UNIVERSITÀ DEGLI STUDI DI CATANIA
FACOLTÀ DI INGEGNERIA ELETTRICA ELETTRONICA E INFORMATICA

XXVII cycle Ph.D. in System Engineering

Davide Spina

Multi-Data Sensor Fusion for Robotics Applications

Ph.D. Thesis

Tutor: Prof. G. Muscato
Coordinator: Prof. L. Fortuna

2014

Preface

Thanks to faster and better integrated processing units, as well as increasingly miniaturized and precise sensors, Robotics is experiencing a period of significant development.

New technology is making robots increasingly more autonomous. Data from different sensors can be fused and processed in real time by on-board processing units in order to make the robot perform either more or less complicated tasks.

Unmanned Ground Vehicles (UGV) and Unmanned Aerial Vehicles (UAV) equipped with sensors and microprocessors/microcontrollers can be remotely piloted. They can also move autonomously with the help of different navigation and localization methods.

Power electronics is contributing to a transformation in many fields such as robotics, automotive and consumer devices.

Some new applications of power electronics are discussed in the first chapter of this thesis. One of these describes how new power electronics devices allow the use of distributed instead of centralized control in industrial robotics.

Multi-sensor data fusion theory is presented in the second chapter; the Kalman Filter and Particle Filter are described.

Inertial sensors and magnetometers are introduced in the third chapter with the description of a calibration procedure for each sensor.

Two new methods of multi-sensor data fusion, based on low-cost Micro Electro-Mechanical Systems (MEMS) Inertial Sensors, to estimate joint angles of industrial manipulators are described in the fourth chapter. The results of two experimental tests are also presented to evaluate and to compare the performances of the two methods.

A method to estimate the attitude and heading of an UAV is described in the fifth chapter. In order to check the performance of the developed method, at the end of the fifth chapter, a comparison test with a high accuracy system is presented.

A localization algorithm for a wheeled UGV equipped with a GPS and an inertial platform is described in the sixth chapter. Simulation tests and experimental results are also presented at the end of the sixth chapter.

Summary

Preface 2
Summary 4
1 Power Electronics 8
1.1 Introduction 8
1.2 Automotive applications 9
1.3 Mechatronics applications 10
1.4 Power Electronics for Robotics Application 13
2 Multi-sensor data fusion 15
2.1 Introduction 15
2.2 The Kalman Filter 17
2.3 The Extended Kalman Filter 19
2.4 Particle Filter 20
3 Inertial Sensors 23
3.1 Introduction 23
3.2 Gyroscopes 23
3.2.1 CALIBRATION 26
3.3 Accelerometers 27
3.3.1 STATIC CONDITIONS 28
3.3.2 DYNAMIC CONDITIONS 30
3.3.3 CALIBRATION 31
3.4 Magnetometers 33
3.4.1 CALIBRATION 35
4 Industrial Robotics 40

4.1 Introduction 40
4.2 Model of the Manipulator 43
4.3 Methodology 48
4.4 Kalman Filter 51
4.5 Particle Filter 53
4.6 Experimental Setup 55
4.7 Experimental Results 56
4.7.1 First Experiment 57
4.7.2 Second Experiment 63
4.8 Dynamic Acceleration 70
5 UAV Attitude Heading Reference System 77
5.1 Introduction 77
5.2 CANaerospace 77
5.2.1 Main features 78
5.2.2 Frame Format 79
5.2.3 Frame NSH 81
5.2.4 Frame NOD 90
5.2.5 Frame DSD 93
5.2.6 Errors Management 95
5.3 EKF UAV 95
5.3.1 QUATERNIONS 97
5.3.2 UAV MODEL AND STATE PREDICTION 102
5.3.3 STATE CORRECTION 104
5.4 Chebyshev Filtering 106
5.5 Dynamic compensation 109

5.6 Firmware Development and Optimization 110
5.6.1 Firmware Block Scheme 110
5.7 Results 113
6 UGV Localization 117
6.1 Introduction 117
6.2 The model of the robot 117
6.3 The Localization algorithm 120
6.4 GPS Accuracy and Precision 123
6.4.1 Distance Root Mean Squared (DRMS) 124
6.4.2 Circular Error Probability (CEP) 125
6.4.3 Dilution of Precision (DOP) 126
6.4.4 Variance of the GPS noise 126
6.5 Results 129
7 Conclusion 133
8 Bibliography 135

1 Power Electronics

1.1 Introduction

In recent years, energy saving has become an increasingly important topic for many reasons such as:

- The economic crisis has led the governments of many countries to adopt energy saving policies with the view to becoming self-sufficient.
- Due to of the economic crisis people are more sensitive about energy saving in order to reduce the costs.
- People have become aware of the fact that main energy sources like oil, gas and coal are limited and often pollutant.
- International agreements in order to reduce pollution and global warming.

In this context, technologies that allow energy savings have become more widespread.

Nowadays is easy to see solar panels in the roof of a house, or being overtaken by an electric car.

At the basis of most of the technologies that lead to energy savings that are spreading, there is the power electronics.

Power electronics is one of the technologies that has spread lately even thanks to the big progresses of the engineering science, and that is contributing most to energy savings.

For example, a cooling system that uses a power electronic system, consumes 30% less energy than a normal one.

Power electronics is a branch of electronics that is responsible for managing the electric energy instead of dealing with the information.

1.2 Automotive applications

In the automotive field, examples of power electronic applications are the drive by wire and the kinetic energy recovery system (KERS).

The drive-by-wire technology in the automotive industry refers to the use of electrical or electromechanical systems for performing vehicle functions, like steering or braking, instead of hydraulic or mechanical systems.

Substituting mechanical parts with electronics systems, makes cars lighter but even more complex.

In a modern car it is possible to find tens of microcontrollers and several electronic control units (ECUs) that control all the electronics devices and communicate to each other in order to ensure a safe and enjoyable drive.

Figure 1: drive-by-wire

The kinetic energy recovery system (KERS) is an automotive system for recovering the kinetic energy of moving vehicle during braking.

To accomplish this goal is necessary a sophisticated control system that, for example, allows the power electronics devices to store this energy in a battery or in a super-capacitor.

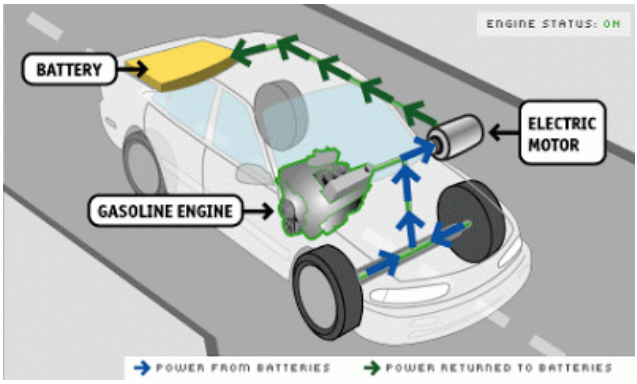

Figure 2: kers

1.3 Mechatronics applications

In the mechatronics field, an application of the power electronics is the distributed control.

A distributed control system (DCS) is a control system wherein control elements are distributed throughout the system.

Figure 3: industrial robot

Typical mechatronics systems, use a single controller at a central location.

In an industrial manipulator with centralized control, the cabinet contains all the hardware for the control (such as the motor drives), and it is large, expensive and inflexible.

Figure 4: cabinet of an industrial robot

Figure 5: centralized control

An example of power electronics device for distributed control is represented by the SPIMD20 of STMicroelectronics.

Figure 6: distributed solution

SPIMD20 is an 2 kW integrated motor drive for decentralized control; it can be installed directly on the motor case [1].

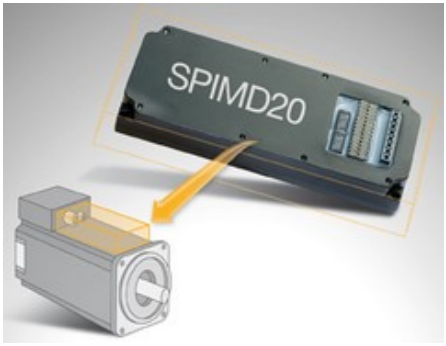

Figure 7: SPIMD20 integrated motor drive

With this module it is possible to obtain the following advantages:

- Minimizing cabling systems and installation cost;
- Control panel optimization from rack to the equipment;
- Flexibility of automation architecture;

SPIMD20 is a complete compact motor drive solution in a single, robust and reliable module, including full bridge IGBT inverter, control unit and communication interface based on real-time industrial Ethernet.

It is ideal for applications with high number of 3-phase Brushless motors and real-time constraints:

- Packaging machines;
- Process machines;
- Multi-axial controls;
- Robotics.

1.4 Power Electronics for Robotics Application

As described in the previous paragraph, the development of power electronics has made many interesting scenarios, such as industrial robotics and mobile robotics, possible in Robotics application.

New families of inverter, for example, allow the use of decentralized control. Thanks to the small dimensions of new inverter, indeed, it is possible to move the controllers of a robot directly on its axes, with advantages in terms of cable costs and flexibility.

Decentralized control in industrial robotics needs the knowledge of the state of each link; this is possible by means multi-sensor data fusion techniques that fuse together measurements coming from different sensors such as accelerometers, gyroscopes and encoders.

Power electronics devices, like inverter, can be very useful also in mobile robotics application, such as the control of unmanned ground vehicles (UGV) or unmanned aerial vehicles. In order to control the movements of an UGV, inverters allow the transformation of the control commands into the appropriate signals for the motors. Therefore, it is very important to know the right position of the motors axes. By using multi-sensor data fusion techniques, is also possible to localize the UGV, as is described in the sixth chapter of this thesis.

2 Multi-sensor data fusion

2.1 Introduction

The knowledge of the state of a dynamic system like an industrial robot is necessary in order to understand its dynamic behavior and to solve many control problems.

Theoretically the dynamics of a linear system can be determined from a complete knowledge of its mathematical model.

The discrete-time model of a linear system is:

$$\boldsymbol{x_k = A\, x_{k-1} + B\, u_k} \qquad (\mathbf{1})$$

Where $x_k \in \Re^n$ is the state vector of the system at the current step *k* , x_{k-1} is the state of the system at the previous step *k-1*, $u_k \in \Re^l$ is the input of the system, $A \in \Re^{nxn}$ is the state matrix that relates the state at the previous time step to the state at the current step, $B \in \Re^{nxl}$is the output matrix that relates the optional control input *u* to the state *x*.

However, unfortunately, the model of the system often is not enough accurate and is subject to the process noise *w*.

$$\boldsymbol{x_k = A\, x_{k-1} + B\, u_k + w_{k-1}} \qquad (\mathbf{2})$$

Another way to determine the state of the system could be by knowing the relation between the sensor measurements $z_k \in \Re^m$ and the state:

$$\boldsymbol{z_k = H\, x_k} \qquad (\mathbf{3})$$

Where the matrix $H \in \Re^{mxn}$ relates the state x_k to the measurement z_k.

But usually the sensor measurements are noisy and the relation with the state is inaccurate:

$$\boldsymbol{z_k = H\, x_k + v_k} \tag{4}$$

where v is the measurement noise.

Multi-sensor data fusion:

- allows to overcome these problems in order to get a better estimation of the state of the system, indeed it permits to exploit the advantages and to compensate the disadvantages of different sensors.
- is the process of combining observations from a number of different sensors to provide a robust and complete description of an environment or process of interest.
- finds wide application in many areas of robotics such as object recognition, environment mapping, and localization.
- method based on the Recursive Bayesian Filtering techniques, allows to integrate measurements from inertial sensors with measurements from different kind of sensors.

Recursive Bayesian Filtering techniques, are widely used for estimating the state of a dynamic system, because allow to overcome problems caused by uncertainty and noise.

Given the state space model of a dynamic system, composed by the state transition function and the measurement function,

$$\begin{cases} \boldsymbol{x_k = f(\, x_{k-1},\, u_k, w_{k-1})} & \textbf{(5)} \\ \boldsymbol{z_k = h(\, x_k, v_k)} & \textbf{(6)} \end{cases}$$

where f and h can be both linear or non-linear, the goal of the Recursive Bayesian Filtering is to estimate the evolution of the system state using all the measurements collected until the current time [2].

As the state of the system is a random variable, all the information about the state provided by the measurements are contained in the posterior probability density function (pdf) *p(x)*, therefore the Recursive Bayesian Filtering has to provide an estimation of the posterior pdf.

$$\boldsymbol{p(x) = p(x_{0:k}|z_{1:k})} \quad \textbf{(7)}$$

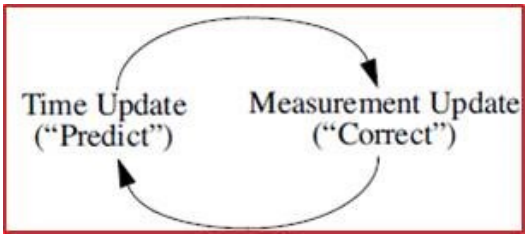

Figure 8: Steps of Recursive Bayesian filter

2.2 The Kalman Filter

In the case in which both the transition and the measurement functions are linear and either the process and the measurement noise are Gaussian and additive, the optimal solution is given by the Kalman Filter (KF) [3].

The Kalman filter is essentially a set of mathematical equations that implement a predictor-corrector type estimator that is optimal in the sense that it minimizes the estimated error covariance when the previously cited conditions are met [3].

Given the linear dynamic state-space model of a system:

$$\begin{cases} \boldsymbol{x_k = A\,x_{k-1} + B\,u_k + w_{k-1}} & \textbf{(8)} \\ \boldsymbol{z_k = H\,x_k + v_k} & \textbf{(9)} \end{cases}$$

where the process and measurement noise w_k and v_k have a normal probability distributions:

$$\boldsymbol{p(w) \sim N(0, Q)} \quad \textbf{(10)}$$

$$\boldsymbol{p(v) \sim N(0, R)} \quad \textbf{(11)}$$

the Kalman filter provides an *a posteriori* state estimate $\boldsymbol{\hat{x}_k}$ as a linear combination of an *a priori* estimate $\boldsymbol{\hat{x}_k^-}$ and a weighted difference between an actual measurement $\boldsymbol{z_k}$ and a measurement prediction $\boldsymbol{H\,\hat{x}_k^-}$:

$$\boldsymbol{\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\,\hat{x}_k^-)} \quad \textbf{(12)}$$

The difference $\boldsymbol{(z_k - H\,\hat{x}_k^-)}$ in equation 12 is called *measurement innovation*, or *residual*. The residual reflects the discrepancy between the predicted and the actual measurements . A residual of zero means that the two are in complete agreement.

