Drone SLAM with Extended Kalman Filter using Ultra-Wide Band sensors and a Inertial Measurement Unit

Afonso Luis, Jose Silva, Samuel Amaro, Tiago Silva

***Abstract*— The following report will cover the topic of the simultaneous localisation and mapping with a drone. To do so an Extended Kalman Filter (EKF) was used with measurements from Ultra-Wide Band sensors (UWB) and from an Inertial Measurement unit (IMU). An interesting approach was also been made to determine the estimate of the initial system's state using Least Squares. In the end, both simulation and the real results are shown which validate both EKF and LS initialisation.**

***Index Terms*— SLAM, Extended Kalman Filter, Drone, Ultra-Wide Band, Inertial Measurement Unit, EKF, UWB, IMU.**

I. Introduction

The objective of this project was to correctly estimate the vehicle's position and pose, as well as the location of the three landmark features using a SLAM algorithm which is the Extended Kalman Filter (EKF). One of the main goals of this report is to provide the readers not only a solid introduction to Kalman Filters but also to give enough information about the filter's implementations to serve as a guideline for similar projects. The first section is dedicated to a brief explanation of the Kalman Filter followed by an in depth explanation of the Extended Kalman Filter. Then, in the next section the process of the filter design is explained as well as the procedure of it's initialisation. The final two sections are dedicated to the EKF's experimental implementation on both simulation environment and real world environment.

II. Theoretical Foundations

The key component for the solution to the SLAM problem is the Extended Kalman Filter (EKF). Therefore, it is crucial to have a solid understanding of it, which will be provided in this section. The EKF is a variation of the Kalman Filter used for linear systems, so firstly a brief overview of the linear filter will be made to introduce the concept and some notation, followed by a detailed exploration of the Extended Kalman filter.

A. Kalman Filter

The Kalman filter is an algorithm that uses a series of measurements z_k over time and produces estimates of unknown variables x_k that tend to be more accurate than those based on a single measurement alone.

The algorithm is a two-step process: the prediction step and the update step.

1) Prediction Step:

In the prediction step, the algorithm makes a prediction of the current state of the unknown variables $\hat{x}_{k+1}$ based on the previous state x_k. To do so, the filter must know an approximation of the system's behaviour given the current state. Said approximation will be from now on referred as the motion model $f(x)$, meaning that the prediction step can be described by

$$\hat{x}_{k+1} = f(x_k) + \epsilon, \tag{1}$$

where ϵ is the variance between the motion model and the vehicle's true state.

2) Update Step:

The second step is known as the Update step in which the filter makes a correction to the prediction based on the available system measurements z_{k+1}. To do so the filter must have an approximation on which observations it should see, given the current state. Said approximation will be referred as the observation model $h(x)$ such that

$$z_{k+1} = h(\hat{x}_{k+1}) + \delta, \tag{2}$$

where δ is the variance between the sensor readings and the output of the observation model.

To apply the correction to the initial prediction, the Kalman Gain K_k is computed, which will be covered in detail in the next section. Then the correction is applied by matching both steps as follows

$$x_{k+1} = \hat{x}(k+1) + K_k(z_{k+1} - h(\hat{x}_{k+1})). \tag{3}$$

The algorithm's estimations are done by computing the joint probability distribution over the variables for each time frame. To do so the filter yields the exact conditional probability estimate in the special case that all errors are Gaussian.

As mentioned before the intent of this section was to introduce the concept of the Kalman filter and some of it's notation, meaning that some of the intermediate steps, although extremely necessary, were occluded and will be covered in the section.

In summary the Kalman Filter can be applied to estimate ANY number of unknown variables, in ANY linear system with behaviour described by $f(x)$, whose observations can be modelled by $h(x)$ assuming that the system's errors and uncertainties can be assumed to have a Gaussian Probability Distribution.

