Comparative study for Robotic Airship
State estimation with LPF, EKF and UKF

Apolo S. Marton ∗ Lucas A. C. de O. Nogueira ∗ André R. Fioravanti ∗ Ely C. de Paiva ∗

* School of Mechanical Engineering, University of Campinas, São Paulo, Brazil, 13083-860 apolosm@fem.unicamp.br; lucas.nogueira@fem.unicamp.br; fioravanti@fem.unicamp.br; elypaiva@fem.unicamp.br.

Abstract: This work presents a comparative study between three approaches of pose and velocity estimation for a robotic airship. The first approach is composed of multiple second order Low-pass filters applied to GPS and IMU measured data. The second approach consists of the sensor fusion between GPS, IMU, Barometer and Thermometer with an Extended Kalman Filter (EKF) based in the kinematic equations of motion for a six degrees of freedom (6-DOF) vehicle. Finally, the same 6-DOF equations of motion and sensors are used for the design of an Unscented Kalman Filter (UKF) as a third estimation approach. As result, we obtain greater precision with EKF with a minor advantage over the UKF.

Keywords: Estimation, Filtering, EKF, UKF, ROS, Robotic Airship

1. INTRODUCTION

Airships are known to be extremely nonlinear and under-filled systems. Thus, automatic control, guidance and navigation are difficult tasks to accomplish. Good state estimation techniques are crucial to the development of guidance and control techniques. Initial techniques used only the yaw rate to go-to-waypoint missions as described by Azinheira et al. (2000). However, more sophisticated control strategies such as Sliding-modes as proposed by Vieira et al. (2017) and Dynamic Inversion proposed by Azinheira et al. (2015) require the knowledge of velocities and accelerations. Meanwhile, guidance requires information about attitude, position and airspeed as stated by Moutinho et al. (2016).

The solution for this problem relies in filtering, estimation and sensor fusion methods. Several algorithms have been developed addressing this issue. A traditional algorithm is the Kalman Filter, which can be applied to linear systems. For nonlinear systems, extensions such as the Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) exist.

Kim et al. (2006) showed that such approaches are also useful for treating information redundancies that are common in sensor fusion problems. Nevertheless, Oh (2010) compensates sensor dynamics (such as bias) by adding some modifications to the algorithm.

The choice of a estimation technique plays a crucial role in any autonomous navigation system. Aerospace systems are critical and, usually, do not tolerate failure. As a consequence, it demands comparative studies in simulation environment before embedding the chosen technique(s) in the platform. As an example, in Giannitrapani et al. (2011), the authors presented a performance comparison between EKF and UKF addressing the spacecraft localization problem in simulation environment.

This work presents a comparative study evaluating three solutions for state estimation of a robotic airship (see Figure 1). It is equipped with four vectoring propellers and tail surfaces responsible to guarantee maneuverability during flight. Also, it is instrumented with a pitot tube and a Xsens Mti-G 700 which contains: accelerometer, gyroscopes, GPS, barometer and thermometer.

Firstly, a summary of the robotic airship modeling is presented. Then the sensors modeling take place considering the manufacturer specifications of each device, such as bias and sample frequency. Finally, two classical approaches are compared to a baseline approach that consists of a second-order filter in discrete time. Results obtained in simulation are presented. Then, the performance difference between the three approaches is highlighted. The algorithms were developed in C/C++ using the Robotic Operating System.

Figure 1. Robotic airship performing its first flight
(ROS) developed by Quigley et al. (2009), aiming future tests in the actual platform.

2. AIRSHIP MODEL

This section presents a summary of the mathematical modeling of an airship. For a detailed description, please refer to (Moutinho et al., 2016). The airship nonlinear model can be expressed as a state-space model given by:

\[ \dot{\xi} = T(\xi)x, \]

\[ x = f(x, u, d), \]

where:

- the kinematic states \( \xi = [P_{NED}^T \Phi]^T \) includes the cartesian positions \( P_{NED} = [P_N \ P_E \ P_D]^T \) and angular position \( \Phi = [\phi \ \theta \ \psi]^T \) in the North-East-Down (NED) frame;
- the dynamic states \( x = [V_g^T \ \Omega^T]^T \) includes the linear \( V_g = [u \ v \ w]^T \) and angular velocities \( \Omega = [p \ q \ r]^T \) in the body frame;
- the input vector \( u = [\delta_e \ \delta_a \ \delta_c \ \delta_0]^T \) includes: \( \delta_e, \delta_a \) and \( \delta_c \) which are elevator, aileron and rudder deflection (rad) with saturation limit of \( \pm 0.44 \) rad, respectively; \( \delta_0 \) as the normalized thrusters voltage \( (V/V) \); \( \mu_0 \) as the common vectoring angle of the thrusters (rad);
- and, finally, the disturbance vector \( d \) that includes wind velocities and gust parameters.