$\boldsymbol{K_k}$ is the *Kalman gain* or *blending factor* that minimizes the covariance $\boldsymbol{P_k}$ of the posteriori error $\boldsymbol{e_k}$:

$$\boldsymbol{P_k = E\left[e_k e_k^T\right]} \quad \textbf{(13)}$$

$$\boldsymbol{e_k = x_k - \hat{x}_k} \quad \textbf{(14)}$$

One form of *K* that minimizes the posteriori error covariance is given by:

$$\boldsymbol{K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}} \quad \textbf{(15)}$$

Finally, the equations of the Kalman filter are:

Table 1

	Initial estimates for $\widehat{\boldsymbol{x}}_{\boldsymbol{k-1}}$and $\boldsymbol{P}_{\boldsymbol{k-1}}$
Time Update ("Predict")	
Project the state ahead	$\widehat{\boldsymbol{x}}_{\boldsymbol{k}}^{-} = \boldsymbol{A}\,\widehat{\boldsymbol{x}}_{\boldsymbol{k-1}} + \boldsymbol{B}\,\boldsymbol{u}_{\boldsymbol{k}}$
Project the error covariance ahead	$\boldsymbol{P}_{\boldsymbol{k}}^{-} = \boldsymbol{A}\boldsymbol{P}_{\boldsymbol{k-1}}\boldsymbol{A}^{\boldsymbol{T}} + \boldsymbol{Q}$
Measurement Update ("Correct")	
Compute the Kalman gain	$\boldsymbol{K}_{\boldsymbol{k}} = \boldsymbol{P}_{\boldsymbol{k}}^{-}\boldsymbol{H}^{\boldsymbol{T}}(\boldsymbol{H}\boldsymbol{P}_{\boldsymbol{k}}^{-}\boldsymbol{H}^{\boldsymbol{T}} + \boldsymbol{R})^{-\boldsymbol{1}}$
Update estimate with measurement z_k	$\widehat{\boldsymbol{x}}_{\boldsymbol{k}} = \widehat{\boldsymbol{x}}_{\boldsymbol{k}}^{-} + \boldsymbol{K}_{\boldsymbol{k}}(\boldsymbol{z}_{\boldsymbol{k}} - \boldsymbol{H}\,\widehat{\boldsymbol{x}}_{\boldsymbol{k}}^{-})$
Update the error covariance	$\boldsymbol{P}_{\boldsymbol{k}} = (\boldsymbol{I} - \boldsymbol{K}_{\boldsymbol{k}}\boldsymbol{H})\,\boldsymbol{P}_{\boldsymbol{k}}^{-}$

2.3 The Extended Kalman Filter

In the real world, the previously mentioned hypotheses are rarely encountered and therefore you have to cope with nonlinearities.

Generally, for nonlinear filtering, no exact optimal solutions can be obtained, however, under the assumption that the process and the measurement noise are Gaussian and additive, a classical method to solve the nonlinear filtering problem is the Extended Kalman Filter (EKF). In the EKF a linearization around the current estimate is applied to the system and the conventional Kalman filtering technique is further employed.

Let us consider a process governed by the non-linear equations:

$$\begin{cases} \boldsymbol{x}_{\boldsymbol{k}} = \boldsymbol{f}(\,\boldsymbol{x}_{\boldsymbol{k-1}},\,\boldsymbol{u}_{\boldsymbol{k}},\boldsymbol{w}_{\boldsymbol{k-1}}) & \textbf{(16)} \\ \boldsymbol{z}_{\boldsymbol{k}} = \boldsymbol{h}(\,\boldsymbol{x}_{\boldsymbol{k}},\boldsymbol{v}_{\boldsymbol{k}}) & \textbf{(17)} \end{cases}$$

with *f* and *h* non-linear functions, to estimate the state we need first to linearize the system around the current estimate by means of the following Jacobians [3]:

$$W_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}}(\hat{x}_k, u_k, 0) \quad (17)$$

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}_k, u_k, 0) \quad (18)$$

$$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}(\hat{x}_k, 0) \quad (19)$$

$$V_{[i,j]} = \frac{\partial h_{[i]}}{\partial v_{[j]}}(\hat{x}_k, 0) \quad (20)$$

Finally, the equations of the extended Kalman filter are:

Table 2

	Initial estimates for $\hat{\boldsymbol{x}}_{\boldsymbol{k-1}}$ and $\boldsymbol{P_{k-1}}$
Time Update ("Predict")	
Project the state ahead	$\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{\boldsymbol{k-1}}, \boldsymbol{u_k}, \boldsymbol{0})$
Project the error covariance ahead	$\boldsymbol{P_k^-} = \boldsymbol{A_k P_{k-1} A_k^T} + \boldsymbol{W_k Q_{k-1} W_k^T}$
Measurement Update ("Correct")	
Compute the Kalman gain	$\boldsymbol{K_k} = \boldsymbol{P_k^- H^T (H P_k^- H^T} + \boldsymbol{V_k R_{k-1} V_k^T)^{-1}}$
Update estimate with measurement z_k	$\hat{\boldsymbol{x}}_{\boldsymbol{k}} = \hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-} + \boldsymbol{K_k}(\boldsymbol{z_k} - \boldsymbol{h}(\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-}, \boldsymbol{0}))$
Update the error covariance	$\boldsymbol{P_k} = (\boldsymbol{I} - \boldsymbol{K_k H})\, \boldsymbol{P_k^-}$

2.4 Particle Filter

Because EKF always approximates the posterior pdf as a Gaussian, it works well for some types of nonlinear problems, but it may provide a poor performance in some cases when the true posterior pdf is non-Gaussian (e.g. heavily skewed or multi-modal).

When the model is highly nonlinear or the noise is non-Gaussian, KF and its variants are no longer suited for estimating the state of a dynamic system.

The alternative to the Gaussian filters to which KF belong, is represented by the nonparametric filters.

The main difference between Gaussian filters and nonparametric filters is the different representation of the posterior pdf. The first class represents this pdf with a fixed function, the Gaussian function, while, on the other hand, the nonparametric filters represents the posterior pdf through a finite number of values, each of which roughly corresponds to a portion of the state space.

Among the nonparametric filters, the PF represent the posterior pdf through a high number of samples or particles. The aim of PF, as the *Recursive Bayesian Filtering* technique, is to estimate the posterior pdf but, since it is generally impossible to sample from it, the *Monte Carlo Importance Sampling* method is used [4]. The key idea of the *Monte Carlo Importance Sampling* method is to generate samples from a different *pdf* with particular characteristics called *importance function* instead of sampling from the posterior pdf directly and for each generated sample use a weighting factor to account for the mismatch between functions.

Therefore, ultimately the PF represents the required posterior pdf by a set of random samples or particles, generated as aforementioned, with associated weights.

In order to avoid the divergence of the estimation, a resampling step is executed by replicating the samples with high importance weights and removing samples with low weights.

Let us consider a process governed by the non-linear equations:

$$\begin{cases} \boldsymbol{x_k = f(\, x_{k-1},\, u_k, w_{k-1})} & \textbf{(21)} \\ \boldsymbol{z_k = h(\, x_k, v_k)} & \textbf{(22)} \end{cases}$$

with *f* and *h* non-linear functions, the equations of the particle filter are:

Table 3

$for\ i = 1{:}\ N_s$
$x_k^i \sim q(x_k \mid x_{k-1}^i, z_k) = p(x_k \mid x_{k-1}^i)$
$w_k^i = w_{k-1}^i\ p_v(z_k \mid x_k^i)$
End for
Normalize weights
Resample
$\hat{\boldsymbol{x}}_{\boldsymbol{k}} = \hat{\boldsymbol{x}}_{\boldsymbol{k-1}} + \sum_{i=1}^{N_s} w_k^i\ x_k^i$

A comparison between the equations of the extended Kalman filter and the equations of the particle filter is shown in the following figure:

	EKF	PF
Prediction	$x_k = f(x_{k-1}, u_k, 0)$ ***P* = Error Covariance prediction**	$x_k^i \sim p(x_k \mid x_{k-1}^i)$
Correction	$K_k = Kalman\ Gain$(**P**, **Jacobians**) $\hat{x}_k = \hat{x}_{k-1} + K_k(z_k - h(x_k))$	$w_k^i = w_{k-1}^i\ p_v(z_k - h(x_k^i))$ $\widehat{x_k} = \hat{x}_{k-1} + \sum_{i=1}^{N_s} w_k^i\ x_k^i$

Figure 9: comparison between EKF and PF

3 Inertial Sensors

3.1 Introduction

Inertial sensors are particular sensors that exploit the inertia of a mass contained inside them, to measure angular rates (gyroscopes) or linear accelerations (accelerometers).

In recent years the performances of MEMS inertial sensors have improved and their cost has decreased and thus they have been used successfully in many applications such as virtual reality, autonomous vehicle navigation and robotics.

As described in the following pages of this chapter, the accelerometers and gyroscopes measures are not sufficient for those applications where is necessary to have complete information about the attitude and the orientation of a rigid body like a robot.

In this chapter, indeed, the magnetometers are described as well.

Moreover, in addition to the main features and the principles of operation, for each sensor a calibration procedure is described.

For the calibration of the inertial sensors, a Kuka robot manipulator was used as a precision instrument to move the gyroscopes at known angular velocities and to let accelerometers assume known orientations.

3.2 Gyroscopes

A Gyroscope is a device that measures the angular velocity.

The first gyroscope was a mechanical device (Fig.10) composed by a rotor and a Cardan suspension (gimbal) that, due to the law of conservation of angular momentum, when the rotor rotates at a fixed angular velocity, tends to maintain its rotation axis (spin axis) oriented in a fixed direction.

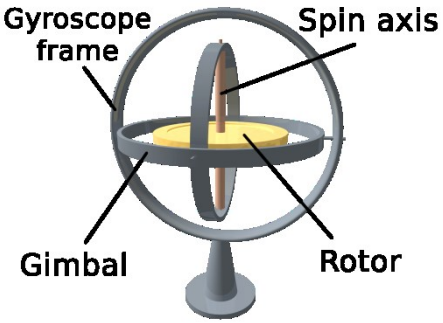

Figure 10: gyroscope

Each torque applied to the spin axis, due to the rotation of the gyroscope frame, causes a momentum orthogonal to the torque. The measure of the action of a servomotor that maintains the spin axes parallel to itself, provides the measure of the angular velocity of the initial rotation.

Today in precise application the optical gyroscopes are widely diffused and in particular the optical fiber gyroscopes that have high precision, are very simple and are not expensive since not require high precision mechanical parts.

Inexpensive vibrating structure gyroscopes manufactured with MEMS technology have become widely available. Internally, MEMS gyroscopes use lithographically constructed versions of one or more of the following mechanisms: piezoelectric, tuning forks, vibrating wheels, or resonant solids of various designs. MEMS gyroscopes are used in automotive roll-over prevention and airbag systems, image stabilization, and have many other potential applications.

The angular velocity measured by a gyroscope ω_m $[rad/sec]$ differs from the expected one ω, due to the not unitary scale factors s, the constant component of the bias b and a random process $n(t)$ that models the noise and the variable component of the bias.

$$\boldsymbol{\omega_m = s\omega + b + n(t)} \qquad \textbf{(23)}$$

The performance of the different types of gyroscopes are usually described by the stability of the bias and by the stability of the scale factor.

Nowadays gyroscopes are mainly used to get the angular velocity and the orientation in applications such as compasses, aircraft, mobile robots, computer pointing devices, consumer electronics, etc.

By integrating, with a processing unit, the angular velocity measured by a gyroscope, it is possible to get the tilt angle of the device.

Therefore, in addition to the angular velocity, gyroscopes allow to get information about the orientation of an object like for example an aircraft (fig.11):

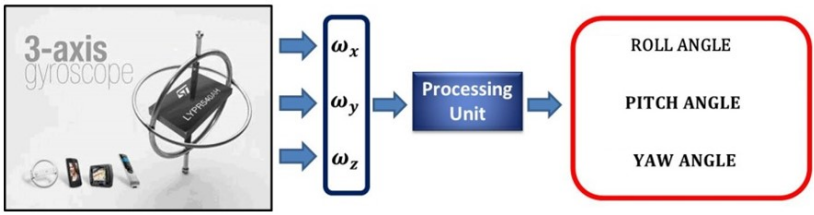

Figure 11: Euler angles from gyroscope measures

However the integration of the measured angular velocity causes an accumulation of the noise over time that brings to the drift problem.

To contrast the drift problem of the gyroscopes measurements integration, it can be occasionally subtracted from the measurement or, as in this work is explained, it is possible to use a multi-sensor fusion method.

3.2.1 CALIBRATION

To convert the gyroscope's raw measurements to meaningful angular velocities a calibration procedure could be necessary, in particular for highly sensitive applications [5].

Considering a triaxial gyroscope, the relationship between the raw measurements and the meaningful angular velocity, can be written in the following way:

$$\begin{bmatrix} R_x \\ R_y \\ R_z \end{bmatrix} = G\_m_{3x3} \begin{bmatrix} SC_x & 0 & 0 \\ 0 & SC_y & 0 \\ 0 & 0 & SC_z \end{bmatrix} * \begin{bmatrix} R'_x - R_{x0} \\ R'_y - R_{y0} \\ R'_z - R_{z0} \end{bmatrix}$$
$$= \begin{bmatrix} G_{11} & G_{12} & G_{13} \\ G_{21} & G_{22} & G_{23} \\ G_{31} & G_{32} & G_{33} \end{bmatrix} * \begin{bmatrix} R'_x - G_{10} \\ R'_y - G_{20} \\ R'_z - G_{30} \end{bmatrix}$$

Figure 12: relationship between raw and calibrated measures

Where *Rx, Ry, Rz* are the calibrated angular velocity of each axis, *G_m* is the misalignment matrix between the gyro sensing axes and the device body axes, *SCx, SCy, SCz* are the scale factors caused by the mismatch of the sensitivity of each axis, *Rx', Ry', Rz'* are the raw measurement of the gyroscope and R_{x0}, R_{y0}, R_{z0} are the zero-rate level or bias for each axis.

One possible calibration procedure consists in the calculation of 12 parameters ($G_{11}…G_{30}$) by means of the least square method.

In particular the steps of the procedure are:

- Measure the zero-rate level R_{x0}, R_{y0}, R_{z0} of each axis;
- Rotate the board around each axis at different known angular velocity and collect the measurements;
- Construct the known applied angular velocity matrix (Y);
- Construct the gyroscope's raw measurements matrix (w);
- Apply the Least Squares method to determine the 12 calibration parameters (X);

$$\mathbf{Y = w \cdot X} \quad \rightarrow \quad \mathbf{X = [w^T w]^{-1} w^T Y} \qquad (24)$$

3.3 Accelerometers

An accelerometer is a device that measures linear accelerations.

Conceptually, an accelerometer behaves as a damped mass on a spring; it is composed of a mass m constrained to two guides that can move only along a specific direction. The mass is also connected to the frame by a spring (with stiffness k) that acts parallel to the guides.

By measuring the displacement of the mass it is possible to get the acceleration to which the mass is subjected.