B. Extended Kalman Filter

In most cases, the desired unknowns come from an non-linear system which means that the previously showcased Kalman filter does not work. However not everything is lost since the Kalman filter principal can still be used with some modifications to it's equations. Unlike it's linear counterpart, the extended Kalman filter in general is not an optimal estimator and, if the initial estimate of the state is wrong, the filter will quickly diverge

1) Prediction Step:

The Extended version of the Kalman filter uses both prediction and update steps that were previously mentioned and the three shown equations are used as well. Starting with the prediction step, the expected value for the state is given by equation (1), but since the filter results are based on the the joint probability distribution over the variables, the variance and co-variance of each variable must be computed. To do so, first the system is linearized around the previous state, computing it's matrix $F \in R^{D \times D}$ (D is the size of the state

vector, or number of unknowns) given by the Jacobian of the motion model

$$F_{k+1} = \frac{\partial f}{\partial x}(x_k). \tag{4}$$

It is now possible to store a prediciton of each unknown's variance and covariances by computing the $\hat{\Sigma}_{k+1}$ matrix which is given by

$$\hat{\Sigma}_{k+1} = F_{k+1}\Sigma_k F_{k+1}^T + Q, \tag{5}$$

where Q corresponds to a $D \times D$ diagonal matrix containing estimates of the uncertainty ϵ introduced in (1).

2) Update Step:

The same goes for the update step. The observations expected values are computed by the observation model as mentioned in (2). The goal of the update step is to adjust the previous estimate by computing the Kalman Gain. In order to do that, the Jacobian of the observation model needs to be computed as the matrix $H \in R^{N \times D}$ where D is the size of the state vector and N the number of different observations available.

$$H_{k+1} = \frac{\partial h}{\partial x}(\hat{x}_{k+1}). \tag{6}$$

The Kalman Gain K_{k+1} can now be computed as

$$S_{k+1} = H_{k+1}\hat{\Sigma}_{k+1}H_{k+1}^T + R, \tag{7}$$

$$K_{k+1} = \hat{\Sigma}_{k+1}H_{k+1}^T S_{k+1}^{-1}, \tag{8}$$

the term R is similar to Q, it as a diagonal $R^{N \times N}$ matrix containing the variation of each measurement.

Finally, the matching of the two previous steps, prediction and update can be made, estimating the current state by

$$x_{k+1} = \hat{x}(k+1) + K_{k+1}(z_{k+1} - h(\hat{x}_{k+1})). \tag{9}$$

$$\Sigma_{k+1} = (I - K_{k+1}H_{k+1}E_{k+1})\hat{\Sigma}_{k+1}. \tag{10}$$

.

III. Implementation

A. Sensors

Before starting the filter design, it is important to evaluate the system to figure out which observations can be obtain with the given sensors and what are systems limitations and advantages.

1) Ultra-Wide Band (UWB):

The Ultra-Wide Band sensors are composed by anchors and tags. The anchors are attached to each landmark and the tag is attached to the Drone. With this setup, it is possible to measure the distance from the Drone (Tag) to each landmark (Anchor), but it is not possible to figure out the tag's direction, in relation to the anchor. Also, the sensor readings provide the variance of each measurement, which will be useful later on.

2) Inertial Measurement Unit (IMU):

The Inertial Measurement Unit provides readings on a body's specific force and angular rate. It usually used in air crafts for localisation using dead reckoning. The available IMU can be considered very low budget, meaning that its readings are very noisy, thus not making it adequate for dead reckoning use.

3) Drone:

Although it is not a sensor, it is also important to know the limitations of the used vehicle. The drone is going to be used almost as an hovercraft, flying at constant altitude, small variations in roll (α) and pitch (β) angles and its orientation θ can change freely. Therefore some assumptions can be later on made. Also, being a drone means that there is no odometry available.

B. Motion Model (Prediction)

The following sections will be dedicated to the filter design, starting by the filter's prediction step. The classic approaches or the system's motion model are based on odometry or dead reckoning, but like it was mentioned before, the current system neither has odometry available nor is a good idea to use a low budget IMU for position prediction. Therefore an alternative solution is needed.