The dynamics are based on the Newton-Euler equations including 5 components of forces and moments, namely: \( F_d \) containing the Coriolis and centrifugal force terms, and also the wind-induced forces and moments; \( F_a \) given by aerodynamic forces and moments; \( F_g \) given by propulsion forces and moments; \( F_d \) given by gravity forces and moments, which are function of the difference between the weight and buoyancy forces; and \( F_w \) given by the wind forces and moments. Therefore, the linear and angular accelerations are given by:

\[ f(x, u, d) = M^{-1} \left( F_d + F_a + F_p + F_g + F_w \right). \]

where \( M \) includes mass and inertial coefficients of the airship. These equations are referenced in the body frame centered in the Center of Buoyancy (CB) that is approximately equivalent to the Center of Volume (CV) as shown in Figure 2 as shown by Elfes et al. (2002). The linear and angular positions are updated through kinematic equations (1a).

![Figure 2. Body frame located at the Center of Buoyancy](image)

Table 1. Sensor noise specification \( (w_y) \)

<table>
<thead>
<tr>
<th>Sensor</th>
<th>( \sigma_{RMS} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \phi, \theta, \psi ) (rad)</td>
<td>5.2 \times 10^{-3}</td>
</tr>
<tr>
<td>( \psi ) (rad)</td>
<td>0.1</td>
</tr>
<tr>
<td>( P_N, P_E, P_D ) (m)</td>
<td>2.5</td>
</tr>
<tr>
<td>( V_{pitot} ) (m/s)</td>
<td>0.002</td>
</tr>
<tr>
<td>( V_g ) (m/s)</td>
<td>0.4</td>
</tr>
<tr>
<td>( P_b ) (hPa)</td>
<td>0.01</td>
</tr>
<tr>
<td>( T_h ) (K)</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 2. Sample frequency specification

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Frequency</th>
<th>Sampled data</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMU</td>
<td>100 Hz</td>
<td>( \Phi, \Omega, a )</td>
</tr>
<tr>
<td>GPS</td>
<td>4 Hz</td>
<td>( V_N, P_{NED} )</td>
</tr>
<tr>
<td>Barometer</td>
<td>50 Hz</td>
<td>( P_h )</td>
</tr>
<tr>
<td>Thermometer</td>
<td>1 Hz</td>
<td>( T_h )</td>
</tr>
</tbody>
</table>

Table 3. Sensor bias specification \( (w_b) \)

<table>
<thead>
<tr>
<th>Sensor</th>
<th>( \sigma_{RMS} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gyroscope ( (\circ/s) )</td>
<td>1.3 \times 10^{-4}</td>
</tr>
<tr>
<td>Accelerometer ( (m/s^2) )</td>
<td>2.3 \times 10^{-5}</td>
</tr>
<tr>
<td>Barometer ( (hPa) )</td>
<td>1.4 \times 10^{-5}</td>
</tr>
</tbody>
</table>

3. SENSORS MODELING

The robotic airship is instrumented with: a Pitot Tube developed by Rueda et al. (2017) responsible for measuring airspeed \( (V_{pitot}) \); and a Xsens Mti-G 700 which is composed by a set of sensors providing the following information:

- absolute position in the world \( P_{NED} \) measured by GPS;
- absolute orientation in the world \( \Phi \) (rad), as result of a sensor fusion between Gyroscope and Magnetometer;
- inertial linear velocity \( (V_g) \) in three-axis \( (m/s) \), estimated by the GPS;
- inertial angular velocity \( (\Omega) \) in three-axis \( (m/s) \), estimated by the IMU;
- inertial acceleration in three-axis \( (a = [a_x \ a_y \ a_z]^T) \) \( (m/s^2) \), measured by the Accelerometer;
- atmospheric pressure \( P_b \) (hPa), measured by the barometer;
- and temperature \( T_h \) (K), measured by the thermometer.