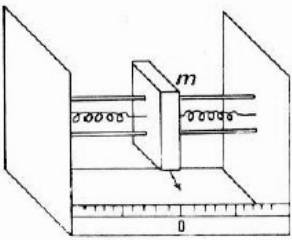

Figure 13: scheme of an accelerometer

Similarly to the case of the gyroscopes, the acceleration a_m [m/s^2] measured by an accelerometer differs from the expected one, due to the not unitary scale factor, the constant component of the bias b and due to a random process $n(t)$ that models the noise and the variable component of the bias.

$$\boldsymbol{a_m = sa + b + n(t)} \qquad \textbf{(25)}$$

The various types of accelerometers can differ for the materials used, manufacturing technology, principle of operation (pendulum, quartz resonant, floated mechanical), the operating range, bandwidth, etc.

Therefore important performance parameters are the stability of the bias and the stability of the scale factor.

Accelerometers can be heavily miniaturized and represent the reference product for the MEMS industries.

3.3.1 STATIC CONDITIONS

The accelerometers, in static conditions, are sensitive only to the acceleration of gravity g, and then can be used to partially reconstruct the orientation of a body [6].

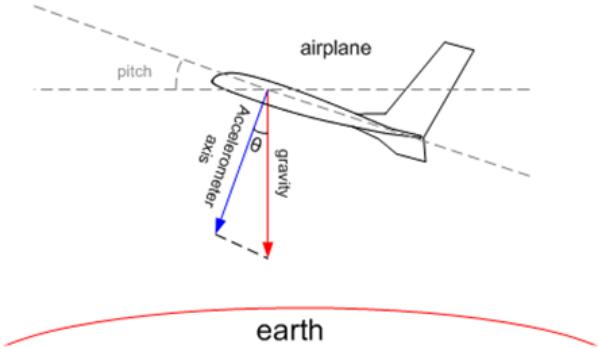

Figure 14: pitch angle

$$\textbf{ROLL ANGLE} \quad \boldsymbol{\gamma = atan2(A_y, A_z)} \tag{26}$$

$$\textbf{PITCH ANGLE} \quad \boldsymbol{\theta = atan2(A_x, A_z)} \tag{27}$$

Where A_x, A_y, A_z are the components of gravity vector, that are measured in static conditions.

The reconstruction of the attitude of a body is partial because accelerometers are insensitive to rotations about an axis parallel to the vector g.

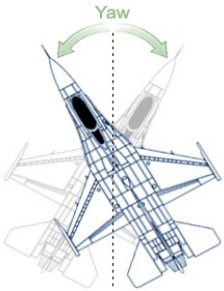

Figure 15: heading angle

$$\boldsymbol{YAW\ ANGLE = atan2(A_y, A_x) = atan2(0, 0)} \tag{28}$$

For this reason, in many applications, where it is necessary reconstruct the complete orientation of a body, magnetometers are used as well.

3.3.2 DYNAMIC CONDITIONS

In dynamic conditions accelerometers sense also the external inertial accelerations to which it is subjected:

For movements on **linear paths**, accelerometers in addition to gravity detect the component of the linear acceleration too:

Figure 16: linear path

For movements on **curved paths**, accelerometers detect centripetal (a_{cp}) and tangential acceleration (a_t) in addition to gravity:

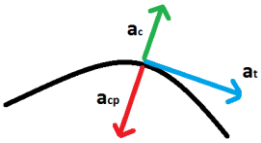

Figure 17: curved path

Nowadays, accelerometers are also used to help the GNSS (Global Navigation Satellite Systems) as the GPS to evaluate the speed or the position of an object when a satellite signal is not available, for example inside buildings or in tunnels. However, the speed provided by the accelerometer is increasingly unreliable over time due to the drift of integration and to the noise.

$$a(t) = \frac{dv(t)}{dt} = \frac{d^2p(t)}{dt^2} \tag{29}$$

$$v(t) = v_0 + \int_{t_0}^{t} a(t)dt \tag{30}$$

$$p(t) = p_0 + v_0(t - t_0) + \int \int_{t_0}^{t} a(t)dt \tag{31}$$

3.3.3 CALIBRATION

To convert the accelerometer's raw data to meaningful angular accelerations, a calibration procedure could be necessary, in particular for highly sensitive applications [6].

Considering a tri-axial accelerometer, the relationship between the raw measurements and the meaningful acceleration, can be written in the following way:

$$\begin{bmatrix} A_{x1} \\ A_{y1} \\ A_{z1} \end{bmatrix} = [A\_m]_{3x3} \begin{bmatrix} 1/A\_SC_x & 0 & 0 \\ 0 & 1/A\_SC_y & 0 \\ 0 & 0 & 1/A\_SC_z \end{bmatrix} \cdot \begin{bmatrix} A_x - A\_OS_x \\ A_y - A\_OS_y \\ A_z - A\_OS_z \end{bmatrix}$$

$$= \begin{bmatrix} ACC_{11} & ACC_{12} & ACC_{13} \\ ACC_{21} & ACC_{22} & ACC_{23} \\ ACC_{31} & ACC_{32} & ACC_{33} \end{bmatrix} \cdot \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} + \begin{bmatrix} ACC_{10} \\ ACC_{20} \\ ACC_{30} \end{bmatrix}$$

Figure 18: relationship between raw and calibrated measures

Where:

A_{x1}, A_{y1}, A_{z1} are the normalized acceleration of each axis;

A_m is the **misalignment matrix** between the accelerometer sensing axes and the device body axes;

A_SC_x, A_SC_y, A_SC_z are the **scale factors** of each axis that represent the **mismatch of the sensitivity**;

A_x, A_y, A_z are the accelerometer raw measurements of each axis;

A_OS_x, A_OS_y, A_Os_z are the **offsets** of each axis;

The previously relation can be rewritten in the following way:

$$\begin{bmatrix} A_{x1} & A_{y1} & A_{z1} \end{bmatrix} = \begin{bmatrix} A_x & A_y & A_z & 1 \end{bmatrix} \cdot \begin{bmatrix} ACC_{11} & ACC_{21} & ACC_{31} \\ ACC_{12} & ACC_{22} & ACC_{32} \\ ACC_{13} & ACC_{23} & ACC_{33} \\ ACC_{10} & ACC_{20} & ACC_{30} \end{bmatrix}$$

Figure 19: matrix form of the relationship

One possible calibration procedure consists in the calculation of 12 parameters (ACC_{11}…ACC_{30}) by means of the least square method.

In particular the steps of the procedure are:

1. Collect accelerometer raw data at 6 known stationary positions;
2. Construct the matrix of sensor raw data (w);
3. Construct the known earth gravity vector (Y);
4. Then apply the **Least Squares method** to determine the 12 calibration parameters (X);

$$\mathbf{Y = w \cdot X} \quad \rightarrow \quad \mathbf{X = [w^T w]^{-1} w^T Y} \qquad (32)$$

3.4 Magnetometers

Magnetometers are measurement instruments that measure magnetic fields such as the geomagnetic field.

Local earth magnetic field *H* has a component *Hh(Xh,Yh)* on the horizontal plane pointing to the earth's magnetic north.

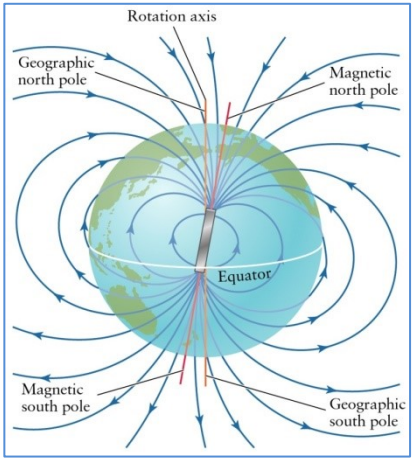

Figure 20: magnetic and geographic poles

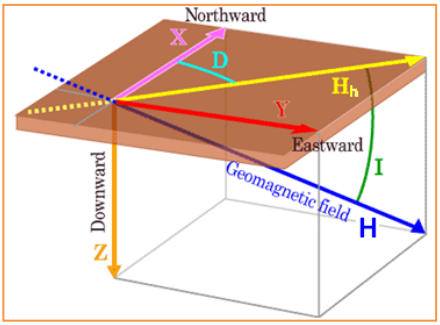

Figure 21: geomagnetic field

As it is possible to notice in the figures 19 and 20, magnetic poles are not aligned with the Geographic poles, and the angle *D* takes into account this misalignment.

The projection of the geomagnetic field on the 3 axes allows the heading angle to be computed.

Heading is defined as the angle between the X axis (northward) and *Hb(Xh,Yh)* [6].

$$\boldsymbol{YAW\ ANGLE = arctan\left(\frac{Y_h}{X_h}\right)} \qquad \textbf{(33)}$$

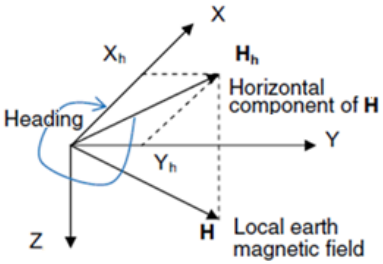

Figure 18: heading angle

In order to get correct heading information from the magnetic measurements, it is necessary to consider two different cases:

1. Leveled position:

If Roll and Pitch angles are both equal to zero, no compensation is necessary and *Xh* and *Yh* are equal to the magnetic measurements.

2. Tilted position:

If Roll and/or Pitch angles are not equal to zero, a tilt compensation is necessary.

$$X_h = X_m cos(Pitch) + Z_m sin(Pitch) \quad (34)$$

$$Y_h = X_m sin(Roll)sin(Pitch) + Y_m cos(Roll) - Z_m sin(Roll)cos(Pitch) \quad (35)$$

Where pitch and roll angle can be computed using the accelerometer measurements.

3.4.1 CALIBRATION

To convert the magnetometer's raw data to a meaningful magnetic information a calibration procedure could be necessary, in particular for highly sensitive applications [6].

Considering a tri-axial magnetometer, the relationship between the raw measurements and the meaningful acceleration can be written in the following way:

$$\begin{bmatrix} M_{x1} \\ M_{y1} \\ M_{z1} \end{bmatrix} = [M\_m]_{3x3} \begin{bmatrix} 1/M\_SC_x & 0 & 0 \\ 0 & 1/M\_SC_y & 0 \\ 0 & 0 & 1/M\_SC_z \end{bmatrix} \cdot [M\_si]_{3x3} \begin{bmatrix} M_x - M\_OS_x \\ M_y - M\_OS_y \\ M_z - M\_OS_z \end{bmatrix}$$

$$= \begin{bmatrix} MR_{11} & MR_{12} & MR_{13} \\ MR_{21} & MR_{22} & MR_{23} \\ MR_{31} & MR_{32} & MR_{33} \end{bmatrix} \cdot \begin{bmatrix} M_x - MR_{10} \\ M_y - MR_{20} \\ M_z - MR_{30} \end{bmatrix}$$

Figure 19: relationship between raw and calibrated measures

where *Mx1, My1* and *Mz1* are the normalized magnetic measurements of each axis, *M_m* is the misalignment matrix between the magnetic sensor sensing axes and the device body axes, *M_SCx, M_SCy* and *M_SCz* are the scale factors of each axis (mismatch of the sensitivity of the sensor sensing axes), *M_si* is the alignment matrix to compensate the soft-iron distortion, *Mx, My* and *Mz* are the

magnetometer raw measurements of each axis, *M_OSx, M_OSy* and *M_Osz* are the offsets of each axis caused by hard-iron distortion [6].

Hard-iron interference magnetic field is normally generated by ferromagnetic materials with permanent magnetic fields that are part of the handheld device structure. These materials could be permanent magnets or magnetized iron or steel. They are time invariant. These unwanted magnetic fields are superimposed on the output of the magnetic sensor measurements of the earth's magnetic field. The effect of this superposition is to bias the magnetic sensor outputs.

A soft-iron interference magnetic field is generated by the items inside the handheld device [6]. They could be current carrying traces on the PCB or magnetically soft materials. They generate a time varying magnetic field that is superimposed on the magnetic sensor output in response to the earth's magnetic field. The effect of the soft-iron distortion is to make a full round rotation circle become a tilted ellipse.

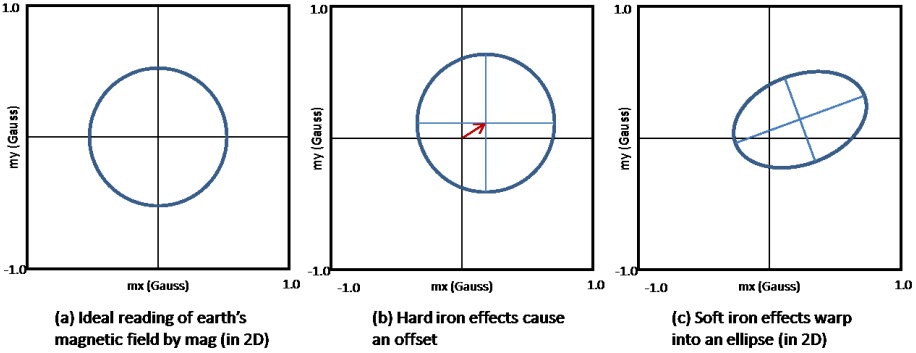

Figure 20: hard and soft iron distortions

Without hard-iron and soft-iron distortions the magnetic measures should be in a sphere of radius equal to the earth's magnetic field strength, but actually they draw a tilt ellipsoid which can be described by the following equation:

$$\frac{(x-x0)^2}{a^2}+\frac{(y-y0)^2}{b^2}+\frac{(z-z0)^2}{c^2}+\frac{(x-x0)(y-y0)}{d^2}+\frac{(x-x0)(z-z0)}{e^2}+\frac{(y-y0)(z-z0)}{f^2}=R^2$$

Figure 21: magnetic measures ellipsoid equation

where x0, y0 and z0 are the offsets *M_OSi* (*i* = *x, y, z*) caused by hard-iron distortion, x, y and z are the magnetic sensor raw data *Mx, My* and *Mz, a, b* and *c* are the semi-axes lengths, *d*, *e* and *f* are the cross axis effect to make the ellipsoid tilted, *R* is the earth's magnetic field strength.

If there is no soft-iron distortion inside the device, or the soft-iron effect is very small and can be ignored:

$$\frac{(x-x0)^2}{a^2}+\frac{(y-y0)^2}{b^2}+\frac{(z-z0)^2}{c^2}=R^2$$

Figure 22: simplified ellipsoid equation

Therefore, the least square fitting ellipsoid method can be used to discover the parameters of *M_SCi, M_OSi (i = x, y, z)* and [*M_si*].