Suppose the drone will maintain roll and pitch angles relatively constant. This means that the speed at which he translates through space will not be very significant. Therefore, a candidate for a motion model could be a 0'th order state model given by

$$x_{k+1} = x_k + \epsilon, \tag{11}$$

where ϵ is treated as noise. This is a solid candidate but it relies too much on the assumption that the drone will be flying at low speeds which is very restrictive and unlikely.

Alternatively, a first order state model could be considered where, instead of preserving the position, the system preserves velocity which is equivalent to assuming that the drone will not make sudden changes to its trajectory. Said model can be expressed by

$$\begin{cases} x_{k+1} = x_k + v_{k+1} \\ v_{k+1} = v_k + \epsilon \end{cases}. \tag{12}$$

It can also be considered an even more general motion model given by a second order state model, where the system preserves its speed but introduces acceleration as a function of the the drone's pitch and roll angles, expressed by

$$\begin{cases} x_{k+1} = x_k + v_{k+1} \\ v_{k+1} = v_k + \mu_{k+1} \\ \mu_{k+1} = \mu(\alpha_k, \beta_k) + \epsilon \end{cases}. \tag{13}$$

The complexity of this solution compared to the previous one is much higher. Given the fact that the drone will nor be making sudden changes to its trajectory, the improvements that the third solution gives are not worth its complexity and also, due to shortage of time, said solution was not implemented, instead the first order state model was chosen.

Now that the system's state model is defined the only thing left to do is to define the elements of the state vector and compute the matrix F. Since this is a SLAM problem, the Drone's x, y, α, β and θ must be present in the state vector as well as the landmarks positions (Li_x and Li_y). Additionally, the linear and angular velocities ($v_{x,y}, w_{\alpha,\beta,\theta}$ respectively) are also needed in the state vector. For the latter on simplicity when programming the filter, the variables were arranged in the following order

$$x_k = \begin{bmatrix} x & y & v_x & v_y & l_{1_x} & l_{1_y} & \dots & l_{N_x} & l_{N_y} & \alpha & \beta & \theta & w_\alpha & w_\beta & w_\theta \end{bmatrix}^T. \tag{14}$$

The matrix F considering the selected order of the state vector is given by

$$F = \begin{bmatrix} 1\,0\,1\,0\,0\,0\,...\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,1\,0\,1\,0\,0\,...\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,1\,0\,0\,0\,...\,0\,0\,0\,0\,0\,0\,0\,0 \\ ...\ ...\ ...\ ...\ ...\ ...\ ...\ ...\ ...\ ...\ ...\ ...\ ...\ ...\ ... \\ 0\,0\,0\,0\,0\,0\,...\,0\,1\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,...\,0\,0\,1\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,0\,0\,...\,0\,0\,0\,1\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,0\,...\,0\,0\,0\,0\,1\,0\,0\,1 \\ 0\,0\,0\,0\,0\,0\,...\,0\,0\,0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,0\,0\,...\,0\,0\,0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,0\,...\,0\,0\,0\,0\,0\,0\,0\,1 \end{bmatrix}. \quad (15)$$

C. Observation Model (Update)

In the observation model, it is wished to correct the estimates of every unknown in the state vector. This can be divided into two parts, the first one concerning measurements from the UWB sensors, and the second from the IMU.

1) UWB Readings:

The observations given by the UWB are distances between the Tag position T = (x, y) and a anchor k position A_k = (l_{k_x}, l_{k_y}), which can be described by

$$d_k = ||A_k - T|| = \sqrt{(l_{k_x} - x)^2 + (l_{k_y} - y)^2} \quad (16)$$

This could be considered the observation model but since the norm's gradient isn't particularly simple to compute some manipulations can be made to improve the simplicity of the model. Squaring both sides of the equations of the model is an alternative, which gives

$$d_k^2 = (l_{k_x} - x)^2 + (l_{k_y} - y)^2 \quad (17)$$

The partial derivatives with respect to the drone's position (x,y) and to the landmarks positions (l_{k_x}, l_{k_y}) are

$$\begin{cases} \frac{\partial d_k^2}{\partial x} = 2(x - l_{k_x}) \\ \frac{\partial d_k^2}{\partial y} = 2(y - l_{k_y}) \\ \frac{\partial d_k^2}{\partial l_{k_x}} = 2(l_{k_x} - x) \\ \frac{\partial d_k^2}{\partial l_{k_y}} = 2(l_{k_y} - y) \end{cases} \quad (18)$$

where i is the H matrix column index, and j is the i'th landmark's x coordinate index in the state vector.

