Drone Localization Using EKF and Markers

Marina Moreira, *78460* Sara Sequeira, *76258* Sofia Fernandes, *78563* and Tiago Valentim, *78400*

Abstract—In this project we estimate the pose (position and orientation) of a *crazyflie* drone by applying an Extended Kalman Filter (EKF) algorithm. Position and orientation were obtained using an on board analog camera to detect *alvar* markers. The orientation information is fused with IMU sensor data that provided angular rate (for integration) and acceleration (for bias correction). It should be noted that although the IMU sensor provided acceleration, it was not used to derive position as measurements were not reliable. Thus, the only source of position estimation was by detecting the *alvar* markers. In order to evaluate the estimated results, data from the *OptiTrack* Motion Capture System was used as reference.

Index Terms—Localization, alvar markers, IMU, EKF, ROS.

1 Introduction

AUTONOMOUS systems have been increasingly researched and used either for commercial or military use. The tasks performed by the robots are not only to make everyday life easier but also to ease important applications such as surveillance and rescue missions. Many of these tasks take place in circumstances where there is no access to Systems like GPS and, hence, a certain level of position estimation is necessary.

In fact, the motivation to proceed with this project was the challenging pose estimation associated with the poor camera quality.

Through this paper it will be described the methods and algorithms used and its implementation as well as experimental results.

2 Methods and Algorithms

2.1 Extended Kalman Filter

The Kalman Filter is an iterative mathematical process that uses a set of equations and consecutive data inputs to quickly estimates the true position or velocity of the object being measured when the measured values contain unpredicted or random errors, uncertainty or variation. The Extended Kalman Filter (EKF) consists of the implementation of the Kalman Filter for a system dynamics that results from the linearization of the original non-linear filter dynamics around the previous state estimates. The system dynamics of the Kalman Filter is represented by the state X with measurements Z by the following functions:

$$X_{k+1} = f(X_k, U_k) + v(k), v(k) \sim \mathcal{N}(0, Q_k) \tag{1}$$

$$Z_k = h(X_k) + w_k, w_k \sim \mathcal{N}(0, R_k) \tag{2}$$

Where v_k and w_k are the process and observation noises with a Gaussian distribution an U is the control variable matrix. The Extended Kalman Filter consists on a cycle of prediction, matching and update and there are three main equations that need to be considered. Firstly calculating the Kalman Gain, then the estimate, each time the estimate is updated, and finally the error in the estimate. In the prediction state it is defined a predicted state based on the physical model and the previous state. The new state (predicted) is represented in the following equations:

$$\hat{X}_{k+1|k} = f(\hat{X}_{k|k}, U_k) \tag{3}$$

$$\Sigma_{k+1|k} = F\Sigma_{k|k}F^T + Q_k \tag{4}$$

Where Σ is the process covariance matrix and Q is the process noise covariance matrix. After de prediction, the matching step occurs. In the EKF, this step combines the measurements and the prediction. This reduces the risk of the iterative posture estimation process to diverge and discards noisy observations and erroneous or inexistent predictions. The matching step is processed as follows:

$$v_{ij_{k+1}} = z_{j_{k+1}} - \hat{z}_{i_{k+1|k}} \tag{5}$$

Where $z_{j_{k+1}}$ refers to the observations and $\hat{z}_{i_{k+1|k}}$ refers to the prediction.

$$S_{ij_{k+1}} = E[v_{ij_{k+1}} v_{ij_{k+1}}^T] \tag{6}$$

$$v_{ij_{k+1}}^T S_{ij_{k+1}} v_{ij_{k+1}} \leq \gamma \tag{7}$$

This corresponds to a validity region centered in $\hat{z}_{i_{k+1|k}}$. If the inclusion of $z_{j_{k+1}}$ in the validity region is confirmed, the pair $(\hat{z}_{i_{k+1|k}}, z_{j_{k+1}})$ is valid and can be used for the next step, the update step.