Let's assume there is no soft-iron distortion. The soft-iron matrix [M_si] is a 3x3 identity matrix. Then above equation can be rewritten as:

$$x^2=\begin{bmatrix}x & y & z & -y^2 & -z^2 & 1\end{bmatrix}\cdot\begin{bmatrix}2x_0\\ \frac{a^2}{b^2}2y_0\\ \frac{a^2}{c^2}2z_0\\ \frac{a^2}{b^2}\\ \frac{a^2}{c^2}\\ a^2R^2-x_0^2-\frac{a^2}{b^2}y_0^2-\frac{a^2}{c^2}z_0^2\end{bmatrix}$$

Figure 237:calibration equation

After three full round rotations magnetic sensor raw data have been collected, it is possible to combine *Mx, My*, and *Mz* as column vector and row vector. Then the above equation becomes:

$$w_{nx1} = [H]_{nx6} \cdot X_{6x1}$$

The least square method can be applied to determine the parameters *X* vector as:

$$X = \left[H^T H\right]^{-1} H^T \cdot w$$

Then,

$$M\_OS_x = x_0 = X(1)/2$$
$$M\_OS_y = y_0 = X(2)/(2 \cdot X(4))$$
$$M\_OS_z = z_0 = X(3)/(2 \cdot X(5))$$
$$A = a^2R^2 = X(6) + x_0^2 + X(4) \cdot y_0^2 + X(5) \cdot z_0^2$$
$$B = A/X(4)$$
$$C = A/X(5)$$

Let,

$$xx = M_x - M\_OS_x$$
$$yy = M_y - M\_OS_y$$
$$zz = M_z - M\_OS_z$$

Then obtain:

$$\frac{xx^2}{A} + \frac{yy^2}{B} + \frac{zz^2}{C} = 1$$

Therefore,

$$M\_SC_x = \sqrt{A}$$
$$M\_SC_y = \sqrt{B}$$
$$M\_SC_z = \sqrt{C}$$

Let,

$$xxx = xx / M\_SC_x$$
$$yyy = yy / M\_SC_y$$
$$zzz = zz / M\_SC_z$$

Then,

$$xxx^2 + yyy^2 + zzz^2 = 1$$

The scale factor *M_SCi (i = x, y, z)*, the offset caused by hard-iron distortion *M_OSi*, and the 3x3 matrix [*M_si*] caused by soft-iron distortion have been determined. Applying these parameters to the measurements of the rotations, we obtain a centered unit sphere.

Similarly, the least square method can be used to determine the [*M_si*] 3x3 matrix when there is soft-iron distortion.

4 Industrial Robotics

4.1 Introduction

Current efforts of research activity in Industrial Robotics are mainly focused on improving the performance of the robot, reducing the cost, improving the safety, introducing new features (plug and play , modularity, redundancy, human robot interaction) [7][8].

Nowadays the most common sensors to measure the joints angles of an industrial manipulator are the encoders. Through the joints angles by using kinematic equations you can get the position and the orientation of every link of the manipulator.

High precision and accuracy are necessary in many robotics applications such as manufacturing [8][9], welding [9], and increasingly medical applications [10].

The encoders, in particular the high resolution ones, have the advantage of being very accurate, but unfortunately they suffer of some weaknesses such as the high cost and the need of a tight mechanical coupling [11] with the motor shaft.

Therefore, measurements from the encoders of the robot joints do not always suffice to provide the accurate position of the joints, since the motor angle does not exactly coincide with the joint angle, due to gear backlash and joint flexibility.

MEMS Inertial sensors overcome these problems since they are low cost and do not need mechanical coupling; thus, they can represent another way to measure the joints angles of an industrial manipulator.

The aim of this work is to estimate the joints angles of a 6-DOF manipulator by integrating inertial sensors measurements by means of Recursive Bayesian Filtering techniques such as the Kalman Filter (KF) and the Particle Filter (PF).

A brief state of the art of the use of inertial sensors in the estimation of joint angle is now presented.

Regarding to the use of the MEMS inertial sensors to measure the joints angles of a robotic arm, some methods with different configuration of sensors based on the rigid-body kinematics are analyzed in [12]. Experimental results show that after calibration, it is possible to get good estimation of joints angles and angular rates.

MEMS inertial sensors need a calibration procedure to ensure meaningful measurements. Two calibration procedures are described in [11]. The proposed techniques allow detecting the joint angles of a mini-excavator that works in harsh working environment, by using accelerometers.

Low cost MEMS inertial sensors can be even used in combination with low resolution encoders in order to exploit the advantages and to compensate the disadvantages of both and eventually, obtain a sensor fusion approach as proposed in [13], to estimate angle and angular rate.

In order to get good and reliable results in the estimation of the joints angles of a manipulator, it is essential to know both the mounting position and the orientation of the sensors; a method to estimate the orientation and the mounting position of the accelerometer on the robot is proposed in [14].

In [15] measurements from an optical sensor and inertial sensors are fused together by means of a KF to estimate the end effector position.

In [16] there are some experiments of applying of the EKF to estimate the joints angles of a robotic manipulator by using low-cost MEMS accelerometer in order to control the manipulator. Two calibration accelerometer approaches are illustrated.

In [17] another useful application of the PFs which allow to estimate position and orientation of an object using a position sensor and an inertial measurement unit is shown. In this paper, a PF is used to estimate the orientation and a KF is used to estimate the position and the velocity. The presented experimental results show that this hybrid method is able to find a good estimation of the orientation

even when the initial orientation is completely unknown. Moreover, factors like number of particles and position sensor noise are demonstrated to affect the orientation error.

Recursive Bayesian Filtering based on inertial sensors have been used to track the movements of the human hand and control a robot manipulator in [18].

There are not many publications regarding the application of the PF to industrial robotics. Nevertheless it is very interesting to investigate on this topic, because PFs can offer many advantages compared to the EKF.

In [19] a PF is used to estimate the joints positions and the joints angular velocities of three DOF industrial robot, using measurements that come from an accelerometer, mounted on the end effector of the robotic manipulator, and from the encoders of the joints motors. In addition, the dynamic model of the robot is used and no process noise is considered, moreover a comparison with the EKF is performed.

An efficient approach that incorporates the Kalman filter (KF) and particle filter (PF) to estimate the position and orientation of the manipulator is proposed in [20].

The angle, the angular velocity and the angular acceleration of the first three joints of a 6 DOF robot are instead estimated in the works [21] and [22] using both a PF and an EKF, still through information from an accelerometer on the tool and motor angle measurements. In this case, a laser positioning system is also used to evaluate the performance. In the case of the EKF a Gaussian distribution of the noise is assumed, and a linearization only in the measurement relation is performed.

In order to obtain a high accuracy for positioning and orientation of the end-effector of industrial robots, in [23] the PF and the EKF are compared with the results indirectly obtained by a PD controller. Starting from the differential equation of a dynamic model of a two-link flexible manipulator and using Gamma noise (non-Gaussian) as both system noise and observation noise, the results show that PF and EKF have better performance (lower error variances) than PD controllers.

Similarly, in [24] the PF and the EKF are applied to a standard industrial manipulator as sensor fusion techniques and, moreover, the Cramer Rao Lower Bound is used to evaluate the performance of the angular position estimation.

Also in this case motor angle measurements and accelerometer measurements are used in the filters and the result is evaluated with respect to the tracking performance and in terms of position accuracy of the tool.

Measurements from a 3-axis gyroscope mounted on the end-effector is also used in [25] for estimating the end-effector position. A double weighting system is used to weigh the fusion of the sensors measurements and then to evaluate the particle weight for each sensor. Experiments are performed on a 3-DOF robotic manipulator and the results show that the approach improves the position estimate accuracy.

A method to estimate joints angles of a manipulator arm using one MEMS accelerometer and gyroscope pair instead of one optical encoder for each link is presented in [26].

In this work the kinematic model of the manipulator has been used and, unlike most of the cited works, in the methods proposed in this work, the encoders have not been used.

4.2 Model of the Manipulator

The aim of this work is to estimate the joints angles of an anthropomorphic manipulator using inertial sensors measurements combined together by means of a new Recursive Bayesian Filtering based multi-sensor fusion method.

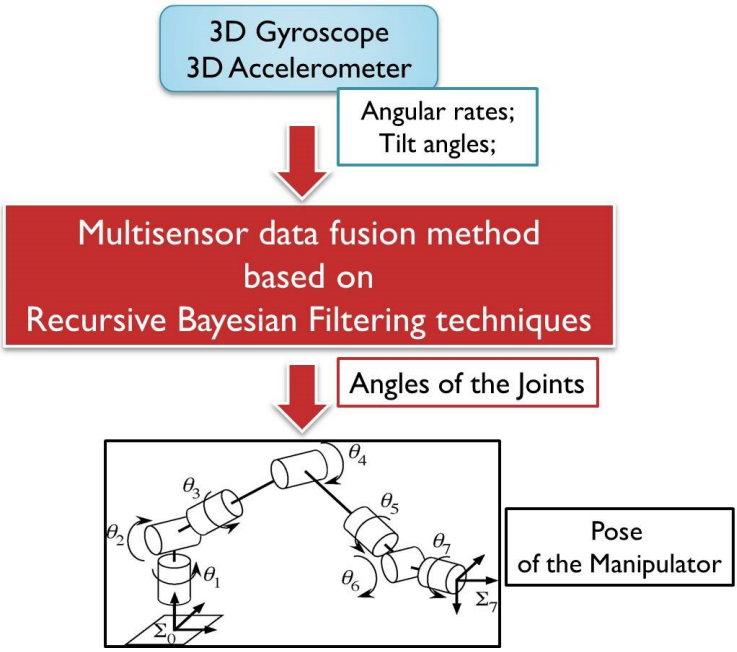

Figure 28: pose estimation scheme

In particular both KF and PF have been used. In order to use both KF and PF, the knowledge of the model of the anthropomorphic manipulator is needed.

The robot can be considered as a multi-body system, composed by 6 joints and the angle of each single joint is considered as state variable.

Regarding to the use of the sensors measurements into the model equations, for each joint, the gyroscope measurements have been used in the transition equation to obtain the derivative of the joint angles, while the accelerometer measurements have been used in the measurements equation.

The equations of the dynamic state-space model that characterize the manipulator are the following:

$$\begin{cases} \boldsymbol{q_j(k+1) = q_j(k) + \Delta t * \dot{q}_j(k) + r_{k-1}} & \textbf{(36)} \\ \boldsymbol{g^j = R_0^j * g^0 + v_k} & \textbf{(37)} \end{cases}$$

Where g^0, g^j are the accelerations of gravity expressed in the base reference system of the manipulator and in the reference system of the *j-th* joint respectively, while R_0^j is the rotation matrix between the base reference system and the reference system of the *j-th* joint.

All the reference systems have been chosen according to the Denavit-Hartenberg convention (Fig. 31) that is the standard approach [27].

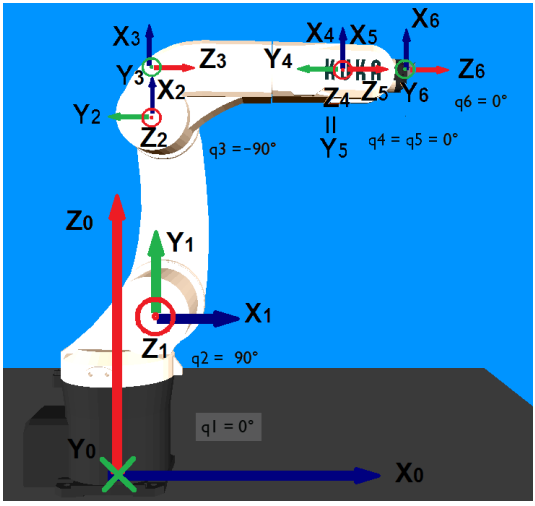

Figure 29: reference systems

The proposed multi-sensor fusion method is based on Recursive Bayesian Filtering, and it is composed by two main steps, prediction and correction. In particular gyroscopes measurements are used for the prediction step in the state transition equation while the accelerometers measurements in the measurement function for the correction of the estimation.

The state transition equation is obtained by discretizing the time, with sampling step Δt, and approximating the derivative of the joint angle with the Euler method for numerical integration:

$$\boldsymbol{q_j(k+1) = q_j(k) + \Delta t\, \dot{q}_j(k)} \tag{38}$$

The derivative of the joint angle $\dot{q}_j$ is obtained using the gyroscopes measurements ω with the following relation:

$$\boldsymbol{\{\omega_j\}^{(0)} = \{\omega_{j-1}\}^{(0)} + \dot{q}_j\{h_{j-1}\}^{(0)}} \tag{39}$$

Where $\{\omega_j\}^{(0)}$ is the angular velocity of the *j-th* arm expressed in the base reference system of the manipulator and h_{j-1} is the rotation versor of the *j-th* arm joint.

In this relation the angular velocity $\{\omega_j\}^{(0)}$ of the *j-th* arm is expressed in the base reference system of the manipulator, however we are interested in evaluating the angular velocity in the reference system of the joint, therefore the following transformation for the *j-th* joint should be applied:

$$\boldsymbol{\omega_j^{(j)} = R_0^j * \omega_j^{(0)}} \tag{40}$$

By substituting:

$$\boldsymbol{\omega_j^{(j)} = R_0^j * \{\omega_{j-1}\}^{(0)} + \dot{q}_j * R_0^j * \{h_{j-1}\}^{(0)}} \tag{41}$$

Finally, the state transition equation is:

$$\boldsymbol{\dot{q}_j = \left(R_0^j * h_{j-1}\right)^{\dagger} * (\omega_j^{(j)} - R_0^j \omega_{j-1}^{(0)})} \tag{42}$$

$$\boldsymbol{q_j(k+1) = q_j(k) + \Delta t * \left(R_0^j * h_{j-1}\right)^{\dagger} * \left(\omega_j^{(j)} - \quad R_0^j \omega_{j-1}^{(0)}\right) + r_k} \quad \textbf{(43)}$$

Where the symbol † indicate the pseudoinverse (left inverse in this case) matrix, that is the generalization of the inverse matrix for the case in which the matrix is not a square matrix .

Both R_0^j and h_{j-1} depends on the state at the previous step $q_j(k)$, hence, considering the angular velocities as input, the relation between the current state and the state at previous step is nonlinear and the process noise is additive.

The prediction of the angle is obtained by integrating the gyroscopes measurements, but it carries to the drift problem.

The drift of the estimation is because of the accumulation of the white noise of the reading in the integration.

To contrast the drift problem of the gyroscopes measurements integration, a new state variable, the bias, is introduced for each gyroscope.

The equations for the biases are slightly different for the EKF and for PF and will be presented in the next sections.

As regards the measurement function, during slow movements, the 3-axis accelerometer senses the components of the acceleration of gravity *g* expressed in the reference system of the *j-th* joint and therefore it is possible to exploit the acceleration of gravity in the base system as reference in order to obtain the joint angle:

$$\boldsymbol{g^j = R_0^j * g^0 + v_k} \quad \textbf{(44)}$$

In the measurement function, that is nonlinear, the rotation matrix R_0^j depends on the predicted state $q_j(k+1)$, obtained at the previous step with the state transition equation, and, therefore, as it is possible to see in the previous equation, also the measurement function is nonlinear.