2) IMU Readings:

As mentioned before, low-budget IMU readings are too noisy and therefore can not be used in the prediction step. However, they can be used in the update step by comparing the predicted angular velocity with the angular rate given by the imu. This can be described as

$$\begin{bmatrix} \Omega_\alpha \\ \Omega_\beta \\ \Omega_\theta \end{bmatrix} = \begin{bmatrix} w_\alpha \\ w_\beta \\ w_\theta \end{bmatrix}. \quad (19)$$

The Observation model can now be completed by merging the previous cases

$$h(x_k) = \begin{bmatrix} h_1(x_k) \\ ... \\ h_N(x_k) \\ h_4(x_k) \\ h_5(x_k) \\ h_6(x_k) \end{bmatrix} = \begin{bmatrix} (l_{1_x} - x)^2 + (l_{1_y} - y)^2 \\ ... \\ (l_{N_x} - x)^2 + (l_{N_y} - y)^2 \\ w_\alpha \\ w_\beta \\ w_\theta \end{bmatrix}, \quad (20)$$

The Jacobian matrix H has dimensions [N x D], where N is the number of measurements (6 in this case) and D the dimension of the state vector, and is given by

$$H = \frac{\partial h}{\partial x_k} = \begin{bmatrix} \frac{\partial h_1}{\partial x} & \frac{\partial h_1}{\partial y} & \frac{\partial h_1}{\partial l_{1_x}} & \frac{\partial h_1}{\partial l_{1_y}} & ... & \frac{\partial h_1}{\partial \omega_\theta} \\ ... & ... & ... & ... & ... & ... \\ \frac{\partial h_n}{\partial x} & \frac{\partial h_n}{\partial y} & \frac{\partial h_n}{\partial l_{n_x}} & \frac{\partial h_n}{\partial l_{n_y}} & ... & \frac{\partial h_n}{\partial \omega_\theta} \end{bmatrix}. \quad (21)$$

D. Initialisation

As mentioned before, a good initialisation is crucial for a better performing EKF. The SLAM turns in a somewhat paradoxal problem: it is desired to estimate the vehicle's location and pose, as well the location of each landmark, but in order to do so it is required to know almost the exact value of each unknown *a priori*. Although it might seem like an impossible problem, a practical solution was found by dividing the problem into two: first solve the drone's pose and location initialisation and then the different landmarks positions.

1) Drone's pose and location:

The EKF algorithm creates it's own referential, meaning that any trajectory and map when applied a rotation and/or a translation still correspond to the starting ones. Meaning that to the algorithm, the absolute values of each unknown do not matter, only the relations between them.

Knowing this, the solution to the first problem becomes trivial: the drone's initial location is assumed to be the reference and with that $x_1(1) = 0$, $x_1(2) = 0$. Not only this can be considered as a valid approximation, but it can also be considered as the exact initial position of the vehicle, meaning that the initial variance of the vehicle's location can be initialised as 0.

The same can be applied to the drone's pose.

2) Landmark locations:

Previously it was mentioned that the sensors used to measure the distance between the drone and landmarks where UWB sensors, which can only measure the distances without any indication of the direction. Since the distance between the three landmarks is also unavailable, with all 4 elements (drone plus 3 landmarks) stationary, it is impossible to figure out the landmarks locations.

The solution is based on using trilateration where several drone positions over time are used as anchors to estimate each landmark position as seen on Figure 1.

