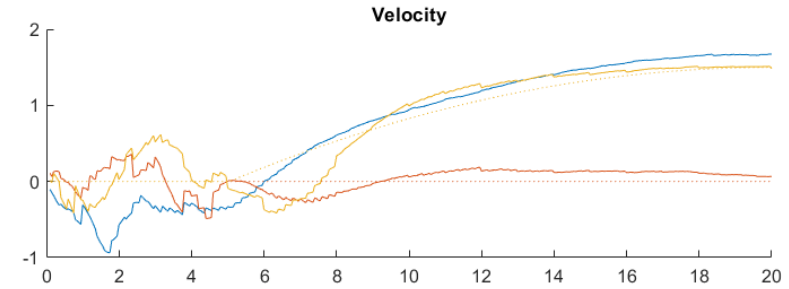
Simulations: Scenario 1

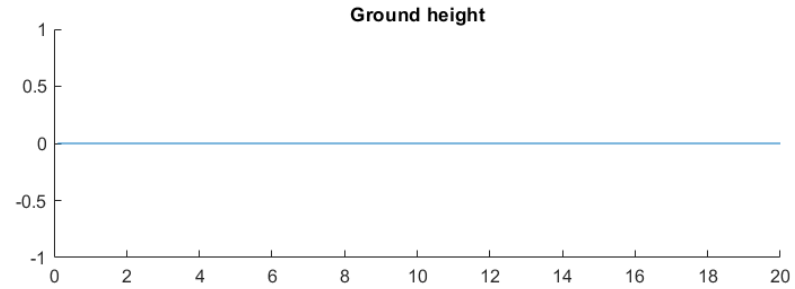




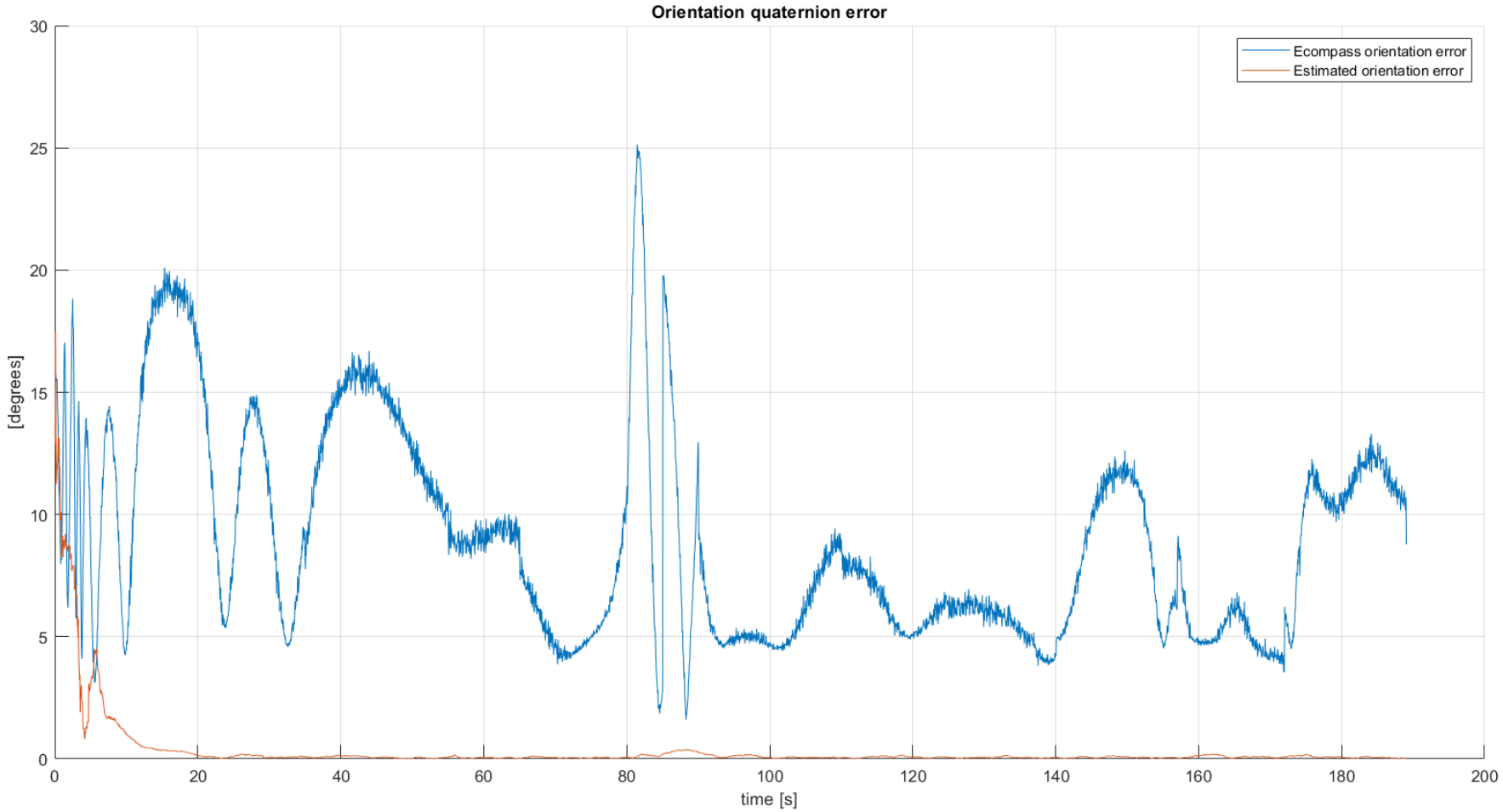




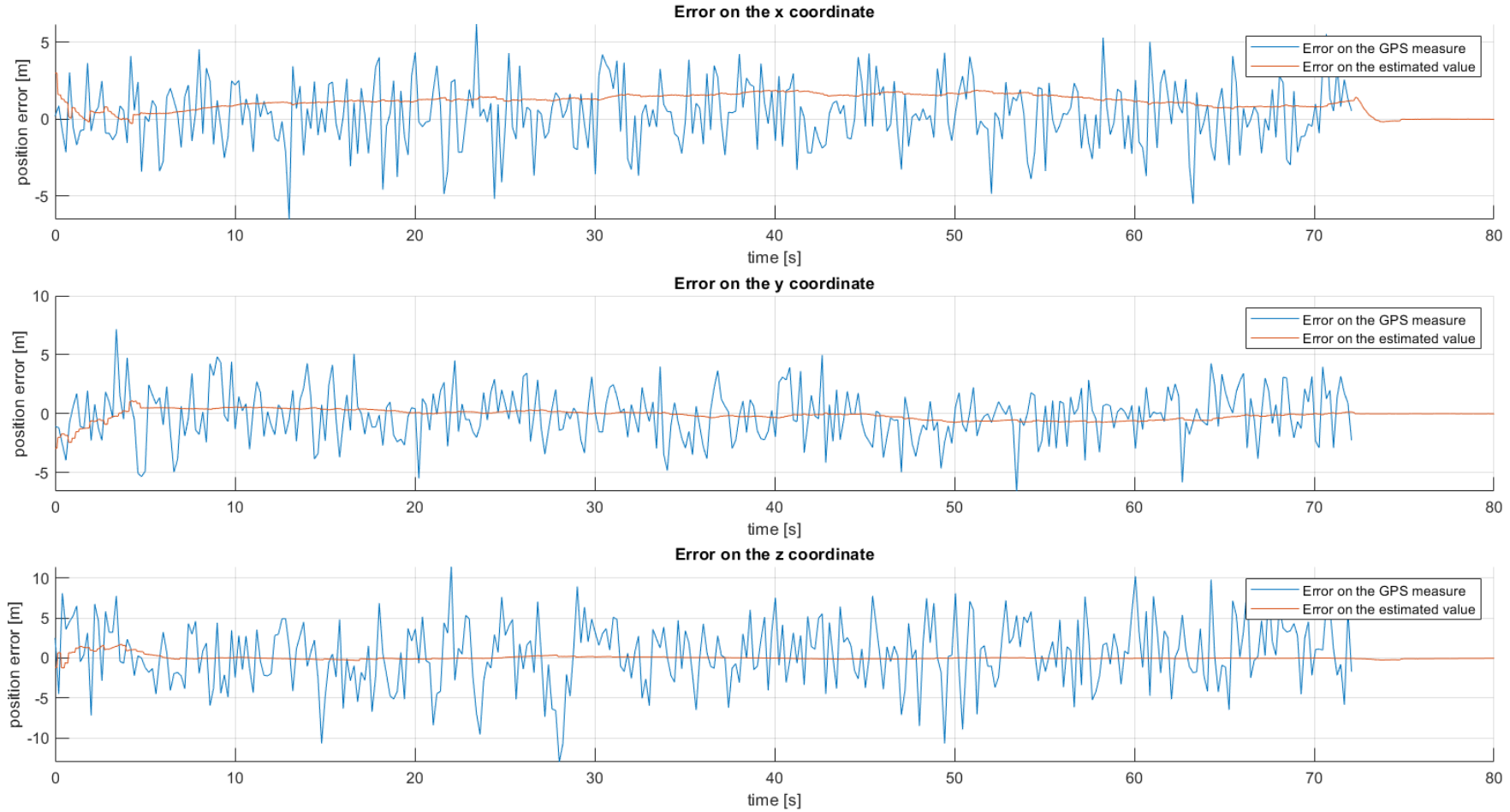




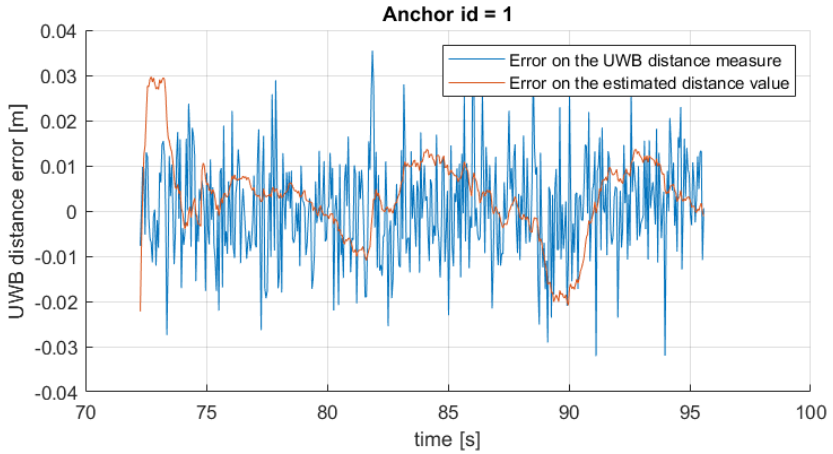
Simulations: Scenario 1

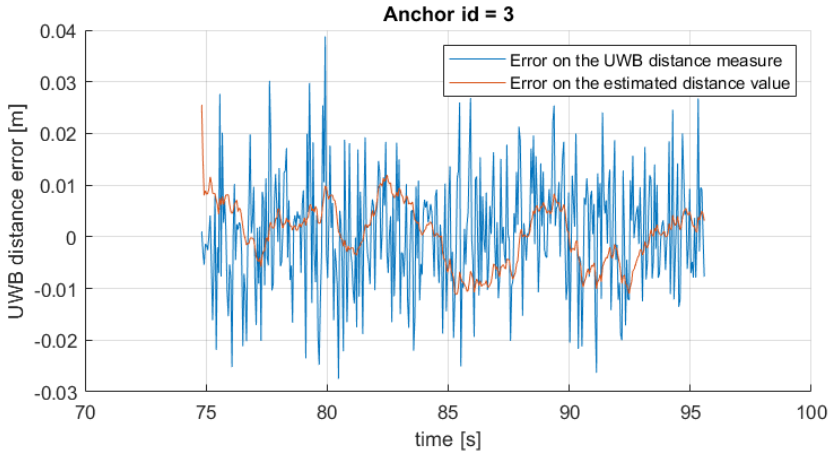


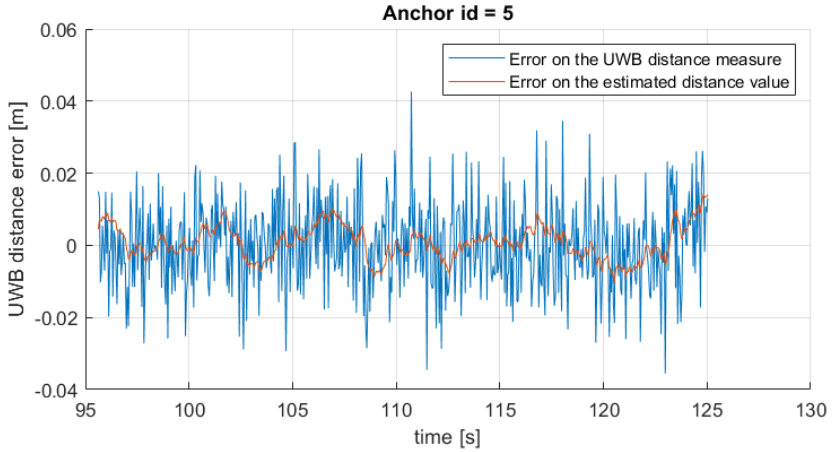
Simulations: Scenario 1

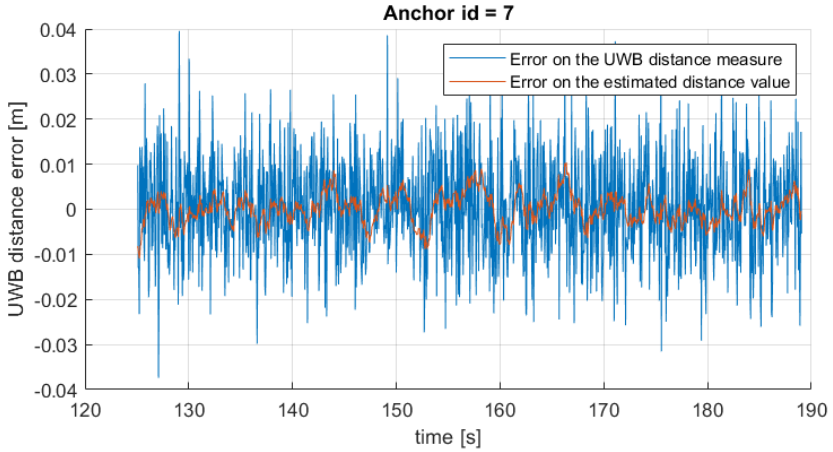


Simulations: Scenario 1

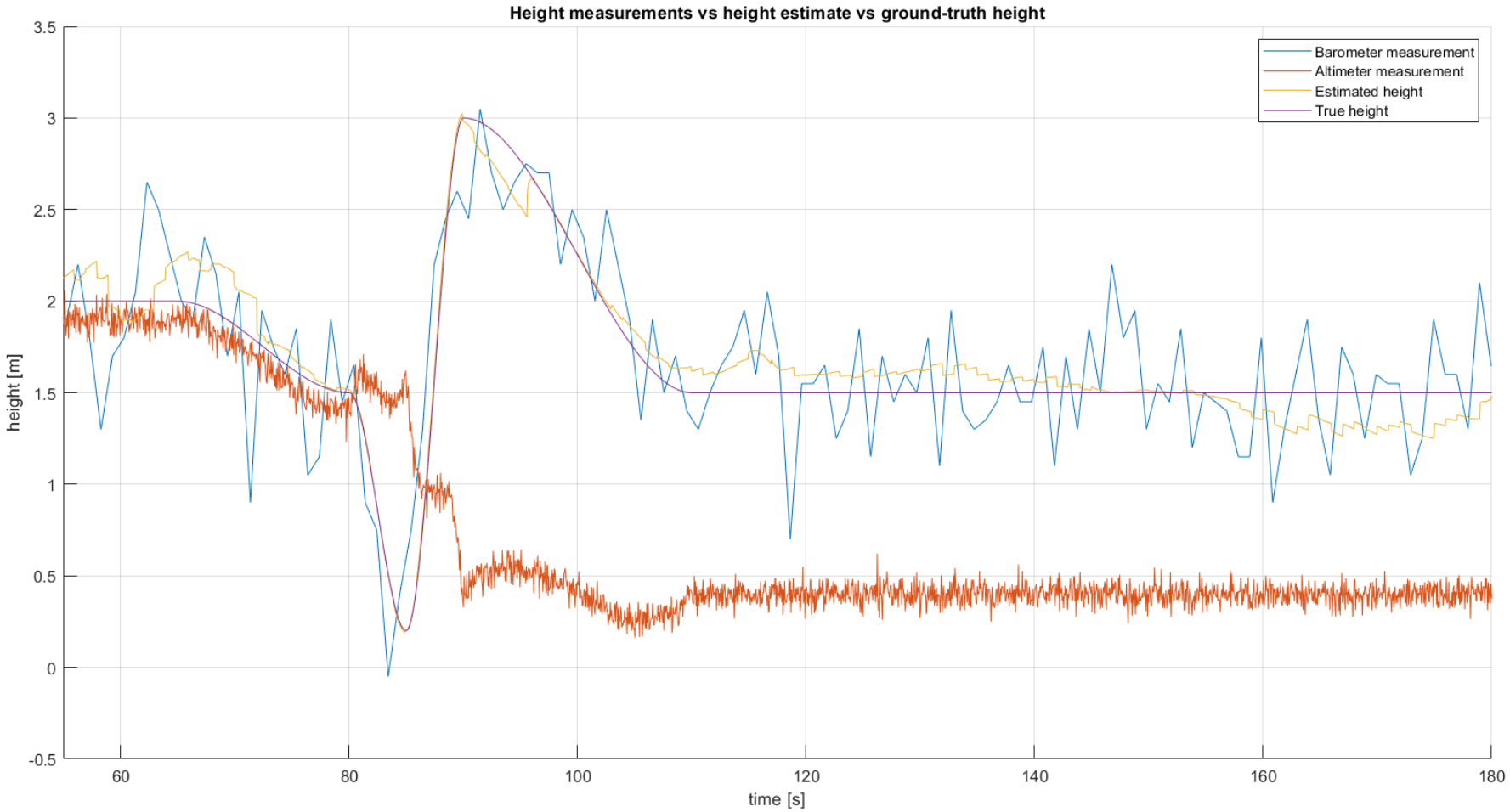








Simulations: Scenario 1



MATLAB R2020a - academic use

HOME PLOTS APPS EDITOR PUBLISH VIEW

Search Documentation

Davide

New Open Save Find Files Compare Print — FILE

Go To Find — NAVIGATE

Insert Comment Indent — EDIT

Breakpoints — BREAKPOINTS

Run Run and Advance Run Section Advance Run and Time — RUN

C: ▸ Users ▸ david ▸ Dropbox ▸ Uni ▸ Progetto Sistemi di Guida e Navigazione ▸

Current Folder

Name

- Backup
- custom_functions
- datasets
- Slideshow
- Theory
- main_1.m
- main_2.m
- main_3.m
- temp.m
- trajectory_data.mat

Slideshow (Folder)

Workspace

Name | Value

Editor - C:\Users\david\Dropbox\Uni\Progetto Sistemi di Guida e Navigazione\main_2.m

main_2.m

```
1   %% Description
2   %
3   % Generate a waypoint trajectory in a semi-structured environment (manually
4   % created) and visualize the estimate of the position of the drone.
5
6
7   %% Initialization
8
9   clc
10  clear variables
11  close all
12
13  rng('shuffle')
14
15  fprintf('Started. \n \n')
16
17  % Add the folder and the sub-folders to MATLAB path
18  addpath(genpath('custom_functions'))
19  addpath(genpath('datasets'))
20
21
```

Command Window

fx >>

UTF-8 script Ln 1 Col 1

Type here to search

20°C Mostly clear ITA 23:43 01/07/2021