In the updating step, the state is updated with the new measurement, Z, and takes place the calculation of the Kalman Gain, K.

$$S_{k+1} = H\Sigma_{k+1|k}H^T + R_{k+1} \tag{8}$$

$$K_{k+1} = \Sigma_{k+1|k}H^T S_{k+1}^{-1} \tag{9}$$

Updating the state, X, and the covariance matrix, Σ, we get:

$$\hat{X}_{k+1|k+1} = \hat{X}_{k+1|k} + K_{k+1}(Z_{k+1} - \hat{Z}_{k+1|k}) \tag{10}$$

Last update: 10 December 2017

$$\Sigma_{k+1|k+1} = \Sigma_{k+1|k} - K_{k+1} S_{k+1} K_{k+1}^T \tag{11}$$

The F matrix corresponds to the state transition matrix and H to the observation matrix. These are defined by the following Jacobians:

$$F = \left.\frac{\partial f}{\partial X}\right|_{X=\hat{X}_{k+1|k}, U=U_k} \tag{12}$$

$$H = \left.\frac{\partial h}{\partial X}\right|_{X=\hat{X}_{k+1|k}} \tag{13}$$

2.1.1 Quaternions

In this project, quaternions were used instead of Euler angles as there are some advantages when it comes to gymbal lock and smooth interpolation, with the downside that they rely on advanced math. The quaternion is defined as

$$\bar{q} = q_0 + q_1 i + q_2 j + q_3 k \tag{14}$$

where i, j and k are hyperimaginary numbers that satisfy

$$\begin{gathered} i^2 = -1, \quad j^2 = -1, \quad k^2 = -1, \\ -ij = ji = k, \quad -jk = kj = i, \quad -ki = ik = j \end{gathered} \tag{15}$$

q_0 corresponds the real part of the quaternion while $q_1 i + q_2 j + q_3 k$ corresponds to the imaginary. This way the quaternion can be written as a column matrix in the following way:

$$\bar{q} = \begin{bmatrix} q_0 \\ \mathbf{q} \end{bmatrix} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T \tag{16}$$

Another definition is the quaternion of rotation which satisfies

$$|\bar{q}| = \sqrt{\bar{q}^T \bar{q}} = \sqrt{|\mathbf{q}|^2 + q_0^2} = 1 \tag{17}$$

For quaternion multiplication, it is also important to define the skew-symmetric matrix operator $\lfloor q \times \rfloor$ as:

$$\lfloor q \times \rfloor = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix} \tag{18}$$

2.2 ROS

The Robot Operating System (ROS) is an open-source system mainly used in robots. It provides the services one would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.

ROS was a natural choice for us as we were really aiming for a framework that could receive the data from the robot in real-time, and there is already compatible packages with the *Crazyflie* and *Optitrack* system available. One major advantage is that the ROS environment allows for easy observer pattern handling, critical for our needs.

2.3 AR Track Alvar

For the marker identification, the *ar_track_alvar* ROS package was used. It is a wrapper for Alvar, an open source AR tag tracking library. This library allows for tracking of 2D markers, accurately and fast. This package is also able to receive the camera distortion information (very important for our case as the image from the *Crazyflie* is distorted).

3 Implementation

3.1 ROS Architecture

The ROS architecture was defined as in Figure 1. Each square represents a node (the Crazyflie is represented as only one node for simplicity). All blue nodes have a functional relation to the Kalman filter, the green nodes are for visualization of the data obtained to evaluate the performance of the algorithm.

Crazyflie: Represented as a node, receives data from the IMU antenna (\crazyflie_imu) as well as the camera antenna(\image_raw) and publishes it. The \camera_info topic comes from the calibration process.

Tracker Identification: Receives the \image_raw and \camera_info topic from Crazyflie and publishes the transformations (pose, as position + attitude in quaternions) from the drone to the respective marker it detected.

Invert marker tf: This node only exists for visualization in Rviz. As the transformations from the previous node need to be inverted. If they weren't the origin would be redefined each time we have new data, allowing for poor visualization (flashing).