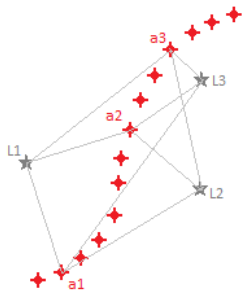

Fig. 1. Illustrative example of using vehicle positions as anchors in trilateration.

Assuming for now that the several vehicle's positions are known, the i'th landmark location can be obtain by solving

$$\begin{cases} (Li_x - a1_x)^2 + (Li_y - a1_y)^2 = d_{i,1}^2 \\ (Li_x - a2_x)^2 + (Li_y - a2_y)^2 = d_{i,2}^2 \\ (Li_x - a3_x)^2 + (Li_y - a3_y)^2 = d_{i,3}^2. \end{cases} \quad (22)$$

Now we face 2 problems. Firstly, in a real-world scenario, sensor measurements are not perfect, meaning that (22) is impossible to solve. The second problem is that the drone's position is not known, so trilateration can't be applied.

The first problem is of easy solution as it is a commonly known optimisation problem given by

$$\begin{aligned} &min \quad ||x_i - a_k||^2 - d_{i,k}^2, \\ &x_i \in R^2 \end{aligned} \tag{23}$$

where k is the number of known vehicle locations, x_i the i'th landmark location to be estimated, a_k the k'th know location and $d_{i,k}$ the observed distance between location k and landmark i.

To solve this problem there are plenty of solutions[1], due to it's simplicity the chosen method was the Least Squares. This method Requires the optimisation problem to be written in the following form

$$\begin{aligned} &min \quad ||Ax - b||^2. \\ &x_i \in R^2 \end{aligned} \tag{24}$$

Therefore, some algebraic manipulation is required in the problem formulation in order to solve it. Since optimisation problems are not the focus of this study, said manipulations will be skipped. The result is given by

$$\begin{aligned} &min \quad ||2(a_1 - a_{k+1})^T x_i - (d_{i,1}^2 - d_{i,k+1}^2 + ||a_k + 1||^2 - ||a_1||^2)||^2. \\ &x_i \in R^2 \end{aligned} \tag{25}$$

where the matrices A and b in (24) are

$$A = \begin{bmatrix} 2(a_1 - a_2)^T \\ ... \\ 2(a_1 - a_N)^T \end{bmatrix}, \; b = \begin{bmatrix} d_{i,1}^2 - d_{i,2}^2 + ||a_2||^2 - ||a_1||^2 \\ ... \\ d_{i,1}^2 - d_{i,N}^2 + ||a_N||^2 - ||a_1||^2 \end{bmatrix}. \tag{26}$$

All there is left to do now is to figure out a way to know the different drone locations. There are several approaches, one could be applying the EKF without the update step, meaning that the vehicle locations would be given by the motion model only. The motion model in this case is not adequate for this procedure since it is a bit "clueless". The IMU could also be used for dead-reckoning but, for reasons mentioned before, this will not be considered.

Therefore, the only way that the vehicle's trajectory can be estimated is to force a known trajectory to it. To do so, before starting the EKF, the drone is forced to travel from a landmark to another moving straightforward on x axis. Reminding that EKF creates its own referential, the final landmark can be initialised as (0,0) with 0 variance and the first landmark can be initialised as (-D, 0), where D is the distance to the final landmark when the drone started it's forced movement. This landmark can be initialised with variance equal to the observations variance given by the UWB sensors.

The positions at each moment are given by (-d(k),0) where d(k) is the distance to the final landmark at the instant k. Feeding every recorded position and distance observed to the Least Squares Algorithm, every landmark other than the 2 used in the forced movement phase can be obtained by computing $A\backslash b$ and initialised with variance equal to the squared of the largest error observed in the Least Squares algorithm.

IV. Simulation

Before using real data gathered from sensors, several MATLAB® simulations were made to evaluate the algorithm's performance. These simulations do not represent a fully real problem because situations where the drone's orientation was independent of its speed were considered.