However, in our considered manipulator configuration, the axis of the first a joint is always parallel to the gravity acceleration vector, then in this case it is impossible to use the accelerometers measures to correct the estimation of the angle of the first joint.

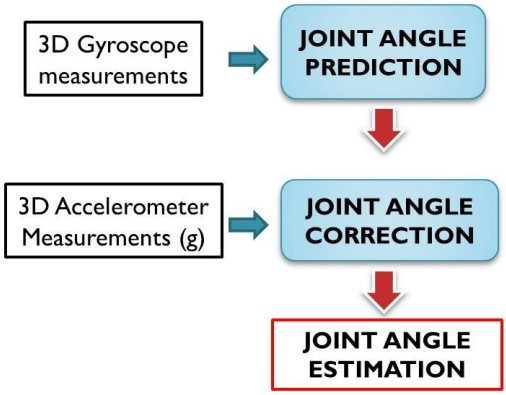

Figure 24: joint estimation scheme

4.3 Methodology

Analyzing the structure of an anthropomorphic manipulator it is possible to notice that in order to reduce costs and to avoid redundancy, the minimum number of inertial platforms needed is three.

As it is possible to see in Fig. 31, the three inertial platforms are mounted on the 2nd, on the 4th, and on the 6th joint, respectively.

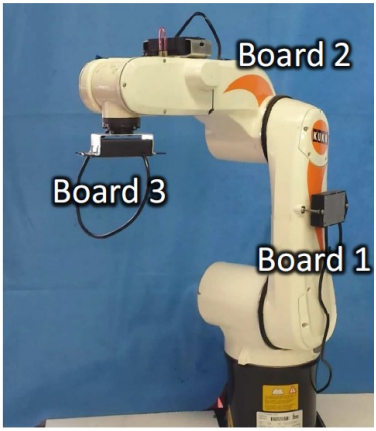

Figure 25: IMU boards

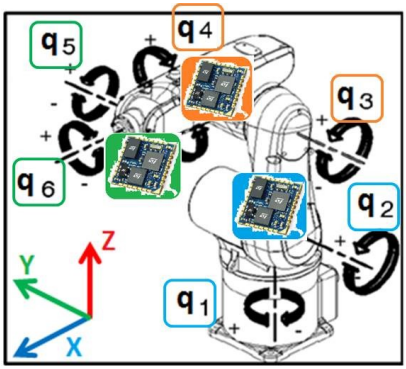

Figure 26: joint angles

With this choice it is possible to decouple the general estimation problem in three sub problems, hence three cascaded Filters may be considered (Fig. 35).

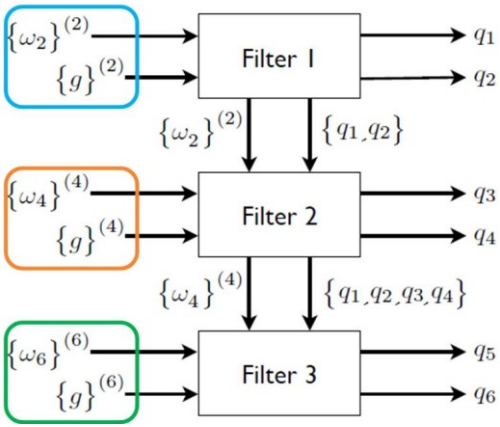

Figure 27: cascaded filters

Each Filter will estimate two joint angles, and in particular the first filter will estimate the angles of the first and the second joint, the second filter will estimate the angles of the third and the fourth joint and finally the third filter will estimate the angles of the fifth and sixth joint.

In addition, each filter will estimate the related three gyroscopes biases:

$$[\boldsymbol{x^n}] = \begin{bmatrix} \boldsymbol{q_{2n-1}} \\ \boldsymbol{q_{2n}} \\ \boldsymbol{bx\_\omega_{2n}} \\ \boldsymbol{by\_\omega_{2n}} \\ \boldsymbol{bz\_\omega_{2n}} \end{bmatrix} \tag{45}$$

Therefore the equations of the derivative of joints angles become:

$$\begin{Bmatrix} \dot{\boldsymbol{q}}_1(\boldsymbol{k}) \\ \dot{\boldsymbol{q}}_2(\boldsymbol{k}) \end{Bmatrix} = \begin{bmatrix} \sin \boldsymbol{q}_2(\boldsymbol{k}) & \boldsymbol{0} \\ \cos \boldsymbol{q}_2(\boldsymbol{k}) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{1} \end{bmatrix}^{\dagger} \{\boldsymbol{\omega_2}\}^{(2)} \tag{46}$$

$$\begin{Bmatrix}\dot{q}_3(k)\\ \dot{q}_4(k)\end{Bmatrix} = \begin{bmatrix}\sin q_4(k) & 0\\ 0 & -1\\ \cos q_4(k) & 0\end{bmatrix}^{\dagger} (\{\omega_4\}^{(4)} - R_2^4\{\omega_2\}^{(2)}) \qquad (47)$$

$$\begin{Bmatrix}\dot{q}_5(k)\\ \dot{q}_6(k)\end{Bmatrix} = \begin{bmatrix}\sin q_6(k) & 0\\ \cos q_6(k) & 0\\ 0 & 1\end{bmatrix}^{\dagger} (\{\omega_6\}^{(6)} - R_4^6\{\omega_4\}^{(4)}) \qquad (48)$$

In the equations (47) and (48) it is possible to notice that, in order to estimate the proper joint angles, for the second and the third filter it is necessary to subtract the gyroscopes measurements of the previous arms.

4.4 Kalman Filter

Since the equations of the model are nonlinear, the Extended version of the KF has been used in this work.

Prediction of the state is realized with the following equations:

$$\{x_{k+1|k}^n\} = [\Phi]\{x_k^n\} + [\Gamma]\{\dot{x}_k^n\} + r_k \qquad (49)$$

$$[P_{k+1|k}^n] = [A_{k+1|k}][P_k^n][A_{k+1|k}]^T + [W_{k+1|k}][Q][W_{k+1|k}]^T \qquad (50)$$

Where $[\Phi] = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$, $[\Gamma] = \begin{bmatrix}\Delta t & 0\\ 0 & \Delta t\end{bmatrix}$ since Δt is the sampling time of the filter and r_k is the process noise.

$P_{k+1|k}^n$ is the matrix of the error covariance, the matrices $A_{k+1|k}$ and $W_{k+1|k}$ are the Jacobian of the state equations:

$$\left[A_{k+1|k}\right] = \frac{\partial f\left(x_{k+1|k}^{n}\right)}{\partial x} \tag{51}$$

$$\left[W_{k+1|k}\right] = \frac{\partial f(x_{k+1|k}^{n})}{\partial r} \tag{52}$$

and the matrix Q is the covariance matrix of the process noise r.

In the EKF the biases are supposed constant during the prediction phase and in each step they are added to the gyroscopes measurements.

Therefore, in the prediction phase, the predicted gyroscopes biases have been added to the gyroscopes measurements, before they are integrated:

$$\left\{b\omega_j^{(j)}(k+1)\right\} = \left\{b\omega_j^{(j)}(k)\right\} \tag{53}$$

$$\left\{\omega_j^{(j)}(k+1)\right\} = \left\{\omega_j^{(j)}(k+1)\right\} + \left\{b\omega_j^{(j)}(k+1)\right\} \tag{54}$$

After the prediction step, accelerometers measurements are used to correct the state estimation; in the EKF the correction is obtained using the following equations:

$$\begin{aligned}\left[K_{k+1}^{n}\right] = {} & \left[P_{k+1|k}^{n}\right]\left[H_{k+1|k}\right]^{T}\left(\left[H_{k+1|k}\right]\left[P_{k+1|k}^{n}\right]\left[H_{k+1|k}\right]^{T}\right. \\ & + \left.\left[V_{k+1|k}\right][S]\left[V_{k+1|k}\right]^{T}\right)^{-1}\end{aligned} \tag{55}$$

$$\left\{x_{k+1}^{n}\right\} = \left\{x_{k+1|k}^{n}\right\} + \left[K_{k+1}^{n}\right]\left(\left\{z_{k+1}^{n}\right\} - \left[R\left(\left\{x_{k+1|k}^{n}\right\}\right)\right]_{2n-2}^{2n}\left\{g_{k+1}\right\}^{2n-2}\right) \tag{56}$$

$$\left[P_{k+1}^{n}\right] = \left(I - \left[K_{k+1}^{n}\right]\left[H_{k+1|k}\right]\right)\left[P_{k+1|k}^{n}\right] \tag{57}$$

Where K_{k+1}^{n} is the Kalman gain, $H_{k+1|k}$ and $V_{k+1|k}$ are the Jacobian of the measurement equation and S is the covariance matrix of the accelerometers noise v.

$$\left[H_{k+1|k}\right] = \frac{\partial h(x_{k+1|k}^{n})}{\partial x} \tag{58}$$

$$\left[V_{k+1|k}\right] = \frac{\partial h(x_{k+1|k}^{n})}{\partial v} \tag{59}$$

Both the Jacobian, related to the process noise and accelerometers noise respectively, are identity matrices, because the noises are additive.

For the second and the third filter, to be able to calculate the Jacobian (58) it is necessary to use the joint angles estimated by the previous filters.

Therefore, the second filter will have as inputs the gyroscopes measurements of the first board and the estimation of the first and the second joint angle; finally, the third filter will have as input the gyroscopes measurements of the second board and the estimation of the first, the second, the third and the fourth joint angles.

4.5 Particle Filter

In the PF, for each particle i, the prediction step is realized with the prior pdf:

$$x_k^i \sim p\left(x_k \middle| x_{k-1}^i\right) = f(x_{k-1}, u_k) \tag{60}$$

It is possible to notice that it consists of evaluating the state transition equation of the model of the manipulator, like for the EKF, many times as the particle number.

As regards the gyroscopes biases, they have been modelled through a Gaussian distribution, with the same standard deviation of the gyroscope

measurements and have been accumulated step by step; two non-linear weighting factors, the sine and cosine functions, have been used for the x and y components, to weigh more the bias of the axis involved in the rotation.

$$\left\{noise\_\omega_j^j(k)\right\} = \ normrnd\left(0, std\left\{\omega_j^j\right\}\right) \tag{61}$$

$$b\omega_{jx}^{(j)}(k+1) = b\omega_{jx}^{(j)}(k) + \ sin\left(q_k^i\right) * noise\_\omega_{jx}^j(k) \tag{62}$$

$$b\omega_{jy}^{(j)}(k+1) = b\omega_{jy}^{(j)}(k) + cos\left(q_k^i\right) * noise\_\omega_{jy}^j(k) \tag{63}$$

$$b\omega_{jz}^{(j)}(k+1) = b\omega_{jz}^{(j)}(k) + noise\_\omega_{jz}^j(k) \tag{64}$$

$$\left\{\omega_j^{(j)}(k)\right\} = \left\{\omega_j^{(j)}(k+1)\right\} + \left\{b\omega_j^{(j)}(k+1)\right\} \tag{65}$$

Where $\left\{noise\_\omega_j^j(k)\right\}$ is the Gaussian noise accumulated step by step, $b\omega_j^{(j)}$ is the gyroscopes bias and *normrnd* is the MATLAB function for Gaussian distribution.

In the PF, a weight is associated to each particle to give more importance to the best particles by means of the accelerometers measurements:

$$w_k^i \ = w_{k-1}^i \, p\left(z_k \middle| x_k^i\right) \tag{66}$$

$$w_k^i \ = w_{k-1}^i \, p_v\left(z_k - h(\, x_k)\right) \tag{67}$$

Where w_k^i is the particle weight index and p_v is the pdf of the accelerometers noise that in this case is Gaussian.

Even for the PFs, like the EKFs, the second filter needs the estimation of the first and the second joint angles, while, the third filter needs the estimation of the first, the second, the third and the fourth joint angles.

In order to avoid the divergence of the estimation, a resampling step is executed by replicating the samples with high importance weights and removing samples with low weights.

Finally the state estimation is:

$$\hat{x}_k = \hat{x}_{k-1} + \sum_{i=1}^{Particles} w_k^i * x_k^i \tag{68}$$

In both techniques the initialization of the state is performed by using the accelerometers measurements in static conditions, but in the EKF the initial state corresponds to the angle obtained from the accelerometers measurements while in the PF the particles are initialized with a Gaussian distribution centered in the angle obtained from the accelerometers measurements.

4.6 Experimental Setup

The robot used for the experiments is a KUKA KR 5 sixx R850 that is a six degrees-of-freedom anthropomorphic robot.

Figure 28: Kuka KR 5 sixx R850

The inertial platform used in this work is the MTi by Xsens that includes a triaxial accelerometer and a triaxial gyroscope.

In the experiments that have been conducted, the three IMUs mounted on the arms send the raw data to a PC for offline processing to get the estimation of the angles of the joints. The estimation of the state can be compared with the joint angles measured by the high-resolution encoders of the robot. In the first part of the experiments the robot is maintained in static conditions in order to initialize the state of the system and to compensate the offset of the gyroscopes.

4.7 Experimental Results

About fifty experimental tests have been performed to assess the real performance and limitations of the method.

The results of two tests are presented.

The sample time Δt used for both filtering techniques is 0.012 s and the number of particles is 2000. The estimation of the joints angles obtained with the simple gyroscopes integration is also presented.

4.7.1 First Experiment

The first test is composed by single joint movements.

In order to evaluate the performance of the proposed methods, the encoders have been used as reference precision system.

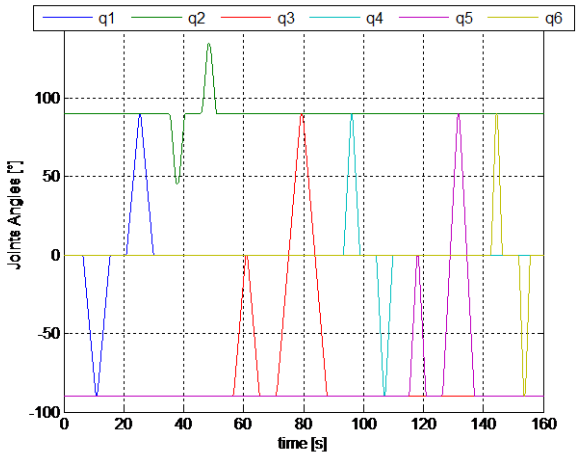

Figure 29: joint angles

As it is possible to notice from the images below, in the first experiment, the first board has the best performance while the last one has the worst performance for both methods.

The error on the estimation of the angles of the two first joints is very low and is, in both methods, between +1° and -1°.