Simulations: Scenario 2

Secondary Simulated Scenario

No particual condition is studied in this scenario.

It only serves as an additional benchmark.

In the late parts of the trajectory, only two UWB beacons are visible simultaneously. They are thus unable to localize the drone autonomously. Nonetheless, the filter estimation accuracy is still very good.

MATLAB R2020a - academic use

HOME PLOTS APPS EDITOR PUBLISH VIEW

Search Documentation

Davide

New Open Save Find Files Compare Print — FILE

Go To Find — NAVIGATE

Insert Comment Indent — EDIT

Breakpoints — BREAKPOINTS

Run Run and Advance Run Section Advance Run and Time — RUN

C: ▸ Users ▸ david ▸ Dropbox ▸ Uni ▸ Progetto Sistemi di Guida e Navigazione ▸

Current Folder

Name

- Backup
- custom_functions
- datasets
- Slideshow
- Theory
- main_1.m
- main_2.m
- main_3.m
- temp.m
- trajectory_data.mat

Slideshow (Folder)

Workspace

Name | Value

Editor - C:\Users\david\Dropbox\Uni\Progetto Sistemi di Guida e Navigazione\main_2.m

main_2.m

```
1   %% Description
2   %
3   % Generate a waypoint trajectory in a semi-structured environment (manually
4   % created) and visualize the estimate of the position of the drone.
5
6
7   %% Initialization
8
9   clc
10  clear variables
11  close all
12
13  rng('shuffle')
14
15  fprintf('Started. \n \n')
16
17  % Add the folder and the sub-folders to MATLAB path
18  addpath(genpath('custom_functions'))
19  addpath(genpath('datasets'))
20
21
```

Command Window

fx >>

UTF-8 script Ln 1 Col 1

Type here to search

20°C Clear ITA 00:10 02/07/2021

Simulations: Scenario 3

Magnetometer Disturbance Simulatiton

It is studied how the system reacts to significative magnetometer disturbances.

MATLAB R2020a - academic use