Fixed tf broadcaster: This node broadcasts each marker pose in respect to the origin (and origin to *OptiTrack*'s world reference). Important for visualization is Rviz, as marker location is unknown otherwise.

Kalman Filtering: This node implements everything related to the Kalman filter, namely the Prediction, Matching and Update step. Receives data from the marker identification and IMU.

OptiTrack: The *OptiTrack* node publishes the ground truth data from the Motion Capture System. This is used solely for result comparison.

Rviz: This nodes runs a specific configuration of Rviz (saved by us) which one can visualize in 3D, all the data (ground truth, Kalman output, markers detected, etc).

Error Visualization: This node receives information from the *OptiTrack* and the Kalman estimated and, using matplotlib, plots in real time the absolute angular error and the magnitude of the position error

Note: A launch file as also developed so that all these node do not need to be run separately.

3.2 System Dynamics

Since the aim of the project is to solve a localization problem, the position and the orientation are, inherently, part of the EKF state - X.

Every element of the state of the EKF is with respect to the origin frame.

The state is given by:

$$X = \begin{bmatrix} xyz & v & q & \omega & q_b \end{bmatrix}^T$$

, xyz, represents the position x, y, z; v the velocity v_x, v_y, v_z; q the quaternion q_0, q_1, q_2, q_3; ω the angular rate $\omega_x, \omega_y, \omega_z$; and q_b a bias associated to the quaternions.

Due to the fact that the state is a vector with 20 elements, the simplified representation is given above.

In the prediction step of the EKF, the predicted state $X_{k+1|k}$ is related to the last updated state X_k, by a state transition model. In fact, the quaternions are given by a non linear relation, which explains the need of an Extended Kalman Filter instead of a Kalman Filter.

As explained in [1], quaternion derivation is given by:

$$\frac{dq(t)}{dt} = \frac{1}{2} W(t) q(t)$$

, being $W = [0\omega] = [0, \omega_x, \omega_y, \omega_z]$.

Therefore,

$$\begin{array}{rcl} \frac{dq_0(t)}{dt} & = & -\frac{1}{2}(\omega_x q_1(t) + \omega_y q_2(t) + \omega_z q_3(t)) \\ \frac{dq_1(t)}{dt} & = & \frac{1}{2}(\omega_x q_0(t) - \omega_z q_2(t) + \omega_y q_3(t)) \\ \frac{dq_2(t)}{dt} & = & \frac{1}{2}(\omega_y q_0(t) + \omega_z q_1(t) - \omega_x q_3(t)) \\ \frac{dq_3(t)}{dt} & = & -\frac{1}{2}(\omega_z q_0(t) - \omega_y q_1(t) + \omega_x q_2(t)) \end{array}$$

Integrating the system of equation above each quaternion is obtained:

$$q_0(k+1) = q_0(k) - \frac{\Delta t}{2}(\omega_x q_1(k) + \omega_y q_2(k) + \omega_z q_3(k)) \quad (19)$$

Analogously the same is done for the other 3 quaternions. Since the quaternions depend on either the angular rate and quaternions (both in the state), the expression is non linear. Note that the simplifications in [1] is only valid because the time step is very small. If the time step wete to increase we would need to calculate the new attitude by the quaternion difference (increased computational complexity).

Remembering equation (12), equation (19) is therefore partially derived by each angular rate and quaternion.

The transition model to the position, angular rate and bias are all linear (all is in the world reference frame).