Each sensor has sample frequency specified as in the Table 2. Also, each sensor has a generic modeling of a first order Gauss Markov error as shown by Meyer et al. (2012). The general sensor modeling is given by:

\[ y = \dot{y} + b + w_y \]

\[ b = -\frac{1}{\tau} b + w_b \]

where \( b \) is the sensor bias, \( w_y \) is a Gaussian noise for updating bias, \( w_b \) is a Gaussian noise for the measure as specified by the manufacturer in Table 1. \( \dot{y} \) is the true simulated value and \( y \) is the sensor output. The bias is not present in all sensors. For the sensors in which it is modeled, the \( \sigma_{RMS} \) \(^1\) is specified in Table 3. Accelerometer, Barometer and Thermometer requires some specific modeling that are evaluated in the subsections below.

\(^1\) The noise power is given by \( (\sigma_{RMS})^2/(sample \ frequency) \)
3.1 Acceleration measurement

Before using the generic modeling (3), the accelerometer has some extra components of acceleration, namely: centripetal acceleration \(a_c\) and gravitational acceleration \(g\), thus resulting in the following model:

\[
a = V^g + \dot{\Omega} \times O_{\text{sens}} - S_b g + a_c,
\]

where \(O_{\text{sens}}\) is the vector from the CB to the sensor location, \(S_b\) is the rotation matrix from CB to the NED frame and \(a_c = \Omega \times (\Omega \times O_{\text{sens}})\).

3.2 Static pressure measurement

For simulating the static pressure at the current altitude, we use the International Standard Atmosphere (ISA) model as defined by the International Civil Aviation Organization (ICAO), as shown below:

\[
P_h = P_0 \left( \frac{T_h}{T_0 - 0.0065 P_D} \right)^{5.257}
\]

where \(P_h\) is the static pressure at the given altitude, \(T_h\) is temperature in Kelvin, \(P_D\) is the altitude in NED frame and \(P_0\) is the pressure at zero altitude (at the sea level \(P_0 = 101.325\) kPa). Then, the barometer is modeled with the generic sensor model (3).

3.3 Temperature measurement

The temperature in simulation is modeled with a constant rate with the altitude as follows:

\[
T_h = T_0 + 0.0065 P_D
\]

where \(T_0\) is the temperature at zero altitude. Note that \(P_D\) is negative since z-axis points down, thus the temperature tends to fall as the airship goes upper. Then, the thermometer is modeled with the generic sensor model (3).

4. FILTERING AND ESTIMATION

In this section, we present three different strategies for filtering sensor data. The first one is a simple second-order filter that serves as a baseline for comparison purposes. This is followed by an EKF and UKF solution. From now on we use \(^\hat{\cdot}\) notation for estimated states and \(\cdot_0\) for previous measurements.

4.1 Low-pass filter

The most simple approach is a second-order Low-pass filter (LPF). It can be defined as the following transfer function:

\[
\frac{\chi(s)}{Y(s)} = \frac{\omega_f^2}{s^2 + 2\omega_f \zeta s + \omega_f^2}
\]

where \(\chi(s)\) is the estimated state at frequency domain, \(Y(s)\) is the sensor measure at frequency domain, \(\omega_f\) is the cut-off frequency given by the designer and \(\zeta\) is the damping coefficient given by the designer.

For each estimated state, the chosen sensor, cut-off frequency and damping coefficients are given in the Table 4. The cut-off frequency was chosen considering that in GPS measures we have higher errors, thus a lower bandwidth is necessary. Meanwhile, the IMU has lower errors, thus the cut-off frequency can be higher. The choice was based in several simulations until obtain acceptable RMS error.

<table>
<thead>
<tr>
<th>Sensor</th>
<th>(\omega_f) (rad/s)</th>
<th>(\zeta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPS</td>
<td>3.33</td>
<td>0.96</td>
</tr>
<tr>
<td>IMU</td>
<td>10</td>
<td>0.96</td>
</tr>
</tbody>
</table>

Table 4. Configuration for Low-pass filter

4.2 Extended Kalman Filter

The EKF algorithm is well known in literature and can be found in many works. Here we use a package\(^2\) for ROS developed by Moore and Stouch (2014). This package (known as robot_localization) provides the EKF algorithm implementation in C/C++ for a generic robot with 6-DOF (omnidirectional). However, it is not required the robot to have actuation (or state progression) in all directions. In the configuration parameters, it is possible to set the states of interest, such as position, attitude, velocity, angular rate and acceleration.