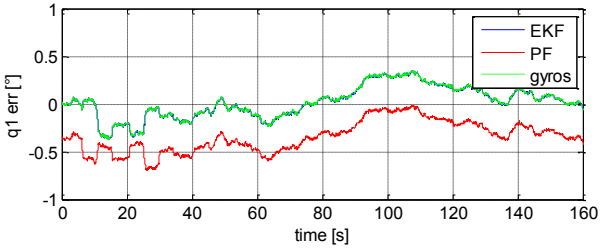

Figure 30: angle error on the first joint

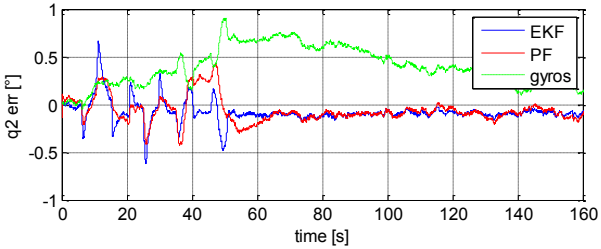

Figure 31: : angle error on the second joint

As regard the estimation of the angles of the third and the fourth joint, the two methods have almost the same performances and the maximum error is for both almost 1.5°.

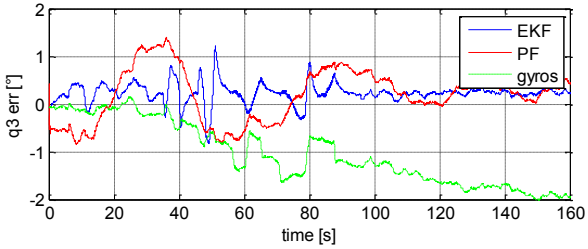

Figure 38: : angle error on the third joint

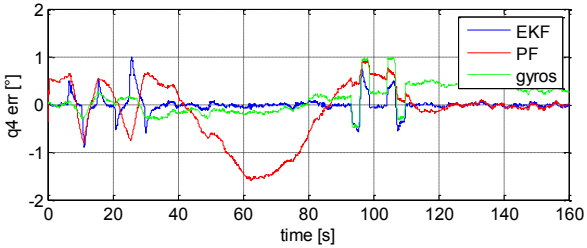

Figure 39: : angle error on the fourth joint

As it is easy to see in the tables 1,2 and 3 , the error in the estimation of the angles of the fifth and sixth joints is higher than that of the other joints, because the last board feels the inaccuracy of the previous estimations and because in the last joint there are more vibrations.

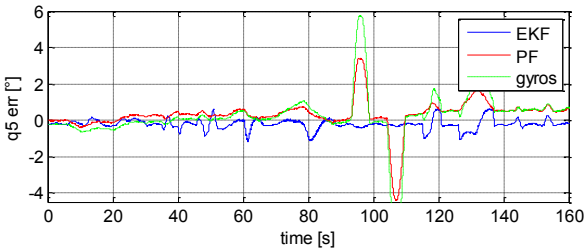

Figure 32: : angle error on the fifth joint

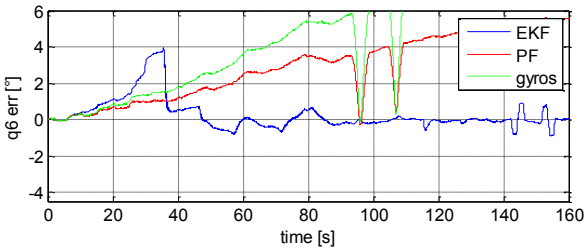

Figure 33: : angle error on the sixth joint

The maximum error in the fifth joint is about 4.5° with PF and about 1° with EKF, while in the sixth joint is almost 4° with EKF and almost 6° with PF.

In the estimation of the angles of the last two joints, the two methods have different performance and in particular the EKF has better performance than the PF because the EKF is able to compensate the noise caused by the vibrations.

Table 4: joints estimation error (EKF) [°]

joint	mean	standard deviation	root mean square	max
1	0.01	0.16	0.16	0.37
2	-0.08	0.11	0.13	0.66
3	0.22	0.20	0.30	1.22
4	-0.01	0.17	0.17	0.99
5	-0.21	0.27	0.35	1.16
6	0.25	0.85	0.89	3.91

Table 5: joints estimation error (PF) [°]

joint	mean	standard deviation	root mean square	max
1	-0.34	0.16	0.37	0.69
2	-0.06	0.13	0.14	0.59
3	0.18	0.54	0.57	1.39
4	-0.14	0.60	0.62	1.59
5	0.31	0.83	0.88	4.41
6	2.85	1.78	3.36	5.59

Table 6: joints estimation error (Gyros integration) [°]

joint	mean	standard deviation	root mean square	max
1	0.02	0.16	0.16	0.37
2	0.39	0.20	0.44	0.90
3	-1.04	0.65	1.23	2.02
4	0.11	0.28	0.30	0.99
5	0.23	1.38	1.40	7.11
6	4.72	3.20	5.71	10.58

In order to understand if the proposed methods are valid alternatives to the classic encoder, is important to analyze the performance in positioning of the end effector.

The position of the end-effector of the manipulator is shown in the following figure:

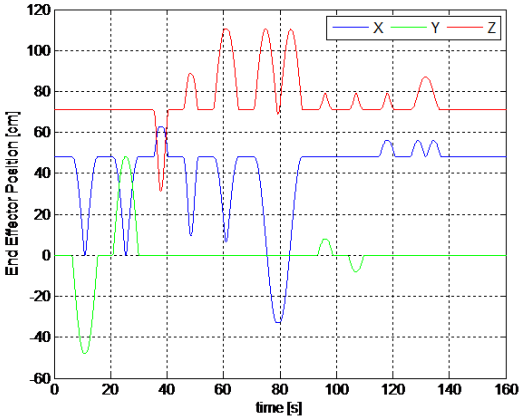

Figure 34: end effector position

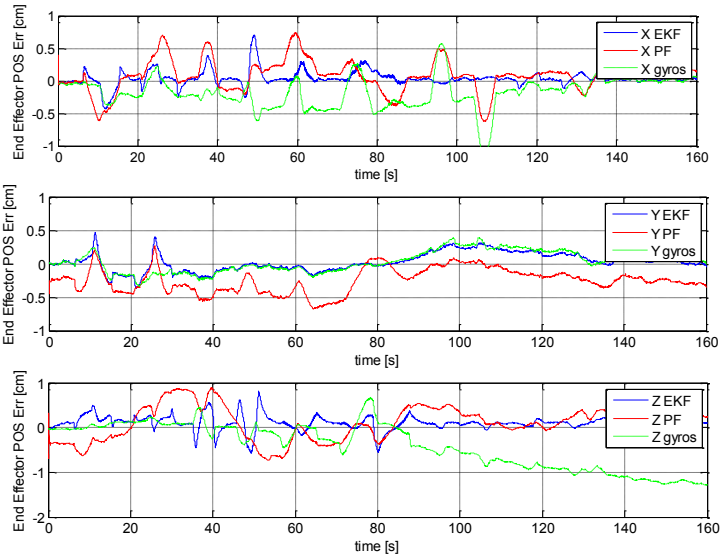

Figure 35: end effector position error

Although each joint angle estimation has a small error, these errors lead to an error in the position of the end effector that is less than 1 cm in both methods. As it is possible to notice from the tables 4, 5 and 6, the simple integration of the gyroscopes has worse performance than the two proposed methods.

Table 7: end-effector position error (EKF) [cm]

Axis	Mean	standard deviation	root mean square	max
X	0.03	0.11	0.12	0.70
Y	0.02	0.13	0.14	0.46
Z	0.10	0.15	0.18	0.81

Table 8: end-effector position error (PF) [cm]

Axis	Mean	standard deviation	root mean square	max
X	0.08	0.17	0.23	0.76

Y	-0.21	0.18	0.19	0.65
Z	0.23	0.25	0.32	0.88

Table 9: end-effector position error (Gyros integration) [cm]

Axis	Mean	standard deviation	root mean square	max
X	-0.17	0.23	0.29	1.07
Y	0.03	0.15	0.16	0.39
Z	-0.40	0.50	0.64	1.29

4.7.2 Second Experiment

In the second experiment reported in this paper, the robot performs movements that simulate a possible realistic industrial application like grasping an object that lies in a given position , in this case at the lower right of the robot, and then it moves in another position that in this case is in the upper left. These movements involve all the joints in combination (Fig.45).

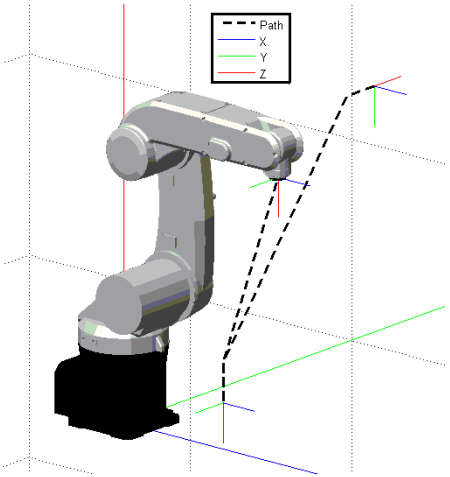

Figure 36: second experiment path

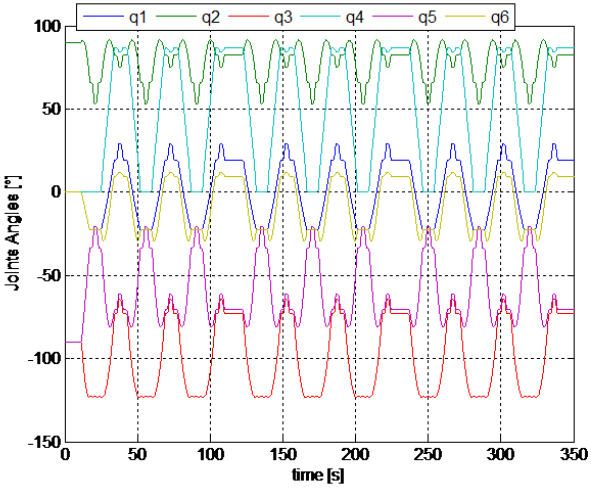

Figure 37: joint angles

As shown in the figures below, the joints that have the rotation axis parallel to the gravity suffer of a bigger error with respect to the other joints, because it is not possible to correct their estimation by means of the accelerometers measurements.

The estimation of the angle of the first joint , that as previously said is always parallel to the gravity, has a maximum error of about 2° in both EKF and PF cases.

From the results of the estimation of the angle of the first joint it is possible to see that there is a small offset between the two methods because, for its own intrinsic characteristics, the PF suffers the unfeasibility to be able to correct the estimation with the accelerometers measurements, more than the EKF.

As it is possible to see in the tables 7, 8 and 9, the estimation of the angle of the second joint has always an error approximately equal to zero with both methods.

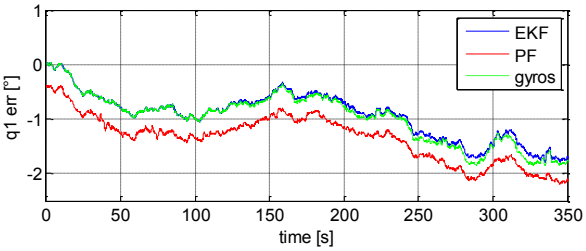

Figure 38: angle error on the first joint

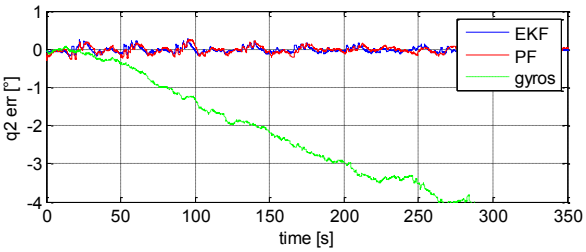

Figure 39: angle error on the second joint

Even for the angles of the third and fourth joints, that are never parallel to the gravity, the error of the estimation is very low with both methods.

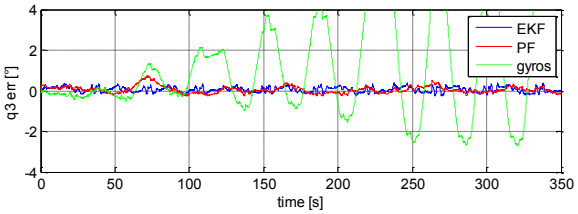

Figure 48: angle error on the third joint

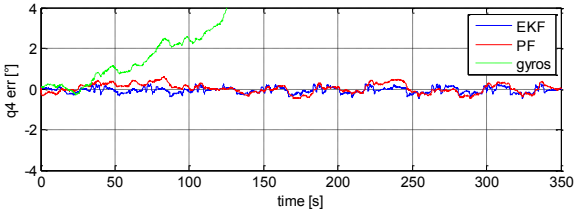

Figure 40: angle error on the fourth joint

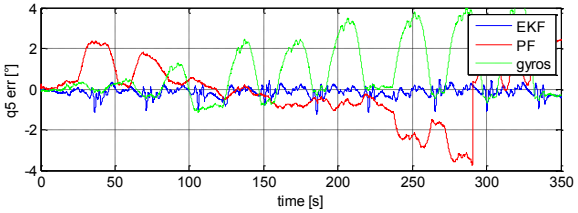

Figure 41: angle error on the fifth joint

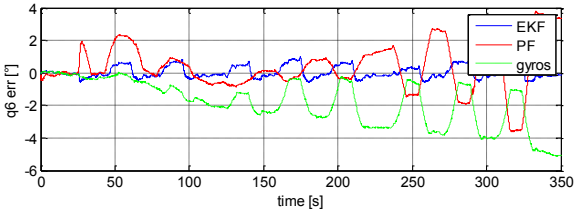

Figure 42: angle error on the sixth joint

The results of the estimation of the angles of the last two joints are the most interesting because, like for the second joint, it is possible to notice that the PF

suffer the presence of more vibrations and the parallelism with the gravity more than the EKF.

By using the PF the estimation error of the angle of the fifth and the sixth joints has a maximum value of about 4°.

As it was said before, the EKF has better performance than the PF, and in fact the estimation error of the angle of the last two joints is always below 1°.

Table 10: joints estimation error (EKF)

joint	mean	standard deviation	root mean square	max
1	0.93	0.43	1.02	1.79
2	-0.01	0.07	0.07	0.26
3	0.05	0.12	0.14	0.40
4	-0.06	0.14	0.15	0.47
5	-0.15	0.24	0.28	1.22
6	0.04	0.36	0.36	0.98

Table 11: joints estimation error (PF)

joint	mean	standard deviation	root mean square	max
1	-1.34	0.42	1.41	2.19
2	-0.01	0.08	0.08	0.29
3	0.05	0.17	0.18	0.81
4	0.02	0.23	0.23	1.03
5	-0.04	1.46	1.46	3.75
6	0.43	1.46	1.52	3.78

Table 12: joints estimation error (Gyros integration)

joint	mean	standard deviation	root mean square	a

1	-0.97	0.46	1.16	1.85
2	-2.46	1.50	8.34	4.81
3	1.23	2.42	7.43	7.68
4	5.54	3.60	4.71	11.29
5	-0.88	1.42	2.82	4.37
6	1.83	1.42	5.39	5.11

The trend of the position error of the end effector of the two methods is very similar, but even in this case it is possible to see that EKF has a better performance than the PF (see tables 10,11 and 12).