The simulations acknowledge that the sensor readings are noisy and simulates them by adding uniform noise with a maximum deviation of 0.2 units on all 3 UWB readings and 0.002 on the 3 angular velocity IMU readings.

Since the maximum deviation values are known, it is possible to build the matrix R where the first 3 diagonal elements correspond to a value similar to the maximum deviation of the UWB readings and the last 3 elements similar to the IMU deviations. The values 0.2 and 0.01 respectively were chosen since they produced good enough results on all 3 simulations. Also, the Q matrix is a diagonal matrix given by

$$Q = diag([0.01\ 0.01\ 0.1\ 0.1\ 0\ 0\ 0\ 0\ 0\ 0\ \frac{\pi}{15}\ \frac{\pi}{15}\ \frac{\pi}{15}\ 1\ 1\ 1]), \tag{27}$$

these values were essentially fine tuned to fit all proposed scenarios well.

A. Scenario I

The first scenario corresponds to drone's circular trajectory given by [1.5cos(0.1t), 1.5sin(0.1t)], where its yaw angular rate corresponds to 0.2sin(0.1t) and the other two angular rates are kept at 0. The simulation can be seen in the following video demonstration

This scenario is a good starting point to evaluate the decisions made in the implementation since it does not violate heavily the motion model (the drone does not make any sudden trajectory changes). Also the drone's angular motion should not be difficult for the filter to estimate it.

1) Initialisation:

Firstly it is desired to evaluate the initialisation procedure previously described. As it can be seen in the filter's first iteration the landmark used as the finish mark of the initialisation process is initialised exactly on the location of the actual landmark (0,0). The landmark where the initiation procedure started is initialised very close to the actual location (deviation of 0.053 units), it is not capable of initialising as accurate as the previous one since it's location is based on a single UWB reading which is noisy. The remaining landmark appears to be much deviated from the actual location, this once again due to the deviated UWB readings which heavily influence the deviation in the LS algorithm as shown. The variances are initialised as described in section IV. The goal of this process was to get a rough estimate of all landmark positions and an estimate of the deviation of this process so that the filter could latter on converge. Given that the filter not only did converge but in a rather low number of iterations needed, the initialisation process is solid enough.

2) Drone and Landmark locations:

Since this is a rather simple scenario it was expected that the EKF would correctly estimate both landmark and drone positions.

The filter's estimate for the drone's position is in general really close to the actual position and the estimated trajectory is rather smooth with barely unnoticeable oscillations. However, there is a

significant deviation between the estimation and the actual position close to the origin. This happens due to very low differences in the expected and measured distances to the landmarks, since the drone's trajectory includes this landmark, the UWB readings will be very low in this section. Since the observation model uses the squared distances, the difference between the expected observations and the actual ones will be very small so the algorithm thinks the estimate is very close to the actual state.

The initial landmark positions were already a good state estimate and as the time passes the correction of this estimate is clearly visible especially on the LS estimated landmark. The final landmarks estimate is almost exactly the actual landmark positions which confirms that the initialisation process provides not only a good enough estimate but also a good variance to begin with.

3) Drone's orientation:

In the simulation it is clearly visible a significant deviation from the estimated orientation to the actual orientation. This is due to the fact that the motion model is basically an integration of the drone's angular speeds which are corrected by the update step. These angular rates are rarely the exact actual values, which means that the motion model is integrating a slightly deviated error and in the end the estimated orientation will visibly differ from the actual one. This deviation however is not very worrying since when the angular rates change direction the filter can slightly correct itself, which reduces the deviation each time the angular velocities change directions. As a result, the rotation axis z" is the one which presents a larger deviation.