The filter implementation is able to receive input from different sensors. Each sensor provides information about a subset of state variables. In this work, we are interested in estimate the whole state vector \((P_{\text{NED}}, \Phi, \Omega, V_g, \Omega, a)\). Thus, the sensors are used in measurement update stage as described in Table 5, where 1 is true and 0 is false. From this table, it is possible to observe that only one redundancy is present, between the GPS and Barometer measured vertical speed \((w)\). Also, note that Barometer does not measure altitude, it measures the static pressure. Thus, with the measured pressure and measured temperature, it was computed a measured altitude \(P_D\) through (5). Then the resulting altitude is used as input for the estimators.

Another requirement imposed by the robot_localization package, is to set all sensor information in the East-North-Upper (ENU) frame. However, our sensors are modeled to retrieve data in NED frame. Therefore, this data is converted from NED to ENU frame. For the specific case of acceleration it was used a cascaded scheme where the acceleration data is corrected by removing the centripetal acceleration as in (8) based in the previous filtered angular velocity resulting in the block diagram shown in Figure 3.

\[
\dot{V}_g = a - \Omega_0 \times \left( \Omega_0 \times O_{\text{sens}} \right)
\]

The covariance matrix of the process is a 15 \(\times\) 15 matrix and it is chosen as a diagonal matrix with the weights described in Table 6. These values were chosen based in several simulations performed. It is known that the airship does not have fast response in the states, therefore the process covariance is presumed to be small.

\(^2\) Available in [https://wiki.ros.org/robot_localization](https://wiki.ros.org/robot_localization)
Figure 3. Cascaded acceleration correction

Table 6. Process covariance matrix

<table>
<thead>
<tr>
<th>State</th>
<th>Weight</th>
<th>State</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Phi ) (rad)</td>
<td>0.01</td>
<td>( a ) (m/s²)</td>
<td>0.1</td>
</tr>
<tr>
<td>( \Omega ) (rad/s)</td>
<td>0.1</td>
<td>( u, v ) (m/s)</td>
<td>0.05</td>
</tr>
<tr>
<td>( P_N ) (m)</td>
<td>0.001</td>
<td>( P_E ) (m)</td>
<td>0.001</td>
</tr>
<tr>
<td>( P_D ) (m)</td>
<td>0.01</td>
<td>( w ) (m/s)</td>
<td>0.001</td>
</tr>
</tbody>
</table>

4.3 Unscented Kalman Filter

The Unscented Kalman Filter has the same measurement update stage as in the EKF. Nevertheless, UKF addresses the approximation issues of the EKF. Therefore, in the time update stage, it is elaborated an Unscented transformation in order to calculate the statistics of a random variable which undergoes a nonlinear transformation as shown by Wan and Merwe (2000).

Our UKF implementation uses the same ROS package (robot_localization). The package also contains the UKF algorithm for a robot with 6-DOF based in the kinematic equations. Therefore, the same data conversions and configuration parameters used for EKF were applied. Other configuration and implementation details are available in a GitHub repository³.

5. RESULTS

In order to establish a comparison between the three approaches, a simulation was performed using ideal state feedback for a controller developed by Azinheira et al. (2015). All three estimators are working at approximately 32Hz. Also, along the path is considered wind flowing from the North to South with 0° of incidence. The simulation was performed in Simulink/MATLAB and a data set was generated for the performance evaluation of each estimation approach.

The results of estimation and ground truth data may be visualized in a 3D environment available as a tool for ROS called RViz (see Figure 4). This environment is configured to receive the ground truth data through ROS communication as well as the estimation results generated by the filters. The 3D visualization shows both the airship pose (position and orientation) of the ground truth (with transparent body) and estimated pose (with solid body). In the estimated positions are added 3D ellipsoids which represents the covariance error of estimation. For each approach a video of the simulation was recorded which are available for visualization in you. In EKF and you. In UKF.

In the interval (a), EKF and LPF position estimation is smoother than the UKF’s, whilst the LPF has lower absolute error. However, in the interval (b) EKF is smoother and also has a lower absolute error. Thus, EKF appears to be smoother and also has less error when considering the whole mission. It can be confirmed by the RMS of position errors shown in Table 7 (in bold the lowest RMS values). The euler angles are not shown in this table, because Xsens already provides a good estimation resulting from a sensor fusion between gyroscope and magnetometer, thus the graphical difference as the error RMS are negligible.