HOME PLOTS APPS EDITOR PUBLISH VIEW

Search Documentation

Davide

New Open Save Find Files Compare Print — FILE

Go To Find — NAVIGATE

Insert Comment Indent — EDIT

Breakpoints — BREAKPOINTS

Run Run and Advance Run Section Advance Run and Time — RUN

Run main_2 (F5)

C: ▸ Users ▸ david ▸ Dropbox ▸ Uni ▸ Progetto Sistemi di Guida e Navigazione ▸

Current Folder

Name

- Backup
- custom_functions
- datasets
- Slideshow
- Theory
- main_1.m
- main_2.m
- main_3.m
- temp.m
- trajectory_data.mat

Slideshow (Folder)

Workspace

Name	Value
accel	[0.7598,-(
accessible_uw...	1x1 Scatte
alt	1x1 altim
alt_h_meas	0.3950
alt_imu	3781x1 d
alt_meas	3781x1 d
altFs	20
altimeter_pos...	[0,0,0.100
anim_est_states	1x16 Anir

Editor - C:\Users\david\Dropbox\Uni\Progetto Sistemi di Guida e Navigazione\main_2.m

main_2.m

```
1   %% Description
2   %
3   % Generate a waypoint trajectory in a semi-structured environment (manually
4   % created) and visualize the estimate of the position of the drone.
5
6
7   %% Initialization
8
9   clc
10  clear variables
11  close all
12
13  rng('shuffle')
14
15  fprintf('Started. \n \n')
16
17  % Add the folder and the sub-folders to MATLAB path
18  addpath(genpath('custom_functions'))
19  addpath(genpath('datasets'))
20
21
```

Command Window

```
End-to-End Simulation Position RMS Error
    X: 0.72 , Y: 0.41, Z: 0.12   (meters)

End-to-End Quaternion Distance RMS Error (degrees)
    1.41 (degrees)

>>
```

Type here to search

20°C Clear ITA 00:19 02/07/2021

Simulations: Scenario 4

Faulty GPS Measurements

The Robust estimation algorithm is put to work.

The GPS measurements are contaminated by outliers.

The behaviour of the " standard" KF is compared to the behaviour of the adaptively-robust one.