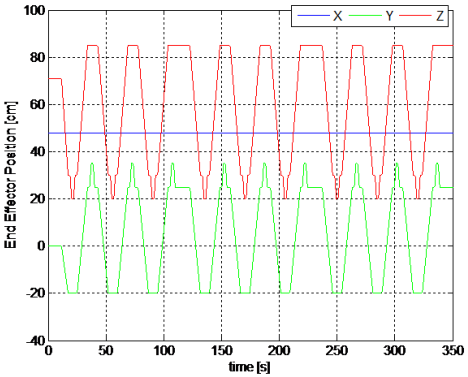

Figure 43: end effector position

For the *X* axis, EKF has a maximum position error markedly below 1 cm while the PF over 1.5 cm. Even for the *Y* axis EKF has better results and the error is below 1.5 cm while for PF, it is higher than 1.5 cm. Finally, with both methods, the error in the *Z* axis is very low and always smaller than 0.3 cm. Even in the second experiment, as it is possible to notice from the figure 15, the simple integration of the gyroscopes has worse performance than the two proposed methods.

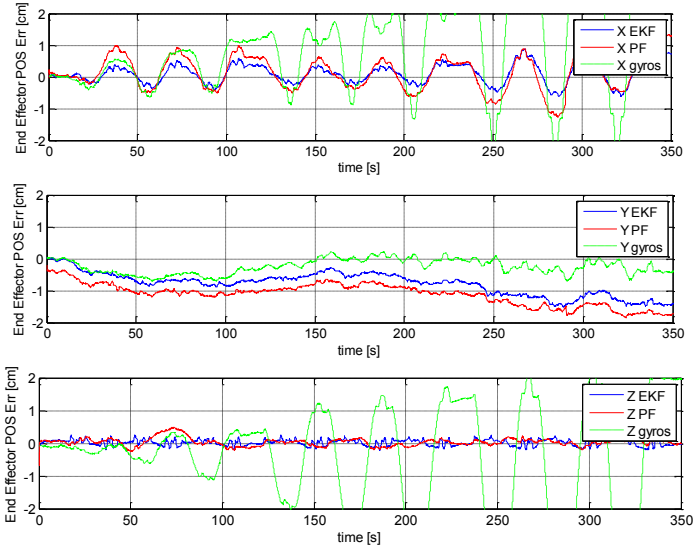

Figure 44: end effector position error

Table 13: end-effector position error (EKF)

Axis	Mean	standard deviation	root mean square	max
X	0.12	0.34	0.36	1.01
Y	-0.78	0.36	0.86	1.51
Z	0.02	0.09	0.09	0.24

Table 14: end-effector position error (PF)

Axis	Mean	standard deviation	root mean square	max
X	0.17	0.59	0.62	1.66
Y	-1.11	0.34	1.16	1.88
Z	0.03	0.12	0.12	0.66

Table 15: end-effector position error (Gyros integration)

Axis	Mean	standard deviation	root mean square	max
X	0.91	1.34	0.62	1.62
Y	-0.24	0.23	1.16	0.33
Z	0.50	1.80	0.12	1.86

4.8 Dynamic Acceleration

In the previous paragraphs only slow movements have been considered. Consequently, only the gravitational acceleration has been used in the correction step of the multi-sensor data fusion algorithms. The dynamic accelerations have not been taken into account because during slow movements these are negligible.

However, it is very important to study fast movements as well, because in these the dynamic accelerations are significant.

To take the dynamic acceleration into account it is possible to subtract it from the measured acceleration in the correction step of the multi-sensor data fusion algorithm.

To calculate the dynamic acceleration it is possible to use the predicted joints angles in the following way:

- Calculate the position of the *j-th* arm *(j = 2,4,6)* by means of the kinematic function

$$\overline{P}_j(t) = f(\bar{q}) \tag{69}$$

- Calculate the derivative of the position in order to obtain the velocity of the *j-th* arm *(j = 2,4,6)*

$$\overline{P_J}'(t) = \frac{df(\bar{q})}{d\bar{q}} \tag{70}$$

- Calculate the derivative of the velocity in order to obtain the dynamic acceleration of the *j-th* arm *(j = 2,4,6)*
-

$$\overline{P_J}''(t) = \frac{d^2 f(\bar{q})}{d\bar{q}^2} \tag{71}$$

Since the dynamic acceleration will be subtracted from the measurements of the accelerometers, it is necessary to calculate the dynamic acceleration of the arms considering the position of the boards.

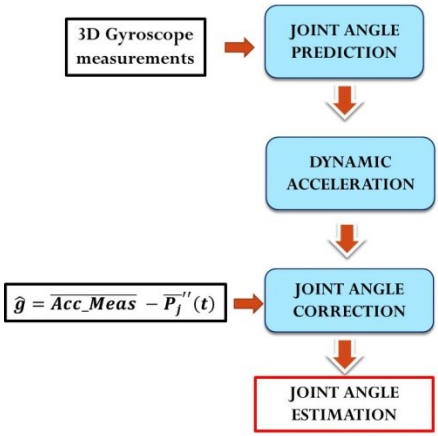

Figure 45: joint estimation scheme

In order to assess the real performance and limitations of the method, the results of an experiment is presented:

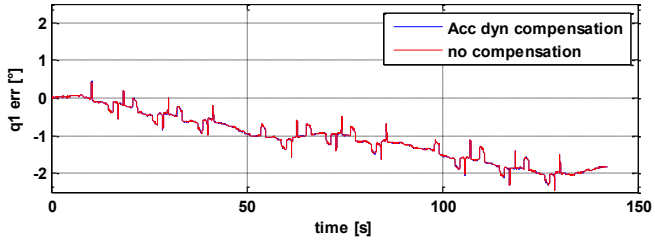

Figure 46: angle error on the first joint

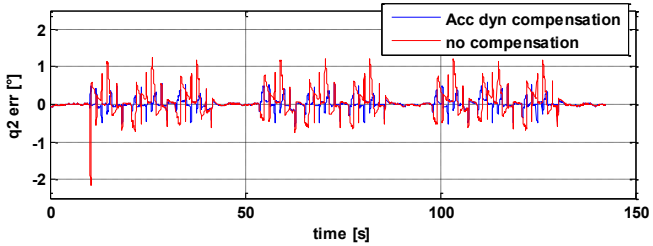

Figure 47: angle error on the second joint

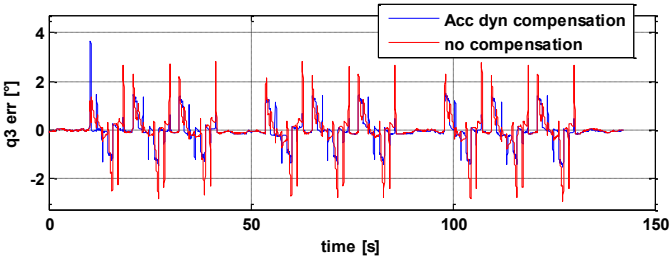

Figure 48: angle error on the third joint

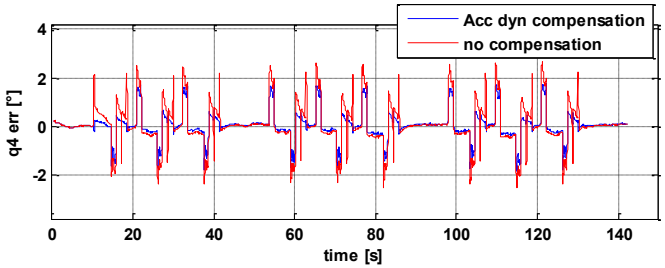

Figure 58: angle error on the fourth joint

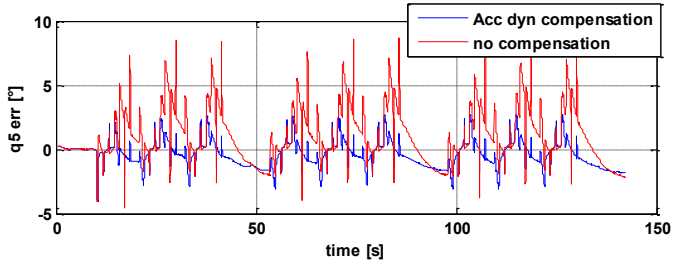

Figure 59: angle error on the fifth joint

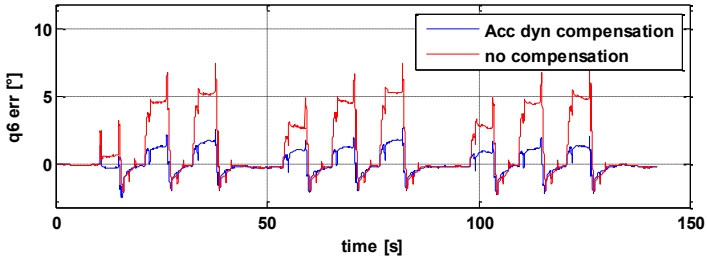

Figure 49: angle error on the sixth joint

Table 16: joints estimation error (EKF) with Acc dyn compensation

joint	mean	standard deviation	root mean square	max
1	-1.06	0.64	1.54	2.46
2	0.01	0.12	0.01	0.59
3	-0.03	0.50	0.25	3.67
4	0.04	0.54	0.29	1.8
5	-0.40	1.00	1.17	4.04
6	0.13	0.83	0.71	2.68

Table 17: joints estimation error (EKF) without Acc dyn compensation

joint	mean	standard deviation	root mean square	max
1	-1.06	0.64	1.54	2.46
2	0.05	0.30	0.09	2.14
3	-0.02	0.66	0.44	2.94
4	0.08	0.82	0.68	2.66
5	1.00	2.10	5.44	8.66
6	1.16	2.19	6.19	7.45

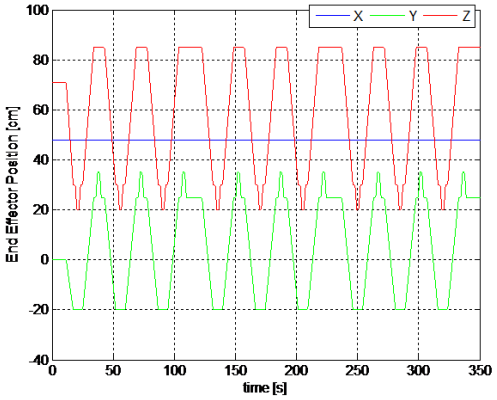

Figure 50: end effector position

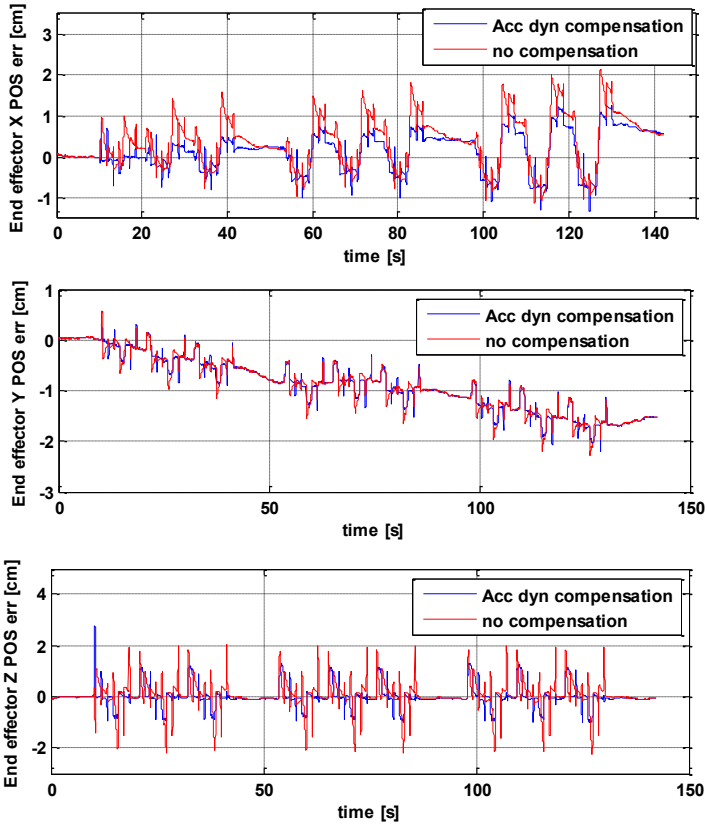

Figure 51: end effector position error

Table 18: end-effector position error (EKF) with Acc dyn compensation

Axis	Mean	standard deviation	root mean square	max
X	0.15	0.47	0.49	1.32
Y	-0.88	0.54	1.03	2.20
Z	-0.01	0.39	0.39	2.78

Table 19: end-effector position error (EKF) without Acc dyn compensation

Axis	Mean	standard deviation	root mean square	max
X	0.37	0.62	0.72	2.14
Y	-0.88	0.54	1.04	2.28
Z	0.05	0.50	0.51	2.25

5 UAV Attitude Heading Reference System

5.1 Introduction

In this chapter the development of an attitude heading reference system (AHRS) for an UAV is described.

This work is part of a bigger project for the development of an autopilot for an UAV.

Data from a inertial platform and a 3D magnetometer are fused together by an Extended Kalman Filter in order to obtain the Euler angles of the drone.

The EKF algorithm has been tested both in simulation and in the real flying robot.

5.2 CANaerospace

In order to manage and communicate with the Attitude and Heading Reference System (AHRS) of an UAV, the CANaerospace protocol has been developed for the iNEMO M1 microcontroller.

CANaerospace is a lightweight protocol developed by Stock Flight Systems in 1998 for avionic and aeronautical applications, which provides a highly reliable communication via Controller Area Network (CAN) [28].

The definition of the CANaerospace protocol is kept widely open to allow implementation of user-defined message types and protocols.

5.2.1 Main features

- **Democratic network**: CANaerospace does not require any master/slave relationships between LRUs or a "bus controller", thereby avoiding a potential single source of failure. Every node in the network has the same rights for participation in the bus traffic.
- **Self-identifying message format**: Each CANaerospace message contains information about the type of the data and the transmitting node. This allows the data to be unambiguously recognized at each receiving node.
- **Continuous Message Numbering**: Each CANaerospace message contains a continuously incremented number which allows coherent processing of messages in the receiving stations.
- **Message Status Code**: Each CANaerospace message contains information about the integrity of the data is conveying. This allows receiving stations to evaluate the quality of the received data and to react accordingly.
- **Emergency Event Signaling**: CANaerospace defines a mechanism that allows each node to transmit information about exception or error situations. This information can be used by other stations to determine the network health.
- **Node Service Interface**: As an enhancement to CAN, CANaerospace provides a means for individual stations on the network to communicate with each other using connection-oriented and connectionless services.
- **Predefined CAN Identifier Assignment**: CANaerospace offers a predefined identifier assignment list for normal operation data. In addition to the predefined list, user-defined identifier assignment lists may be used.
- **Ease of Implementation**: The amount of code to implement CANaerospace is very small by design in order to minimize the effort for testing and certification of flight safety critical systems.