The transition model matrix is thus given by:

$$F = \left[\begin{array}{c|c|c|c} A & O_{6\times4} & O_{6\times3} & O_{6\times4} \\ \hline O_{4\times6} & B & C & -I_{4\times4} \\ \hline O_{3\times6} & O_{3\times4} & I_{3\times3} & O_{3\times4} \\ \hline O_{3\times6} & O_{3\times4} & O_{3\times3} & O_{3\times4} \\ \hline O_{4\times6} & O_{4\times4} & O_{4\times3} & I_{4\times4} \end{array}\right]$$

Being A given by

$$\begin{bmatrix} I_{3\times3} & \Delta t I_{3\times3} \\ O_{3\times3} & I_{3\times3} \end{bmatrix}$$

Block B corresponds to the partial derivative of the quaternions by the quaternions and block C by the angular rate:

$$B = \begin{bmatrix} 1 & \frac{\partial q_0}{\partial q_1} & \frac{\partial q_0}{\partial q_2} & \frac{\partial q_0}{\partial q_3} \\ \frac{\partial q_1}{\partial q_0} & 1 & \frac{\partial q_1}{\partial q_2} & \frac{\partial q_1}{\partial q_3} \\ \frac{\partial q_2}{\partial q_0} & \frac{\partial q_2}{\partial q_1} & 1 & \frac{\partial q_2}{\partial q_3} \\ \frac{\partial q_3}{\partial q_0} & \frac{\partial q_3}{\partial q_1} & \frac{\partial q_3}{\partial q_2} & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} \frac{\partial q_0}{\partial \omega_x} & \frac{\partial q_0}{\partial \omega_y} & \frac{\partial q_0}{\partial \omega_z} \\ \frac{\partial q_1}{\partial \omega_x} & \frac{\partial q_1}{\partial \omega_y} & \frac{\partial q_1}{\partial \omega_z} \\ \frac{\partial q_2}{\partial \omega_x} & \frac{\partial q_2}{\partial \omega_y} & \frac{\partial q_2}{\partial \omega_z} \\ \frac{\partial q_3}{\partial \omega_x} & \frac{\partial q_3}{\partial \omega_y} & \frac{\partial q_3}{\partial \omega_z} \end{bmatrix}$$

Each transition equation for the quaternions is therefore given by:

$$q_n(k+1) = \frac{\partial q_n}{\partial q_0} q_0(k) + \frac{\partial q_n}{\partial q_1} q_1(k) + \frac{\partial q_n}{\partial q_2} q_2(k) + \frac{\partial q_n}{\partial q_2} q_2(k) + \cdots$$

$$\cdots \frac{\partial q_n}{\partial q_3} q_3(k) + \frac{\partial q_n}{\partial \omega_x} \omega_x(k) + \frac{\partial q_n}{\partial \omega_y} \omega_y(k) + \frac{\partial q_n}{\partial \omega_z} \omega_z(k) - q_{b_n}$$

Covariance Definition:

The choice of the quaternion matrix was an iterative process, at which it was noticeable that for different values little difference in the results was made. Thus reasonable values were incorporated in the EKF reflecting the trust we have in the estimate. Therefore, as an example because the position is not estimated from any previous state value (we assume it does not change in the prediction step), its covariance needs to be higher that other measurements that are more precisely calculated.

3.3 Observations Definition

3.3.1 Markers - Pose

The *alvar* markers provide position (x,y,z) and orientation ($q = [q_0 \mathbf{q}]^T$) with respect to each marker frame. Each data from the markers is, rotated to the origin reference frame.

Covariance Definition: To determine the covariance to apply, the error in regard to the *OptiTrack* (ground truth) was plotted, resulting in the Gaussian distributions depicted in figures 2 and 3. We assumed that the position error is independent (covariance matrix is a diagonal). For the quaternion covariance matrix we did not assume independence due to redundancy in quaternion definition.

The covariance matrix for the quaternion measurements is therefore:

3.3.2 IMU- Angular velocity

The IMU provides angular rate ($\omega_x, \omega_y, \omega_z$). Associated to the measurements of the IMU sensor, a bias is included in the state. It is to note that this sensor also provides pressure data and accelerations.

Covariance Definition: For the covariance of the Angular velocity, firstly we went to the data sheets to get the supplier covariance. As results were not satisfactory (the algorithm trusted the gyro data too much, leading to noisy results). For the purpose, the covariance constants were increased until better results were obtained.