MATLAB R2020a - academic use

HOME PLOTS APPS EDITOR PUBLISH VIEW

Search Documentation | Davide

New Open Save | Find Files Compare Print | Go To Find | Insert Comment Indent | Breakpoints | Run Run and Advance | Run Section Advance | Run and Time

FILE NAVIGATE EDIT BREAKPOINTS RUN

C: ▸ Users ▸ david ▸ Dropbox ▸ Uni ▸ Progetto Sistemi di Guida e Navigazione ▸

Current Folder

Name

- Backup
- custom_functions
- datasets
- Slideshow
- Theory
- main_1.m
- main_2.m
- main_3.m
- temp.m
- trajectory_data.mat

Slideshow (Folder)

Workspace

Name	Value
accel	[0.7963,-(
accessible_uw...	1x1 Scatte
alt	1x1 altim
alt_h_meas	0.5090
alt_imu	3781x1 d
alt_meas	3781x1 d
altFs	20
altimeter_pos...	[0,0,0.100
anim_est_states	1x16 Anir

Editor - C:\Users\david\Dropbox\Uni\Progetto Sistemi di Guida e Navigazione\main_2.m

main_2.m

```
1   %% Description
2   %
3   % Generate a waypoint trajectory in a semi-structured environment (manually
4   % created) and visualize the estimate of the position of the drone.
5
6
7   %% Initialization
8
9   clc
10  clear variables
11  close all
12
13  rng('shuffle')
14
15  fprintf('Started. \n \n')
16
17  % Add the folder and the sub-folders to MATLAB path
18  addpath(genpath('custom_functions'))
19  addpath(genpath('datasets'))
20
21
```

Command Window

```
End-to-End Simulation Position RMS Error
    X: 0.65 , Y: 0.25, Z: 0.19   (meters)

End-to-End Quaternion Distance RMS Error (degrees)
    1.77 (degrees)

>>
```

UTF-8 | script | Ln 1 | Col 1

Type here to search

19°C Clear ITA 00:27 02/07/2021

MATLAB R2020a - academic use

HOME PLOTS APPS EDITOR PUBLISH VIEW

Search Documentation | Davide

New Open Save | Find Files Compare Print | Go To Find | Insert Comment Indent | Breakpoints | Run Run and Advance | Run Section Advance | Run and Time

FILE NAVIGATE EDIT BREAKPOINTS RUN

C: ▸ Users ▸ david ▸ Dropbox ▸ Uni ▸ Progetto Sistemi di Guida e Navigazione ▸

Current Folder

Name

- Backup
- custom_functions
- datasets
- Slideshow
- Theory
- main_1.m
- main_2.m*
- main_3.m
- temp.m
- trajectory_data.mat

Slideshow (Folder)

Workspace

Name	Value
accel	[0.8472,-0
accessible_uw...	1x1 Scatte
alt	1x1 altim
alt_h_meas	1.5440
alt_imu	3781x1 d
alt_meas	3781x1 d
altFs	20
altimeter_pos...	[0,0,0.100
anim_est_states	1x16 Anir

Editor - C:\Users\david\Dropbox\Uni\Progetto Sistemi di Guida e Navigazione\main_2.m

main_2.m

```
565
566     % Filter call and properties initialization
567     % FUSE = my_navigation_filter_adaptive;
568 -   FUSE = my_navigation_filter;
569
570 -   dt = Ts;
571
572     % Sample rates
573 -   FUSE.sample_rate = imuFs;
574
575     % Decimation factor
576 -   FUSE.decimation_factor = decimation_factor;
577
578     % Some process noises
579 -   FUSE.pos_noise = 2e-3;
580 -   FUSE.vel_noise = 1e-5;
581
582     % Accel, gyro, magn noise
583 -   FUSE.acc_noise = 1e+0*xxx;
584 -   FUSE.magn_noise = 1e+0*xxx;
585 -   FUSE.gyro_noise = 1e-3*xxx;
```