B. Scenario II

The first additional scenario will consist in a square trajectory, which will violate the assumption of no sudden changes of direction. Also, to test the robustness of the orientation estimate a more complex angular motion is presented where the drone starts rolling with a rate of $\frac{\pi}{180}$ through 60 iterations. Between iteration 40 and 100 the drone begins tilting forward at a rate of $\frac{\pi}{360}$ and lastly, from iterations 80 to to 140, the angular rate of orientation is $\frac{\pi}{180}$.The simulation can be seen in the following video demonstration

1) Location estimate robustness:

The main goal of this scenario is to evaluate the filter's reaction when the motion model is violated. As it can be seen in the simulation, the estimation follows the actual position pretty well even when the trajectory changes abruptly. In this case it was not needed since the selected values for both Q and R matrices were well selected, but in case the estimate wasn't following the actual position when the trajectory changes. The values in R should be lowered and the values on Q raised so that the filter has more trust on the sensor readings than in the estimate of the prediction step.

2) Orientation Estimate:

The obtain results resemble the observations stated previously. Even when the complexity of the drone's angular motion is increased, the EKF still manages to produce an end result similar to a simpler case, which is positive.

C. Scenario III

This final scenario was made to put the filter to the limit: the drone describes a really rocky trajectory with linear patches, sudden trajectory changes and various twists and turns. The same was applied for the drone's angular motion. The simulation can be seen in the following video demonstration

1) Overall Results:

As can be seen in the simulation, the filter has no problem what so ever in estimating correctly the vehicle's position even in the toughest trajectory patches. The prediction of the angular motion is once again similar to previous scenarios where it follows the actual trajectory closely but with a slight deviation.

V. Using Real Data

In this section real data was used to evaluate the algorithm's performance locating both the robot and the landmarks.

The measurements were carried out in the laboratory of autonomous systems in the fifth floor, room LSDC4.

The positioning of the landmarks is represented in Figures 2 and 3, where the green circles represent the locations where the landmarks were placed.

Fig. 2. Positions of the landmarks in the laboratory.

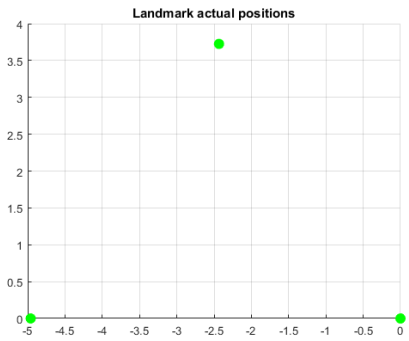

Fig. 3. Positions of the landmarks.

1) Drone's Trajectory:

The drone described an almost elliptical trajectory starting in the reference landmark and finishing in the one with zero y coordinate, passing through 4 key points, being the 4 directional maximums and minimums of an ellipse.

The robot kept its height relatively constant but it was difficult to maintain it exactly the same throughout the motion.

2) Trajectory estimate:

The ellipse estimated by the algorithm in general is very close to the actual trajectory (Figure 4), the exception being the bottom left part of the ellipse. This has already been seen in the first two simulations. When the drone path crosses a landmark, the algorithm tends to deviate its estimate from the actual trajectory due

to the landmark proximity. As seen also in the simulations, if the drone made another lap around the landmarks, this deviation would be slightly corrected. The most problematic region is the drone's final movements where it is almost stationary around the leftmost landmark. The reason why is because when the drone's not moving, the observation errors induce the algorithm in thinking that the vehicle is actually moving due to slightly different readings from an iteration to the next one. This results in the noisy estimates around the leftmost landmark.

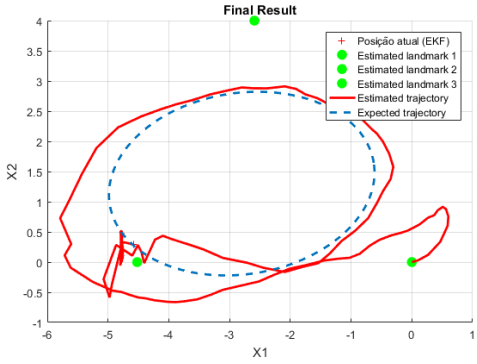

Fig. 4. Comparison between the estimated path (red) and the robot's actual path (blue).

3) Landmarks estimate:

As for the landmarks estimates, by comparing Figures 3 and 4, it can be said that the filter correctly estimated the landmark's positions.