3.3.3 IMU - Acceleration

The acceleration is not to be included in the system dynamics due to the fact that the IMU sensor does not provide reliable acceleration data in order to provide position and velocity. The IMU provides quite noisy acceleration reading which will result in a lot of integrated noise to obtain position and velocity. In the following table are presented some errors of the acceleration and the integrated velocity and position with respect to different angle error [2].

TABLE 1: Acceleration, velocity, and position errors that can be expected given different errors in the orientation estimate of the sensor.

Angle error()	Acc. error (m/s/s)	Vel. error (m/s) at 10s	Pos. error (m) at 10s	Pos. error (m) at 1 min
0.1	0.017	0.17	1.7	61.2
0.5	0.086	0.86	8.6	309.6
1	0.256	2.56	25.6	921.6
2	0.342	3.45	34.2	1231.2
3	0.513	5.13	51.3	1846.8
5	0.854	8.54	85.4	3074.4

Covariance Definition: For the covariance of the Angular velocity, we were firstly also driven to the supplier data sheet. Again the results were not very good and adjustment was necessary, increasing the values systematically, minimizing error.

3.3.4 Pressure

In figure **??** and **??** is shown the pressure measured with respect to the altitude of the drone. By calculating the correlation of the altitude with the pressure it is clearly shown that with the given set of measurement it is not possible to relate pressure with altitude (the results show that these two values are uncorrelated). Therefore, it will not be included in the system dynamics.

The observations is therefore given by the position and quaternions provided by the markers, the angular rate provided by the IMU .

In fact, the camera can detect multiple markers or none at all. Therefore the matrix of observation H changes its size depending whether markers are detected or not. In case more than one marker is obtained, it is made an average of the position and the quaternions. It is important to remember that at this point, data from different markers are already in the same frame - the origin frame.

Hence, when markers are detected, the observation is given by

$$Z = \begin{bmatrix} x & q & \omega & q_{b_{acc}} & q_{b_m} \end{bmatrix}^T.$$

When no markers are detected

$$Z = \begin{bmatrix} \omega & q_{b_{acc}} \end{bmatrix}^T.$$

In contrast to the position and quaternions from the marker, the angular rate from the IMU is provided in the body frame. Since the angular rate of the state is in the origin frame, in order to compare the estimation observation ($\hat{Z}$) to the real observations (Z), the angular rate of the state has to be rotated to the body frame.

The quaternions provided by the markers represent the rotation from the origin to the body, which implies that, to rotate the angular rate from the origin frame do the body frame, the equation $\omega_{rotated} = Rot\omega$ must be applied. The matrix R is derived by the definiton of rotation using a quaternions $\omega_{rotated} = q\omega q*$, being q* the conjugate of quaternion q.

Being the rotation matrix given by

$$Rot = \begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2q_1q_2-2q_0q_3 & 2q_1q_3+2q_0q_2 \\ 2q_1q_2+2q_0q_3 & q_0^2-q_1^2+q_2^2-q_3^2 & 2q_2q_3-2q_0q_1 \\ 2q_1q_3-2q_0q_2 & 2q_2q_3+2q_0q_1 & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix}.$$

The last two elements are bias of the quaternions. These are not in fact measured, but used obtained by the equations:

$$q_{b_{acc}} = q(k+1|k) - q_{acc} \qquad q_{b_m} = q(k+1|k) - q$$

, $q(k+1|k)$ corresponds to the quaternions from the predicted state $X_{k+1|k}$ and q the quaternions observed from the markers.

The component q_{acc} is obtained by making the given approximations:

$$\begin{cases} a_x \approx sin(\theta) \\ a_y \approx sin(\phi) \end{cases}$$

From the IMU acceleration data is obtained a rough approximation to pitch(θ) and roll (ϕ) angles. Since there is no analogous approximation to the yaw angle (ψ) by the acceleration, it is simply made the transformation from the quaternions given by the state $X_{k+1|k}$ to the Euler angles. The yaw angle is the result of the third Euler angle obtained.