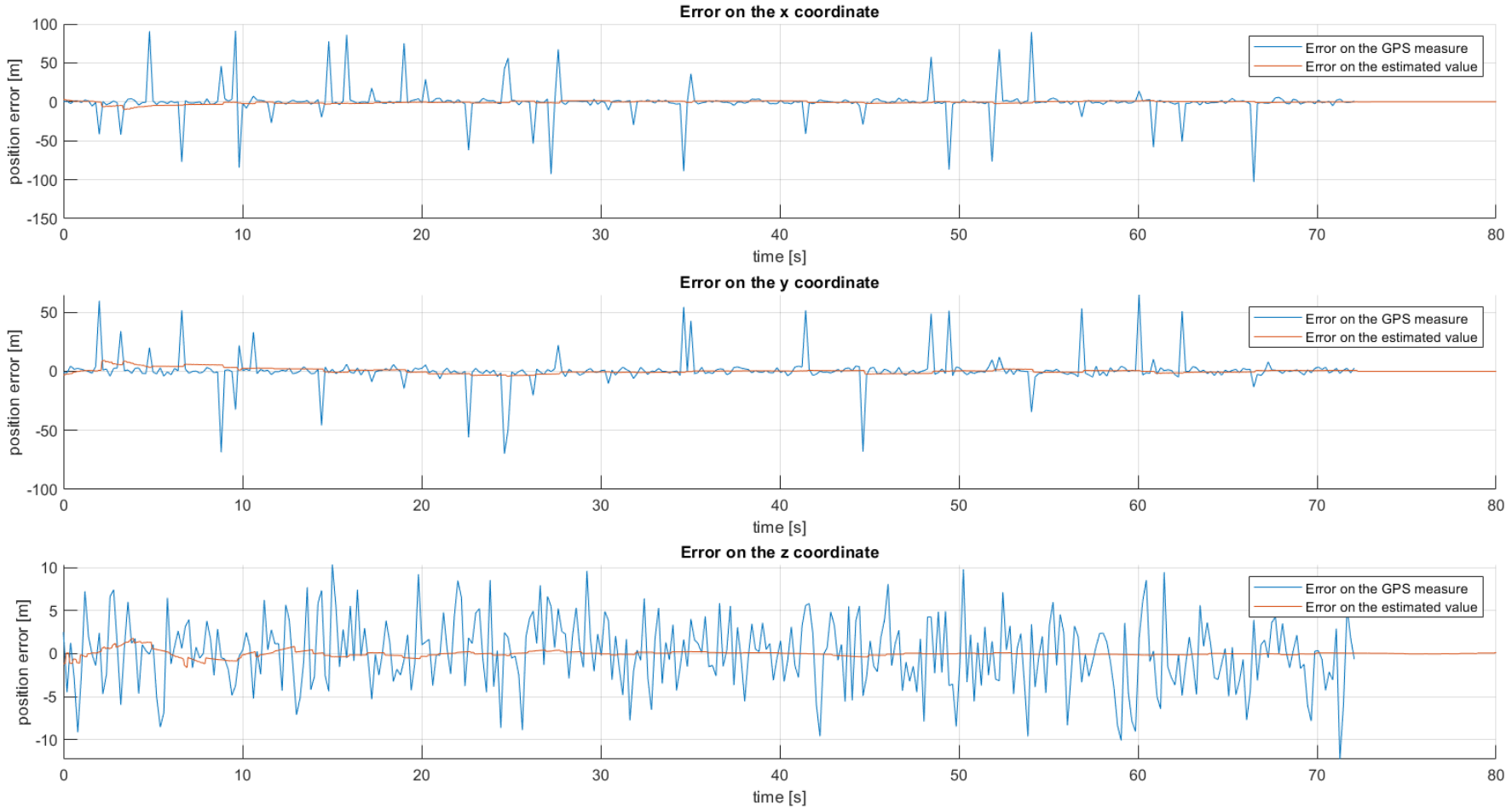
Command Window

>>

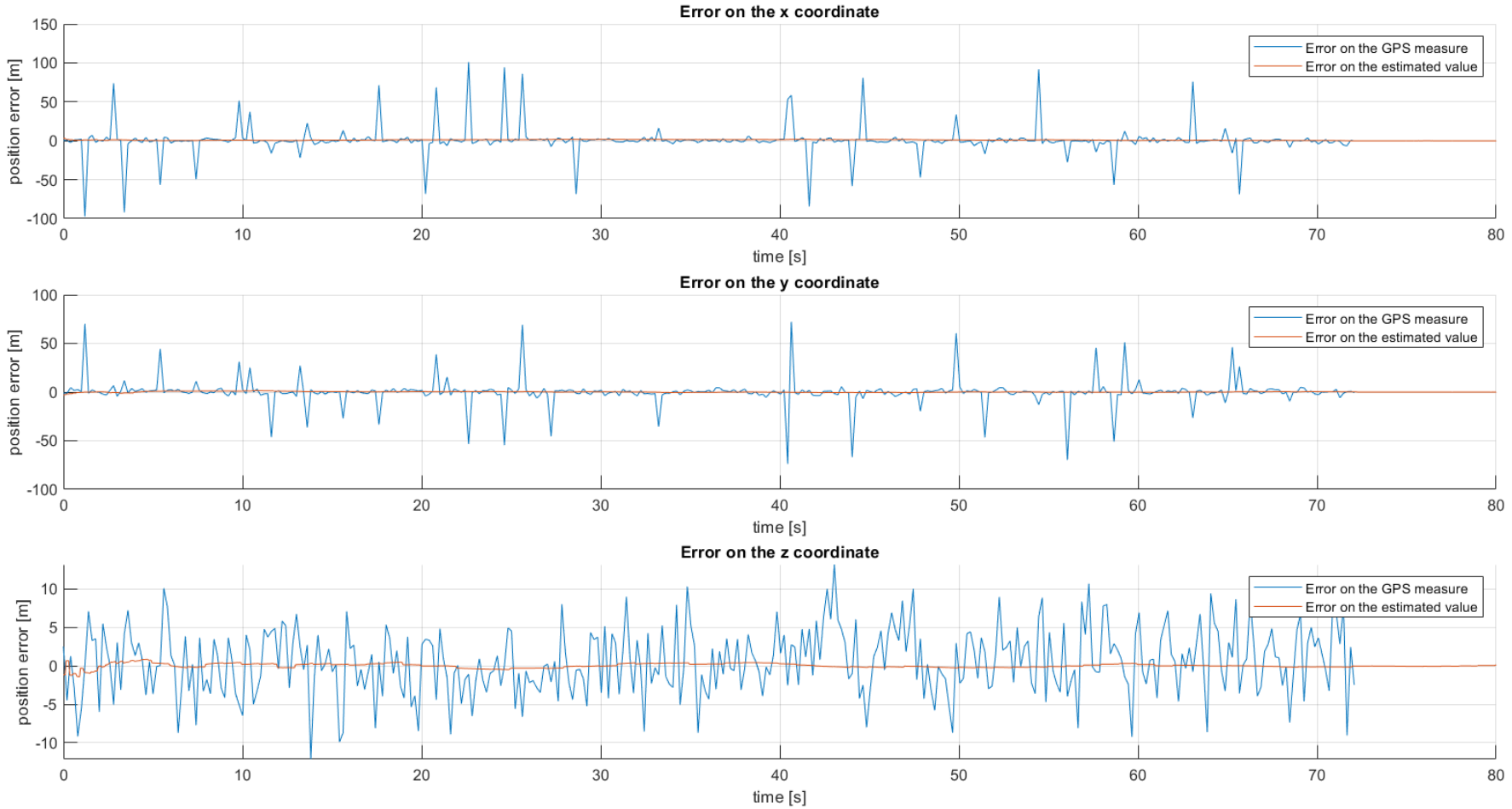
UTF-8 | script | Ln 568 Col 25

Type here to search | 19°C Clear | ITA | 00:31 02/07/2021

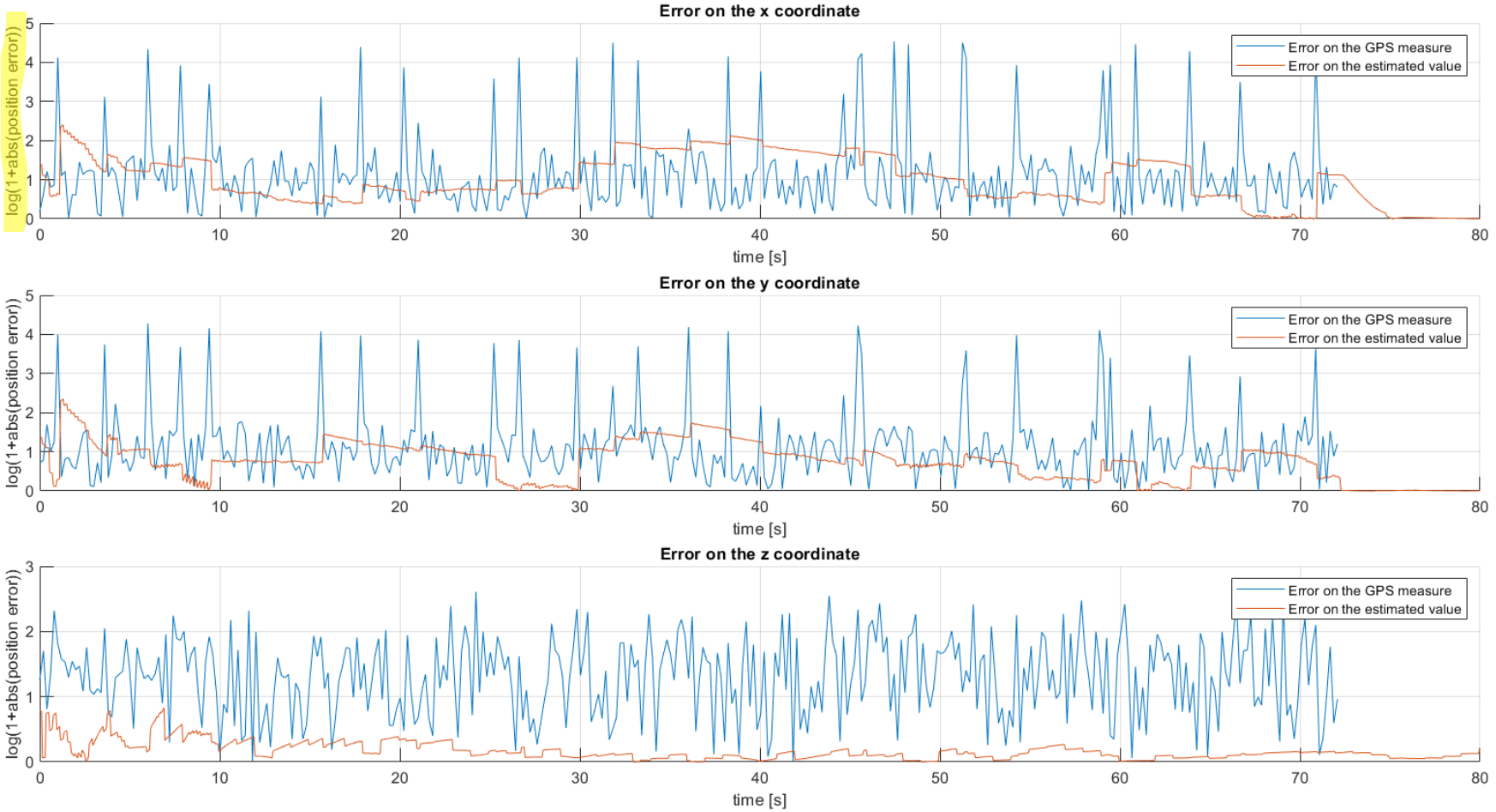
Simulations: Scenario 4 – Non Adaptive Filter