In the interval (a), EKF and LPF position estimation is smoother than the UKF’s, whilst the LPF has lower absolute error. However, in the interval (b) EKF is smoother and also has a lower absolute error. Thus, EKF appears to be smoother and also has less error when considering the whole mission. It can be confirmed by the RMS of position errors shown in Table 7 (in bold the lowest RMS values). The euler angles are not shown in this table, because Xsens already provides a good estimation resulting from a sensor fusion between gyroscope and magnetometer, thus the graphical difference as the error RMS are negligible.

In the interval (a), EKF and LPF position estimation is smoother than the UKF’s, whilst the LPF has lower absolute error. However, in the interval (b) EKF is smoother and also has a lower absolute error. Thus, EKF appears to be smoother and also has less error when considering the whole mission. It can be confirmed by the RMS of position errors shown in Table 7 (in bold the lowest RMS values). The euler angles are not shown in this table, because Xsens already provides a good estimation resulting from a sensor fusion between gyroscope and magnetometer, thus the graphical difference as the error RMS are negligible.

The most relevant states for the control loop are longitudinal (\( u \)), vertical (\( w \)) and angular velocities (\( \Omega \)). For the first interval shown before, the estimation of vertical velocity, longitudinal velocity, and angular in z-axis (\( r \)) are depicted in Figure 8. By these figures, we can conclude that LPF provides good filtering. However, it inserts a significant delay. On the other hand, UKF provides a faster response.

3 https://github.com/leve-fem/airship_estimator.git

with a higher error. Meanwhile, EKF has a time response on par with the UKF while maintaining lower error levels. Thus, for the airship model, the EKF shows a better performance in comparison to the two other estimation approaches.

6. CONCLUSIONS

The presented approaches have shown an acceptable filtering and estimation of the whole state vector, with sufficient precision to facilitate the control techniques developed

Figure 6. Estimated trajectory

Figure 7. Estimated position (a) at high ground speed and (b) low ground speed

Table 7. RMS of estimation error

<table>
<thead>
<tr>
<th>State</th>
<th>LPF</th>
<th>EKF</th>
<th>UKF</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_N$ (m)</td>
<td>3.1329</td>
<td>2.4118</td>
<td>3.3654</td>
</tr>
<tr>
<td>$P_E$ (m)</td>
<td>3.3633</td>
<td>2.6114</td>
<td>3.3945</td>
</tr>
<tr>
<td>$P_D$ (m)</td>
<td>3.8960</td>
<td>2.0382</td>
<td>2.1521</td>
</tr>
<tr>
<td>$v$ (m/s)</td>
<td>0.4215</td>
<td>0.3186</td>
<td>0.3724</td>
</tr>
<tr>
<td>$w$ (m/s)</td>
<td>0.4713</td>
<td>0.2948</td>
<td>0.4267</td>
</tr>
<tr>
<td>$p$ (rad/s)</td>
<td>0.0644</td>
<td>0.0685</td>
<td>0.0664</td>
</tr>
<tr>
<td>$q$ (rad/s)</td>
<td>0.0355</td>
<td>0.0359</td>
<td>0.0366</td>
</tr>
<tr>
<td>$r$ (rad/s)</td>
<td>0.0373</td>
<td>0.2871</td>
<td>0.2915</td>
</tr>
</tbody>
</table>

It is important to highlight that (as we explained before) we have chosen the covariance parameters for the UKF searching for the best results possible. Then the same parameters were applied to EKF, without any additional optimization. Consequently, we initially expected to achieve a better performance with UKF. However, from the results presented in the previous sections, one can observe that the EKF clearly had a better performance. It suggests that, for this specific platform and under the same conditions, the EKF may be a more suitable choice. These are interesting results, since in Giannitrapani et al. (2011) the UKF presented a slightly better performance in terms of average localization accuracy for their specific case of a spacecraft model.

The algorithms were developed in C/C++ using ROS in order to embed the estimator in the platform in future works. Also, error tolerance analyses and closed-loop simulations will take place in future works.

ACKNOWLEDGMENTS

The authors are grateful to Erasmus Mundus, CNPq, CAPES and FAPESP for the financial support. Further acknowledgments to CTI Renato Archer, ACCES-Lab and LEVE.

REFERENCES


DOI: 10.17648/sbai-2019-111114