Afterwards the pitch, roll and yaw angles are transformed from Euler angles to quaternions, and thus obtain q_{acc}.

Since the angular rate is given by a non linear expression (due to the rotation), similarly to what was done in matrix F to the quaternions, in the matrix H has to be included partial derivatives of each angular rate by each element in the rotation matrix, Rot.

The observation matrix when observing markers is given by

$$H = \left[\begin{array}{c|c|c|c|c} I_{3\times3} & O_{3\times3} & O_{3\times4} & O_{3\times3} & O_{3\times4} \\ \hline O_{4\times3} & O_{4\times3} & I_{4\times4} & O_{4\times3} & O_{4\times4} \\ \hline O_{3\times3} & O_{3\times3} & A & B & O_{3\times4} \\ \hline O_{4\times3} & O_{4\times3} & O_{4\times4} & O_{4\times3} & I_{4\times4} \\ \hline O_{4\times3} & O_{4\times3} & O_{4\times4} & O_{4\times3} & I_{4\times4} \end{array}\right]$$

, being A and B given by

$$A = \begin{bmatrix} \frac{\partial\omega_x}{\partial q_0} & \frac{\partial\omega_x}{\partial q_1} & \frac{\partial\omega_x}{\partial q_2} & \frac{\partial\omega_x}{\partial q_3} \\ \frac{\partial\omega_y}{\partial q_0} & \frac{\partial\omega_y}{\partial q_1} & \frac{\partial\omega_y}{\partial q_2} & \frac{\partial\omega_y}{\partial q_3} \\ \frac{\partial\omega_z}{\partial q_0} & \frac{\partial\omega_z}{\partial q_1} & \frac{\partial\omega_z}{\partial q_2} & \frac{\partial\omega_z}{\partial q_3} \end{bmatrix} B = \begin{bmatrix} \frac{\partial\omega_x}{\partial\omega_x} & \frac{\partial\omega_x}{\partial\omega_y} & \frac{\partial\omega_x}{\partial\omega_z} \\ \frac{\partial\omega_y}{\partial\omega_x} & \frac{\partial\omega_y}{\partial\omega_y} & \frac{\partial\omega_y}{\partial\omega_y} \\ \frac{\partial\omega_z}{\partial\omega_x} & \frac{\partial\omega_z}{\partial\omega_y} & \frac{\partial\omega_z}{\partial\omega_z} \end{bmatrix}.$$

When no markers are detected only the lines corresponding to the angular rate and the quaternion bias, using the acceleration, are taken into consideration.

An improvement to be made is instead of averaging the observations from the markers is having Z and H matrices sizes dependent on the number of markers detected. Although no great difference in the results is expected, it is theoretically a better model.

3.4 Matching

During the analysis of the data collected, is was noticeable that if the value of γ (higher than ~ 50) was big enough to allow at every step to proceed with the update, the data would not diverge. However if it was too small ($\gamma \sim 5$) it would diverge simply because too many observations were rejected.

A trade-off was made at $\gamma \sim 20$. In the next section, all the data presented is at $\gamma = 20$

Firstly, the approach after a rejected set of observation by the Matching step was to set the state to the previous updated state, which would then enter at the next prediction. Since the step is $\sim$ to 0.03s, it is not small enough to make such a statement. The second and current approach was to set the updated state as the predicted state, which would enter as an updated state to the next prediction step.

3.5 Task coordination

The EKF cycle has the same frequency as the camera.

This choice was based on the fact that the IMU sensor has a smaller step than the step at which data from the camera is received. Thus, meanwhile each step of the camera, data from the IMU is gathered and when data from the camera is received, IMU data is averaged.

An improvement to be made is to instead of using the frequency of the camera (which is not exactly the same) is using a constant frequency. This avoids problems when occurs some internal error in the camera which will lead to the continuous sum of the IMU data.