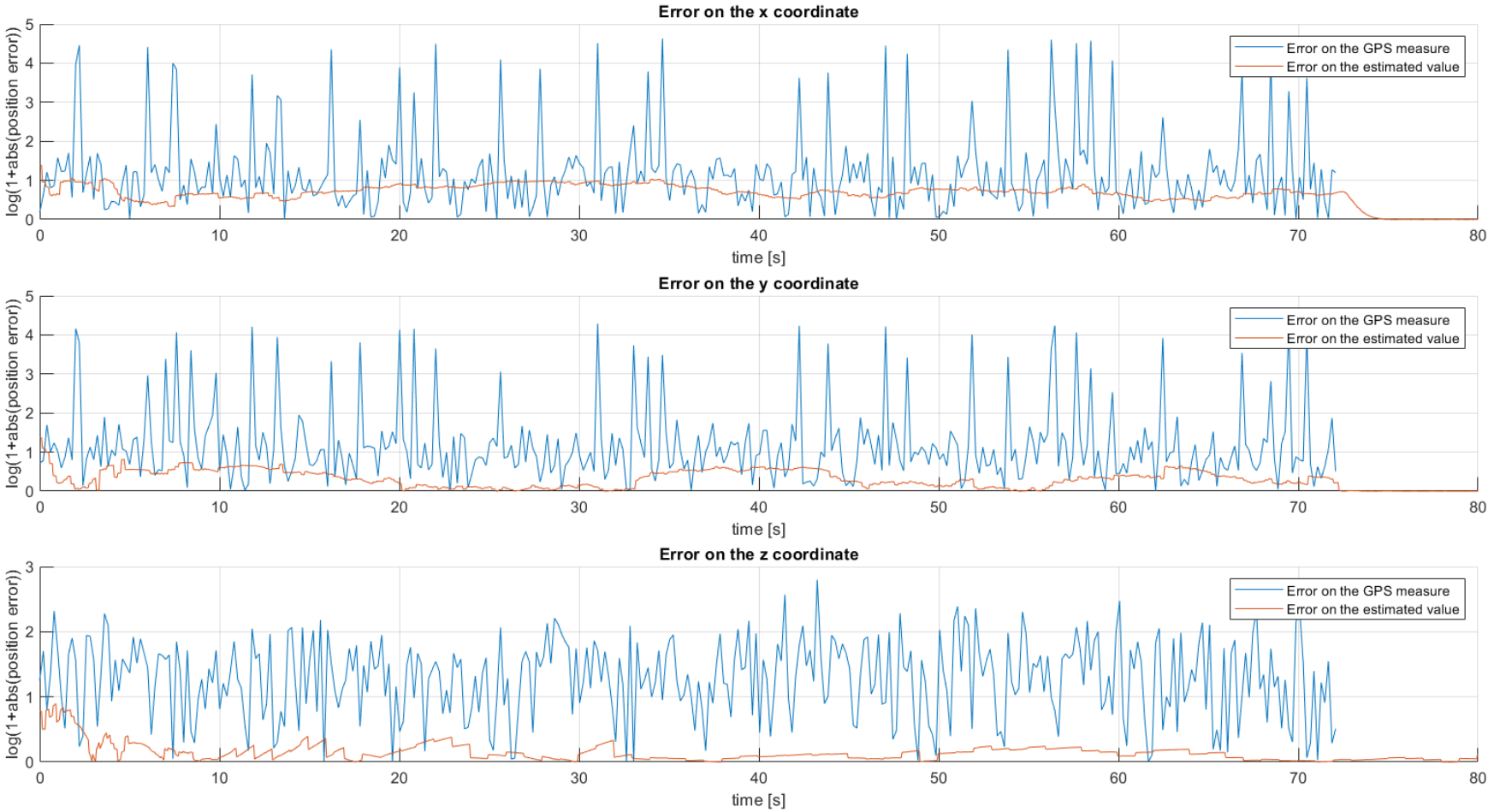
Simulations: Scenario 4 – Adaptive Filter



Simulations: Scenario 4 – Non Adaptive Filter



Simulations: Scenario 4 – Adaptive Filter



Simulations: Scenario 5

Faulty UWB Measurements

The Robust estimation algorithm is put to work, again :o

The GPS measurements are contaminated by NLOS errors.

The behaviour of the " standard" KF is compared to the behaviour of the adaptively-robust one.

MATLAB R2020a - academic use

HOME PLOTS APPS EDITOR PUBLISH VIEW

Search Documentation | Davide

New Open Save | Find Files Compare Print | Go To Find | Insert Comment Indent | Breakpoints | Run Run and Advance | Run Section Advance | Run and Time

FILE NAVIGATE EDIT BREAKPOINTS RUN

C: ▸ Users ▸ david ▸ Dropbox ▸ Uni ▸ Progetto Sistemi di Guida e Navigazione ▸

Current Folder

Name

- Backup
- custom_functions
- datasets
- Slideshow
- Theory
- main_1.m
- main_2.m
- main_3.m
- temp.m
- trajectory_data.mat

Slideshow (Folder)

Workspace

Name	Value
accel	[0.9382,-1
alt	1x1 altim
alt_h_meas	1.8790
alt_imu	3781x1 d
alt_meas	3781x1 d
altFs	20
altimeter_pos...	[0,0,0.100
anim_est_states	1x16 Anir
anim_real_sta...	1x16 Anir

Editor - C:\Users\david\Dropbox\Uni\Progetto Sistemi di Guida e Navigazione\main_2.m

main_2.m

```
565
566     % Filter call and properties initialization
567 -   FUSE = my_navigation_filter_adaptive;
568     % FUSE = my_navigation_filter;
569
570 -   dt = Ts;
571
572     % Sample rates
573 -   FUSE.sample_rate = imuFs;
574
575     % Decimation factor
576 -   FUSE.decimation_factor = decimation_factor;
577
578     % Some process noises
579 -   FUSE.pos_noise = 2e-3;
580 -   FUSE.vel_noise = 1e-5;
581
582     % Accel, gyro, magn noise
583 -   FUSE.acc_noise = 1e+0*xxx;
584 -   FUSE.magn_noise = 1e+0*xxx;
585 -   FUSE.gyro_noise = 1e-3*xxx;
```

Command Window

>>

UTF-8 | script | Ln 568 Col 31

Type here to search

19°C Clear ITA 00:36 02/07/2021

MATLAB R2020a - academic use

HOME PLOTS APPS EDITOR PUBLISH VIEW

Search Documentation | Davide

New Open Save | Find Files Compare Print | Go To Find | Insert Comment Indent | Breakpoints | Run Run and Advance | Run Section Advance | Run and Time

FILE NAVIGATE EDIT BREAKPOINTS RUN

C: ▸ Users ▸ david ▸ Dropbox ▸ Uni ▸ Progetto Sistemi di Guida e Navigazione ▸

Current Folder

Name

- Backup
- custom_functions
- datasets
- Slideshow
- Theory
- main_1.m
- main_2.m
- main_3.m
- temp.m
- trajectory_data.mat

Slideshow (Folder)

Workspace

Name	Value
accel	[0.7586,-(
accessible_uw...	1x1 Scatte
alt	1x1 altim
alt_h_meas	0.3400
alt_imu	3781x1 d
alt_meas	3781x1 d
altFs	20
altimeter_pos...	[0,0,0.100
anim_est_states	1x16 Anir

Editor - C:\Users\david\Dropbox\Uni\Progetto Sistemi di Guida e Navigazione\main_2.m

main_2.m

```
1   %% Description
2   %
3   % Generate a waypoint trajectory in a semi-structured environment (manually
4   % created) and visualize the estimate of the position of the drone.
5
6
7   %% Initialization
8
9   clc
10  clear variables
11  close all
12
13  rng('shuffle')
14
15  fprintf('Started. \n \n')
16
17  % Add the folder and the sub-folders to MATLAB path
18  addpath(genpath('custom_functions'))
19  addpath(genpath('datasets'))
20
21
```