It is evident that the results shown in this section are much worse than the ones studied in the Simulation section.

4) Orientation estimate:

Unfortunately, there is no real data available to showcase regarding the drone's orientation estimate. Some data was collected in form of rosbags but for some reason the information in them was useless and due to time constraints and difficulties obtaining the necessary components for recording the needed data, it was not possible to gather usable information.

5) Additional scenario:

Similar to the Simulation section, it is wished to evaluate situations where the motion model is violated, meaning a square trajectory where the drone follows the laboratory ground's squared pattern, describing a rectangle with 1.5m by 1.8m . In Figure 5 the final result is shown.

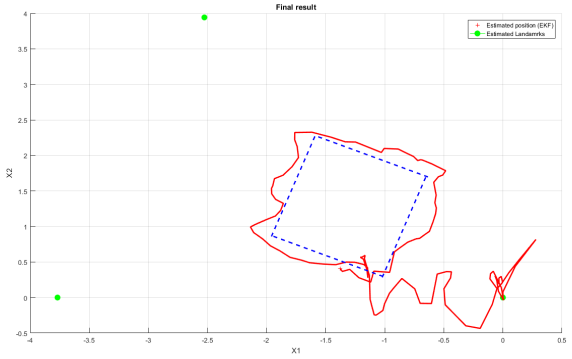

Fig. 5. Comparison between the estimated path (red) and the robot's actual path (blue).

The found result is very noisy but besides that, it goes according to the results found in the corresponding simulation section. The noisiness of the end result is due to the laboratory being very crowed which means lots of movement and lots of electrical equipment that can generate noise in the observations. Nevertheless the values of the matrices Q and R where adjusted in order to make the filter more confident on the prediction step (lower Q values and raise R values). The price to pay is the slow trajectory changes evident in the 4 rectangle corners as expected.

6) Final comments:

It is evident that this section does not cover all the key-points from the simulation section. Once again, time constraints and material difficulties negatively impacted the project. Since both IMU and UWB sensors where not very precise, the test conditions had to be almost impeccable in order to get smooth results as the ones in Figure 4. Most of the time the laboratory was very crowded which generates lots of noise in the observations. Different scenarios where tested but due to the observation noise, these where discarded remaining only the two presented scenarios.

VI. Conclusions

The problem of simultaneously estimating both mapping features and the vehicle's localisation might seem impossible. But as seen in this report, by dividing it into it's key components and solving each of them one by one, the highly complex problem becomes one with simple solutions.

The overall's solution main component is without the Extended Kalman Filter and although it is a very simple Algorithm, it is essential to have solid knowledge of its applications and versatility in order to efficiently use it. Also, it is even more important to know the advantages and limitations of the system in use as well as its sensors. As mentioned, the drone's behaviour implies that some acceptable assumptions can be made which simplifies the system's modelling and the overall solution.

For the filter's correct function a good initialisation is needed. To do so an initialisation procedure was implemented using Least Squares with an unorthodox routine, which produced good results not only for initialising the landmark's estimate but also its variance.

Overall, the filter's response in both simulation and in the real world where robust and comply to the desired outcomes for both landmark and position estimates, the orientation estimate, although being capable of following the actual orientation, does present a significant deviation.

The main difficulties of the project were perhaps finding a way to initialise the landmarks positions which involved some "out of the box" thinking and also the implementation of the algorithm using real data. This last one probably took 70% of the project time due to our little knowledge working with the materials.

References

[1] Jiahong Li, Xianghu Yue, Jie Chen * and Fang Deng * , "A Novel Robust Trilateration Method Applied to Ultra-Wide Bandwidth Location Systems " Key Laboratory of Intelligent Control and Decision of Complex Systems, School of Automation, Beijing Institute of Technology, Beijing 100081, China.

[2] Maria Isabel Ribeiro Pedro U. Lima, "Course Handouts" Instituto Superior Tcnico/Instituto de Sistemas e Robtica 2018/2019.