4 Experimental Results

Several experiments were conducted in order to test the EKF robustness under a number of circumstances and the AR Alvar marker detection. Tests with varying altitude, drone inclination, omitting markers and temporarily covering the camera were made.

From there we obtained results for position error in the 3 axes (x,y,z and absolute position in the world frame) and the pose of drone (in euler angles for easier interpretation). The results displayed are for two specific tests: temporarily covering the camera to test for robustness in the EKF with different times (the third set has a longer time without marker values). Some other tests regarding altitude limits were also considered. The Figures for reference are named ”second set” and ”third set” since they were respectively the second and third recordings we tested in the field. As we can see from tables 2 and 3 the EKF effectively decreases the mean error of all the pose variables. In addition, the means are overall higher in the third set of data, as a bigger covering time allows for a further drift from the real pose of the drone. This is due to the motion model which assumes constant velocity so naturally the drone stays in the same position during absence of markers. The orientation estimate displays a larger variance as we can see from figures 7 to 9 and from 11 to 13. The yaw angle displays the most erratic progression **??** due to lack of information in the sensors to apply a bias correction (the prediction yaw was assumed in the bias correction). Another aspect to point out is the position in the z axis which for the non-kalman case displays a large deviation from the real position (from analysis of the tables). This is possibly due to the camera calibration, as the tests were conducted by holding the drone and one slight touch in the camera may compromise the calibration. Further calibrations between experiments could be implemented. However the Kalman filter removed this problem.

Due to limitations with drone access and size of the recordings, only small recordings of approximately 30 seconds were made. It can be observed that the error decreases over time, so larger experiments would be helpful to observe the filter's convergence. Overall the EKF improved the pose and results could be further improved by optimizing the covariance matrices of the state and observations.

TABLE 2: Mean errors for the second set of data.

Data	Mean
abs_pos	0.0713 m
abs_pos_nk	0.5007 m
x_pos	0.0660 m
x_pos_nk	0.1540 m
y_pos	0.1302 m
y_pos_nk	0.0977 m
z_pos	0.0115 m
z_pos_nk	0.4337 m
roll	0.7455
roll_nk	3.0556
pitch	-3.9374
pitch_nk	3.0556
yaw	-3.42
yaw_nk	-67.446

TABLE 3: Mean errors for the third set of data.

Data	Mean
abs_pos	0.0448 m
abs_pos_nk	0.8047 m
x_pos	0.0501 m
x_pos_nk	0.1376 m
y_pos	0.0959 m
y_pos_nk	0.3280 m
z_pos	0.0141 m
z_pos_nk	0.6644 m
roll	0.7455
roll_nk	3.0556
pitch	-12.9836
pitch_nk	11.1342
yaw	6.8029
yaw_nk	19.36

5 Conclusions

In this project, an EKF (Extended Kalman Filter) capable of estimating the position and attitude of a drone *CrazyFlie* was implemented. For attitude estimation data from the marker detection, angular velocity and acceleration was fused. For position estimation only the marker information was used, even though alternatives were explored, regarding the pressure (for altitude) and acceleration integration. We consider

that this project was successful regarding that we achieved our goals regarding the successful estimation of the drone's pose.

For future work regarding this project, other alternatives for refining the attitude position determination could be explored. The magnetic field is a measurement that it was not possible to explore deeply (due to time requirements), but potentially could correct the yaw angle when marker data is not available.

6 FIGURES

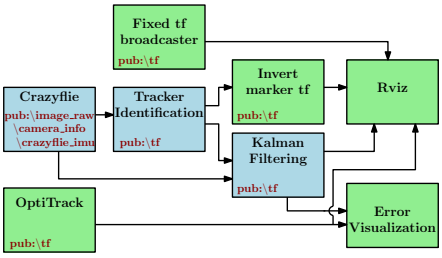

Fig. 1: ROS Architecture Diagram

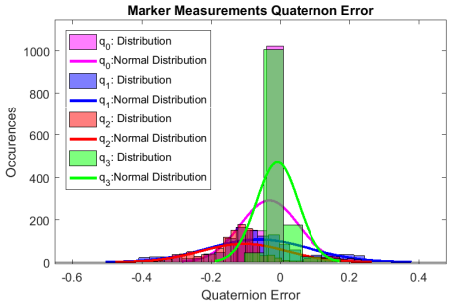

Fig. 2: Distribution of Markers Measurement Quaternion Error.