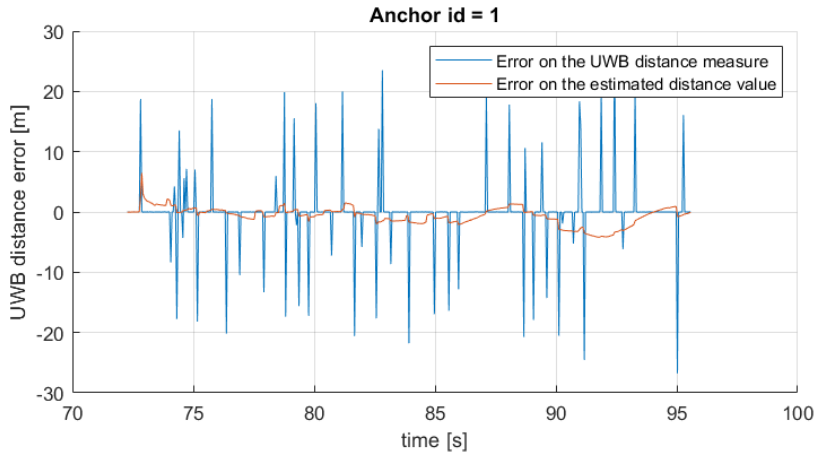
Command Window

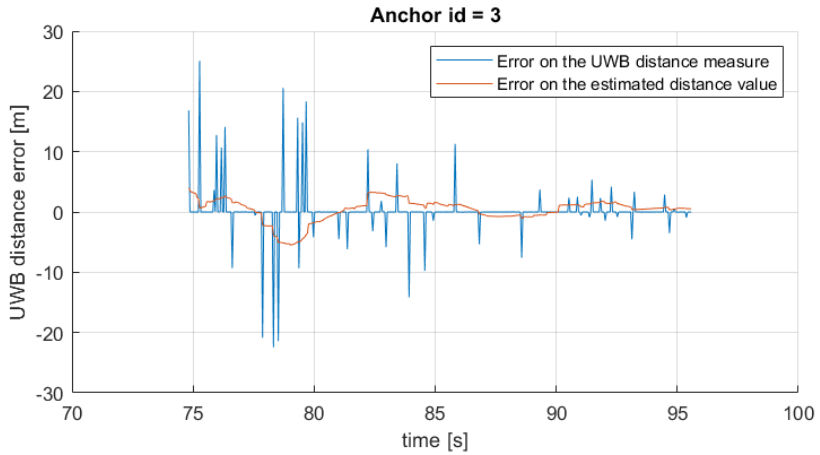
>>

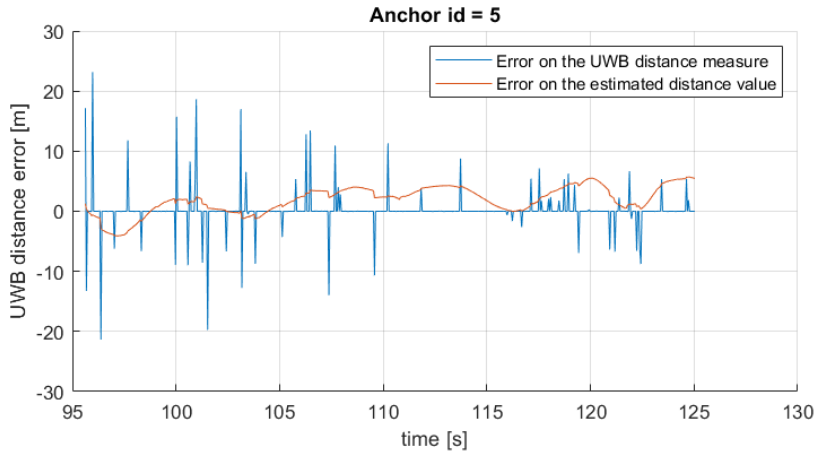
UTF-8 | script | Ln 4 | Col 44

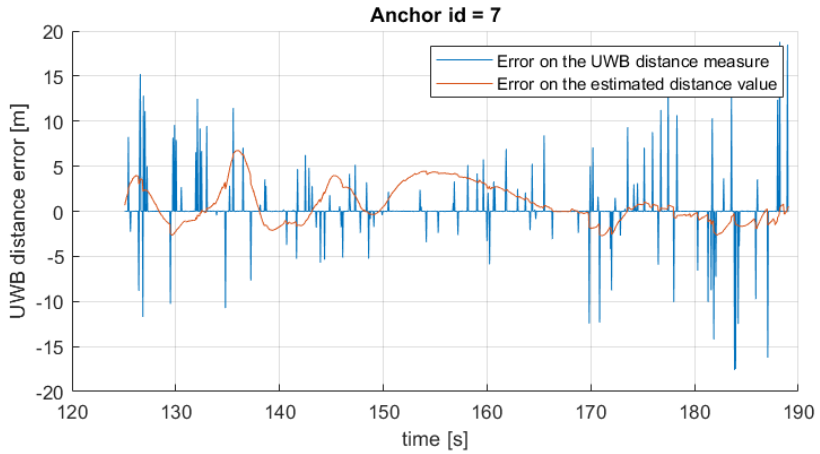
Type here to search | 19°C Clear | ITA | 01:08 | 02/07/2021

Simulations: Scenario 5 – Non Adaptive Filter

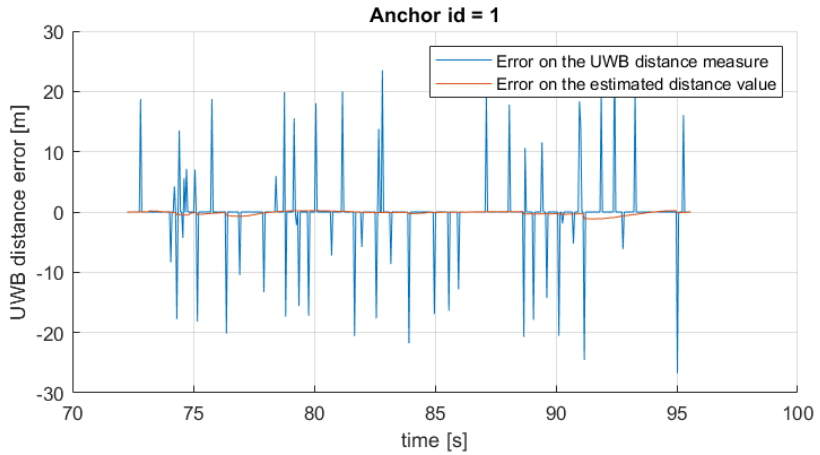


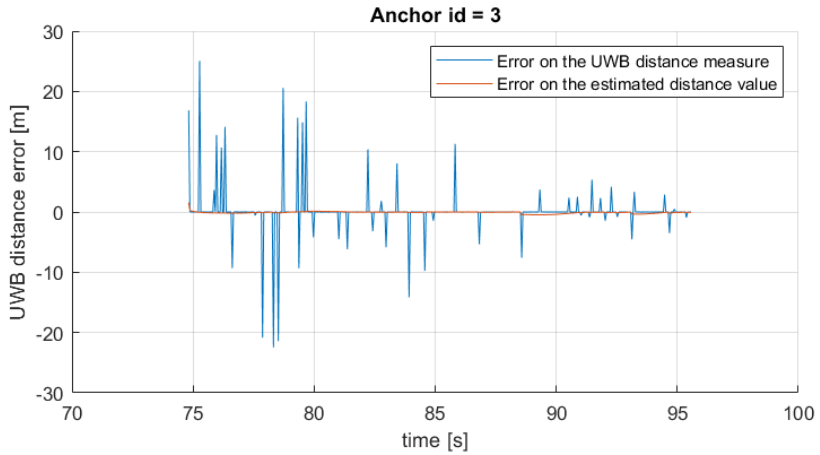


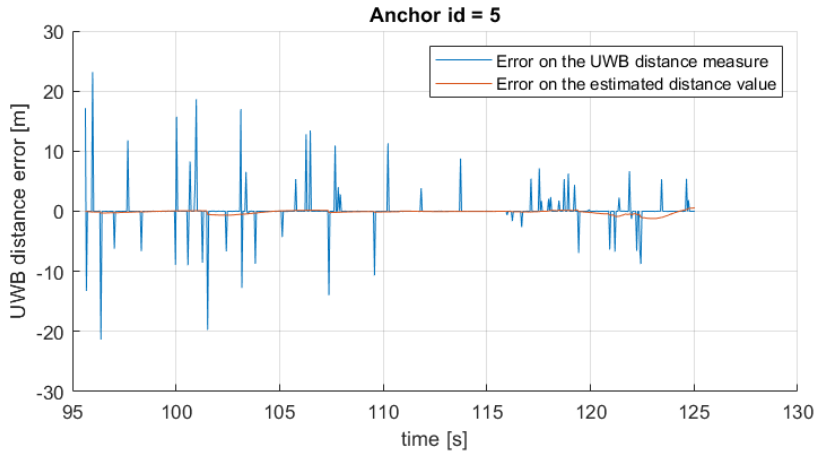


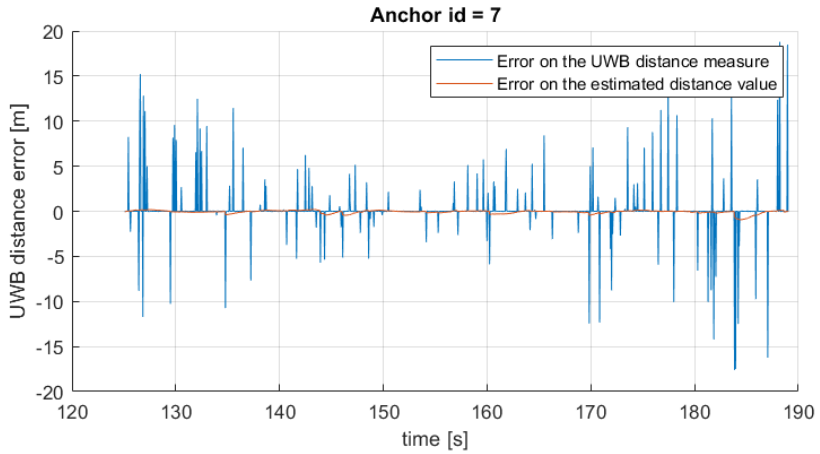


Simulations: Scenario 5 – Adaptive Filter









main_3

The behaviour of `fuse_unknown_uwb` is studied with an appropriate trajectory (appropriate in the sense that it is not singular).