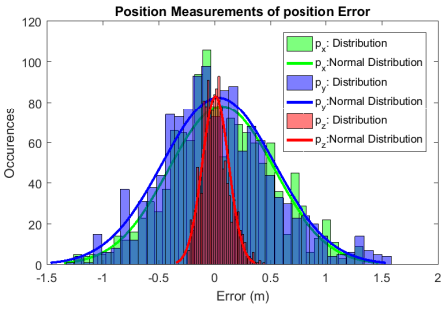

Fig. 3: Distribution of Markers Position Error.

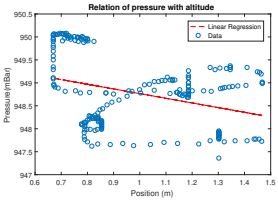

Fig. 4: Pressure from bag 5.

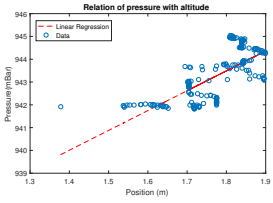

Fig. 5: Pressure from bag 7.

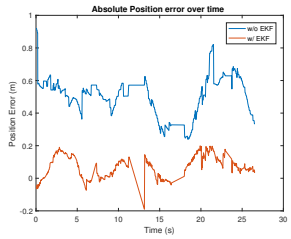

Fig. 6: Error of the absolute position during second set of data

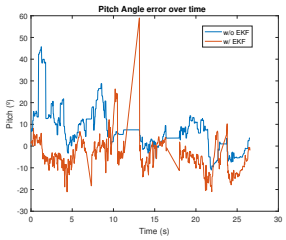

Fig. 7: Error of the pitch during second set of data

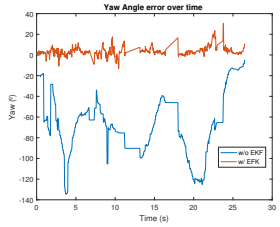

Fig. 8: Error of the yaw angle during second set of data

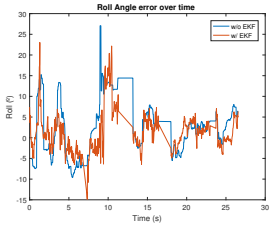

Fig. 9: Error of the roll angle during second set of data

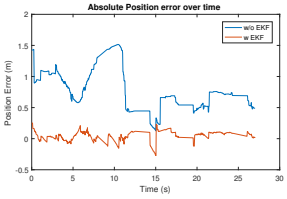

Fig. 10: Error of the absolute position during third set of data

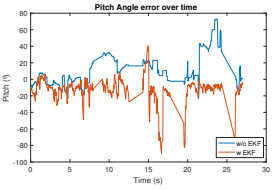

Fig. 11: Error of the pitch during third set of data

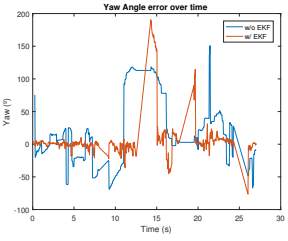

Fig. 12: Error of the yaw angle during third set of data

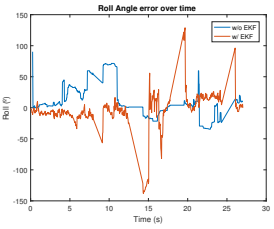

Fig. 13: Error of the roll angle during third set of data

REFERENCES

[1] Yan Bin Jia *Quaternion and Rotation*, 5 Sep. 2017
[2] *CHRobotics*. Accessed in 10/12/2017 in: http://www.chrobotics.com/library/accel-position-velocity