It is first showed how accurate dead-reckoning navigation is.

Then, it is compared to the performance of the SLAM algorithm.

The duration of the simulated scenario is 40 seconds.

MATLAB R2020a - academic use

HOME PLOTS APPS EDITOR PUBLISH VIEW

Search Documentation | Davide

New Open Save | Find Files Compare Print | Go To Find | Insert Comment Indent | Breakpoints | Run Run and Advance | Run Section Advance | Run and Time

FILE NAVIGATE EDIT BREAKPOINTS RUN

C: ▸ Users ▸ david ▸ Dropbox ▸ Uni ▸ Progetto Sistemi di Guida e Navigazione ▸

Current Folder

Name

- Backup
- custom_functions
- datasets
- Slideshow
- Theory
- main_1.m
- main_2_v4.m
- main_2.m
- main_3.asv
- main_3.m
- spiral1.fig
- temp.m
- trajectory_data.mat

Details

Editor - C:\Users\david\Dropbox\Uni\Progetto Sistemi di Guida e Navigazione\main_3.m

main_3.m

```
1   %% Description
2   %
3   % Generate a coil trajectory. This trajectory is useful when there are
4   % unknown UWB beacons and no other absolute position sensors available.
5   % This type of trajectory allows the system to estimate the UWB anchor
6   % position and at the same time use them for navigation (SLAM). The beacon
7   % position is first be initialized with an appropriate algorithm.
8
9
10  %% Initialization
11
12  clc
13  clear variables
14  close all
15
16  sch = 'shuffle';
17
18  rng(sch)
```

Workspace

Name	Value
accBody	2000x3 double
accel	[0.0072,4.3218,8....
alt	1x1 altimeter_sen...
angVelBody	2000x3 double
ans	1
bar	1x1 barometer_se...
beacon_estim...	1x8 Graphics
decimation_f...	1
default_colors	7x3 double
dt	0.0200
duration	40

Command Window

```
Started.

>>
```

UTF-8 | script | Ln 1 | Col 1

Type here to search

27°C Sunny ITA 16:47 24/06/2021

MATLAB R2020a - academic use

HOME PLOTS APPS EDITOR PUBLISH VIEW

Search Documentation | Davide

New Open Save | Find Files Compare Print | Go To Find | Insert Comment Indent | Breakpoints | Run Run and Advance | Run Section Advance | Run and Time

FILE NAVIGATE EDIT BREAKPOINTS RUN

C: ▸ Users ▸ david ▸ Dropbox ▸ Uni ▸ Progetto Sistemi di Guida e Navigazione ▸

Current Folder

Name

- Backup
- custom_functions
- datasets
- Slideshow
- Theory
- main_1.m
- main_2.m
- main_3.m
- spiral1.fig
- temp.m
- trajectory_data.mat

Details

Editor - C:\Users\david\Dropbox\Uni\Progetto Sistemi di Guida e Navigazione\main_3.m

main_3.m | my_navigation_filter.m

```
1   %% Description
2   %
3   % Generate a coil trajectory. This trajectory is useful when there are
4   % unknown UWB beacons and no other absolute position sensors available.
5   % This type of trajectory allows the system to estimate the UWB anchor
6   % position and at the same time use them for navigation (SLAM). The beacon
7   % position is first be initialized with an appropriate algorithm.
8
9
10  %% Initialization
11
12  clc
13  clear variables
14  close all
15
16  sch = 'shuffle';
17
18  rng(sch)
```

Workspace

Name	Value
accBody	4000x3 double
accel	[-0.0808,4.3332,8...
alt	1x1 altimeter_sen...
angVelBody	4000x3 double
ans	1
bar	1x1 barometer_se...
beacon_estim...	1x8 Scatter
decimation_f...	1
default_colors	7x3 double
dt	0.0100
duration	40

Command Window

```
Started.

>>
```

UTF-8 | script | Ln 1 | Col 11

Type here to search

30°C Sunny | ITA | 17:19 | 27/06/2021

Bibliography

Bibliography

1) W. Youn and S. Andrew Gadsden, "Combined Quaternion-Based Error State Kalman Filtering and Smooth Variable Structure Filtering for Robust Attitude Estimation," in IEEE Access, vol. 7, pp. 148989-149004, 2019, doi: 10.1109/ACCESS.2019.2946609.

2) H. Bavle, J. L. Sanchez-Lopez, A. Rodriguez-Ramos, C. Sampedro and P. Campoy, "A flight altitude estimator for multirotor UAVs in dynamic and unstructured indoor environments," 2017 International Conference on Unmanned Aircraft Systems (ICUAS), 2017, pp. 1044-1051, https://doi.org/10.1109/ICUAS.2017.7991467

3) Woo, Rinara, Eun-Ju Yang, and Dae-Wha Seo. 2019. "A Fuzzy-Innovation-Based Adaptive Kalman Filter for Enhanced Vehicle Positioning in Dense Urban Environments" *Sensors* 19, no. 5: 1142. https://doi.org/10.3390/s19051142

4) Feng, Kaiqiang, Jie Li, Xiaoming Zhang, Chong Shen, Yu Bi, Tao Zheng, and Jun Liu. 2017. "A New Quaternion-Based Kalman Filter for Real-Time Attitude Estimation Using the Two-Step Geometrically-Intuitive Correction Algorithm" *Sensors* 17, no. 9: 2146. https://doi.org/10.3390/s17092146

5) Jiang, Chen, and Shu-Bi Zhang. 2018. "A Novel Adaptively-Robust Strategy Based on the Mahalanobis Distance for GPS/INS Integrated Navigation Systems" Sensors 18, no. 3: 695. https://doi.org/10.3390/s18030695

6) Li, Xin, Yan Wang, and Kourosh Khoshelham. 2018. "A Robust and Adaptive Complementary Kalman Filter Based on Mahalanobis Distance for Ultra Wideband/Inertial Measurement Unit Fusion Positioning" Sensors 18, no. 10: 3435. https://doi.org/10.3390/s18103435

7) Escamilla-Ambrosio, P. Jorge & Mort, N.. (2000). Adaptive Kalman filtering through fuzzy logic.

Bibliography

8) Yang Y. (2010) Adaptively Robust Kalman Filters with Applications in Navigation. In: Xu G. (eds) Sciences of Geodesy - I. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-11741-1_2

9) Norrdine, Abdelmoumen. (2015). An Algebraic Solution to the Multilateration Problem. https://doi.org/10.13140/RG.2.1.1681.3602

10) Liu, Jianfeng, Jiexin Pu, Lifan Sun, and Zishu He. 2019. "An Approach to Robust INS/UWB Integrated Positioning for Autonomous Indoor Mobile Robots" Sensors 19, no. 4: 950. https://doi.org/10.3390/s19040950

11) Q. Shi, S. Zhao, X. Cui, M. Lu and M. Jia, "Anchor self-localization algorithm based on UWB ranging and inertial measurements," in Tsinghua Science and Technology, vol. 24, no. 6, pp. 728-737, Dec. 2019, doi: 10.26599/TST.2018.9010102.

12) Xu, Shuqing, Haiyin Zhou, Jiongqi Wang, Zhangming He, and Dayi Wang. 2019. "SINS/CNS/GNSS Integrated Navigation Based on an Improved Federated Sage–Husa Adaptive Filter" Sensors 19, no. 17: 3812. https://doi.org/10.3390/s19173812

13) Chang, G. Robust Kalman filtering based on Mahalanobis distance as outlier judging criterion. J Geod 88, 391–401 (2014). https://doi.org/10.1007/s00190-013-0690-8

14) Almagbile, Ali & Wang, Jinling & Ding, Weidong. (2010). Evaluating the Performances of Adaptive Kalman Filter Methods in GPS/INS Integration. Journal of Global Positioning Systems. 9. 10.5081/jgps.9.1.33.

Bibliography

15) Roetenberg, Daniel & Luinge, Hendrik & Baten, Chris & Veltink, Peter. (2005). Compensation of Magnetic Disturbances Improves Inertial and Magnetic Sensing of Human Body Segment Orientation. Neural Systems and Rehabilitation Engineering, IEEE Transactions on. 13. 395 - 405. 10.1109/TNSRE.2005.847353.

THANK YOU FOR THE ATTENTION