- **Openness to Extensions**: All CANaerospace definitions are extendable to provide flexibility for future enhancements and to allow adaptions to the requirements of specific applications.
- **Free Availability**: No cost whatsoever applies for the use of CANaerospace. The specification can be downloaded from the Internet.

5.2.2 Frame Format

The format of the messages of the CANaerospace protocol [29] is shown in the following figure:

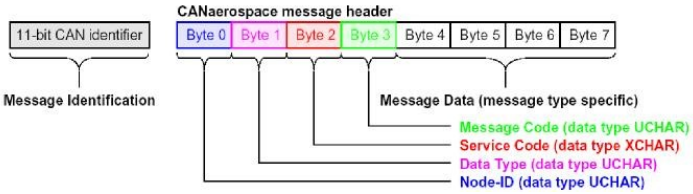

Figure 52: format of CAN frame

CAN messages have an identifier (ID) that specifies the type of the message; there are two different version of identifier: standard identifier of 11 bit or extended identifier of 32 bit. CANaerospace protocol uses the standard identifier.

In order to enable both Anyone-to-Many (ATM) and Peer-to-Peer (PTP) communication for CAN, CANaerospace defines different groups of messages:

Channel Acronym	Communication Type	Description	CAN Identifier Range	Message Priority
EED	ATM	Emergency Event Data Channel	0 - 127	Highest
NSH	PTP	High Priority Node Service Data Channel	128 - 199	
UDH	ATM/PTP	High Priority User-Defined Data Channel	200 - 299	
NOD	ATM	Normal Operation Data Channel	300 - 1799	
UDL	ATM/PTP	Low Priority User-Defined Data Channel	1800 - 1899	
DSD	ATM/PTP	Debug Service Data Channel	1900 - 1999	
NSL	PTP	Low Priority Node Service Data Channel	2000 - 2031	Lowest

Figure 53: CANaerospace identifiers

- **Emergency Event Data Channel** (EED): This communication channel is used for messages which require immediate action (i.e. system degradation or reconFiguretion) and have to be transmitted with very high priority. Emergency Event Data uses ATM communication exclusively.
- **High/Low Priority Node Service Data Channel** (NSH/NSL): These communication channels are used for client/server interactions using PTP communication. The corresponding services may be of the connection-oriented as well as the connectionless type. NSH/NSL may also be used to support test and maintenance functions.
- **Normal Operation Data Channel** (NOD): This communication channel is used for the transmission of the data which is generated during normal system operation and described in the CANaerospace identifier assignment list. These messages may be transmitted periodically or aperiodically as well as synchronously or asynchronously. All messages which cannot be assigned to other communication channels shall use this channel.
- **High/Low Priority User-Defined Data Channel** (UDH/UDL): This channel is dedicated to communication which cannot, due to their specific characteristics, be assigned other channels without violating the CANaerospace specification. As long as the defined identifier range is used, the message content and the communication type (ATM, PTP) for these channels may be specified by the system designer. To ensure interoperability it is highly recommended that the use of these channels is minimized.

- **Debug Service Data Channel** (DSD): This channel is dedicated to messages which are used temporarily for development and test purposes only and are not transmitted during normal operation. As long as the defined identifier range is used, the message content and the communication type (ATM, PTP) for these channels may be specified by the system designer.

In our application we used the channels NSH, NOD and DSD.

A CANaerospace message basically consists of an **header** of 4 byte and a **data** field up to 4 byte.

The **header** includes the following fields:

- **Node ID**: indicates the identifier of the receiver node in the case of NSH messages and the identifier of the transmitter node in the case of NOD messages.
- **Data type**: specifies the data type of the payload and can be float, long, ulong, short, ushort, char or a data type specified by the user.
- **Service Code**: it defines the type of the service in the case of NSH/NSL messages, or is reserved for particular aims of the application in the case of NOD/EED messages.
- **Message Code**: includes further specifications in the case of NSH/NSL messages, or represents a message counter for debugging purposes in the case of NOD/EED messages.

5.2.3 Frame NSH

The following services have been implemented:

IDS: Identification service

This frame is used to identify both the hardware and the software of the AHRS board.

HMI -> M1

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
NS_REQ_1_ID *dec:130 hex:0x82*	*Node ID*	*Data type*	*Service code*	*Message code*				
	5	AS_NODATA (0x0)	IDS (0x0)	0	0	0	0	0

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
NS_RSP_1_ID *dec:131 hex:0x83*	*Node ID*	*Data type*	*Service code*	*Message code*				
	5	AS_UCHAR_2 (0x13)	IDS (0x0)	*ErrorFlag*	*iNemo Hw rev*	*iNemo Sw rev*	0	0

DTS: Data Transfer Service

This frame configures the AHRS data transfer mode.

HMI -> M1

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
NS_REQ_1_ID *dec:130 hex:0x82*	*Node ID*	*Data type*	*Service code*	*Message code*			*DataSelection*	
	5	AS_UCHAR_4 (0x10)	DTS (0xD4)	0	*TransferMode*	*Datarate*	*MSB*	*LSB*

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
NS_RSP_1_ID *dec:131 hex:0x83*	*Node ID*	*Data type*	*Service code*	*Message code*			*DataSelection*	
	5	AS_UCHAR_4 (0x10)	DTS (0xD4)	*ErrorFlag*	*TransferMode*	*Datarate*	*MSB*	*LSB*

Data[4]:

TransferMode = 0 indicates a streaming mode transmission.
TransferMode = 1 indicates a remote request transmission.

Data[5]:

In the case of streaming mode transmission, Datarate indicates the frequency of the transmission in Hz.

Data[6]:

DataSelection indicates the data that the AHRS has to transmit (bit = 1 transmit, bit = 0 no transmit).

Table 20: variable selection

DataSelection	Variable
Bit 0	Roll
Bit 1	Pitch
Bit 2	Yaw
Bit 3	Quat-Q0
Bit 4	Quat-Q1
Bit 5	Quat-Q2
Bit 6	Quat-Q3
Bit 7	AccX
Bit 8	AccY
Bit 9	AccZ
Bit 10	GyrX
Bit 11	GyrY
Bit 12	GyrZ
Bit 13	MagX
Bit 14	MagY
Bit 15	MagZ

RTS: Raw Data Transfer Service

This frame configures the AHRS raw data transfer mode.

HMI -> M1

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
NS_REQ_1_ID *dec:130 hex:0x82*	*Node ID*	*Data type*	*Service code*	*Message code*			*Raw DataSelection*	
	5	AS_UCHAR_3 (0x1B)	RTS (0xD8)	0	*Raw TransferMode*	*Raw Datarate*	*MSB*	*LSB*

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
NS_RSP_1_ID *dec:131 hex:0x83*	*Node ID*	*Data type*	*Service code*	*Message code*			*Raw DataSelection*	
	5	AS_UCHAR_3 (0x1B)	RTS (0xD8)	*ErrorFlag*	*Raw TransferMode*	*Raw Datarate*	*MSB*	*LSB*

Data[4]:

RawTransferMode = 0 indicates a streaming mode transmission.
RawTransferMode = 1 indicates a remote request transmission.

Data[5]:

In the case of streaming mode transmission, RawDatarate indicates the frequency of the transmission in Hz.

Data[6]:

RawDataSelection indicates the data that the AHRS has to transmit (bit = 1 transmit, bit = 0 no transmit).

Table 21: debug frame selection

DataSelection	Variable
Bit 0	Degub Frame 1
Bit 1	Degub Frame 2
Bit 2	Degub Frame 3
Bit 3	Degub Frame 4
Bit 4	Degub Frame 5

SSS: Start and Stop Data Transfer Service

This frame indicates the start and the stop of the data transfer of the iNEMO M1. There are two different flags for the two different flows data, one for DTS command and one for RTS command.

HMI -> M1

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
NS_REQ_1_ID *dec:130 hex:0x82*	*Node ID*	*Data type*	*Service code*	*Message code*				
	5	AS_UCHAR_2 (0x13)	SSS (0xD5)	0	*0*	*0*	*StartStopData*	*StartStopRawData*

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
NS_RSP_1_ID *dec:131 hex:0x83*	*Node ID*	*Data type*	*Service code*	*Message code*				
	5	AS_UCHAR_2 (0x13)	SSS (0xD5)	*ErrorFlag*	*0*	*0*	*StartStopData*	*StartStopRawData*

Data[6]:

StartStopData = 1 starts the data transmission; in the case of remote request mode transfer it is necessary to send this command every time you want receive data.

StartStopData = 2 stops the data transmission in the case of streaming mode transfer.

StartStopData = 0 does not change the flow data.

CDS: Control Parameters Download Service

This frame is used to download the conFiguretion parameters to the iNEMO M1.

HMI -> M1

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
NS_REQ_1_ID *dec:130 hex:0x82*	*Node ID*	*Data type*	*Service code*	*Message code*	*Variable name: Param_x*			
	5	AS_FLOAT_2 (0x02)	CDS (0xA3)	*Parameter identifier*	*FloatMSB*	*Float2*	*Float3*	*FloatLSB*

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
NS_RSP_1_ID *dec:131 hex:0x83*	*Node ID*	*Data type*	*Service code*	*Message code*	*Variable name: Param_x*			
	5	AS_FLOAT_2 (0x02)	CDS (0xA3)	*Parameter identifier*	*FloatMSB*	*Float2*	*Float3*	*FloatLSB*

The configuration parameters are selected through the bits of the *parameter_identifier* variable as shown in the following table:

Table 22: parameters list

parameter_identifier	Description	Parameter Name
0	Ellipsoid X axis eccentricity	EccX
1	Ellipsoid Y axis eccentricity	EccY
2	Ellipsoid Z axis eccentricity	EccZ
3	Ellipsoid X axis radius	ErX
4	Ellipsoid Y axis radius	ErY
5	Ellipsoid Z axis radius	ErZ
6	Earth magnetic field X axis component	EmfX
7	Earth magnetic field Y axis component	EmfY
8	Earth magnetic field Z axis component	EmfZ
9	Gravity vector X axis component	GX
10	Gravity vector Y axis component	GY
11	Gravity vector Z axis component	GZ
12	Roll Offset	offset_Roll
13	Pitch Offset	offset_Pitch
14	Yaw Offset	offset_Yaw

CUS: Control Parameters Upload Service

This frame is used to upload the configuration parameters from the iNEMO M1.

HMI -> M1

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
NS_REQ_1_ID *dec:130 hex:0x82*	*Node ID*	*Data type*	*Service code*	*Message code*				
	5	AS_NODATA (0x0)	CUS (0xA5)	*Parameter identifier*	*0*	*0*	*0*	*0*

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
NS_RSP_1_ID *dec:131 hex:0x83*	*Node ID*	*Data type*	*Service code*	*Message code*	*Variable name: Param_x*			
	5	AS_FLOAT_2 (0x02)	CUS (0xA5)	*Parameter identifier*	*FloatMSB*	*Float2*	*Float3*	*FloatLSB*

PRS: Parameters Reset Service

This frame is used to set the conFiguretion parameters to a default value:

HMI -> M1

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
NS_REQ_1_ID *dec:130 hex:0x82*	*Node ID*	*Data type*	*Service code*	*Message code*				
	5	AS_NODATA (0x0)	CUS (0xA5)	*Parameter identifier*	*0*	*0*	*0*	*0*

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
NS_RSP_1_ID *dec:131 hex:0x83*	*Node ID*	*Data type*	*Service code*	*Message code*	*Variable name: Param_x*			
	5	AS_FLOAT_2 (0x02)	CUS (0xA5)	*Parameter identifier*	*FloatMSB*	*Float2*	*Float3*	*FloatLSB*

The default values of the different conFiguretion parameters are shown in the following table:

Table 23: default conFiguretion parameters

parameter_identifier	Parameter Name	Default value
0	EccX	-0.1152
1	EccY	-0.0234
2	EccZ	0.0179
3	ErX	0.3851
4	ErY	0.3636
5	ErZ	0.4150
6	EmfX	0.82
7	EmfY	0.0
8	EmfZ	0.73
9	GX	0.0
10	GY	0.0
11	GZ	-9.8
12	offset_Roll	0.0
13	offset_Pitch	0.0
14	offset_Yaw	0.0

CMU: Change Measurement Unit

This frame changes the measurement unit of the Euler angles:

HMI -> M1

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
NS_REQ_1_ID *dec:130*	*Node ID*	*Data type*	*Service code*	*Message code*				
hex:0x82	5	AS_UCHAR (0x0A)	CUM(0xD3)	*0*	*0*	*0*	*0*	*0*

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
NS_RSP_1_ID *dec:131*	*Node ID*	*Data type*	*Service code*	*Message code*				
hex:0x83	5	AS_UCHAR (0x0A)	CUM(0xD3)	*ErrorFlag*	*0*	*0*	*0*	*Unit*

Data[7]:

Unit = 0 indicates that Euler angles are in radiant.

Unit = 1 indicates that Euler angles are in degree.

ROS: Reset Orientation Service

This frame is used to reset the roll and pitch angles of the iNEMO M1.

HMI -> M1

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
NS_REQ_1_ID *dec:130* *hex:0x82*	*Node ID*	*Data type*	*Service code*	*Message code*				
	5	AS_NODATA(0x0)	ROS(0xD6)	*0*	*0*	*0*	*0*	*0*

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
NS_RSP_1_ID *dec:131* *hex:0x83*	*Node ID*	*Data type*	*Service code*	*Message code*				
	5	AS_NODATA(0x0)	ROS(0xD6)	*ErrorFlag*	*0*	*0*	*0*	*0*

RHS: Reset Heading Service

This frame is used to reset the yaw angle of the iNEMO M1.

HMI -> M1

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
NS_REQ_1_ID *dec:130* *hex:0x82*	*Node ID*	*Data type*	*Service code*	*Message code*				
	5	AS_NODATA(0x0)	RHS(0xD7)	*0*	*0*	*0*	*0*	*0*

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
NS_RSP_1_ID *dec:131* *hex:0x83*	*Node ID*	*Data type*	*Service code*	*Message code*				
	5	AS_NODATA(0x0)	RHS(0xD7)	*ErrorFlag*	*0*	*0*	*0*	*0*

RCS: Reset CPU Service

This frame is used to reset the iNEMO M1.

HMI -> M1

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
NS_REQ_1_ID *dec:130* *hex:0x82*	*Node ID*	*Data type*	*Service code*	*Message code*				
	5	AS_NODATA(0x0)	RCS(0xD2)	*0*	*0*	*0*	*0*	*0*

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
NS_RSP_1_ID *dec:131* *hex:0x83*	*Node ID*	*Data type*	*Service code*	*Message code*				
	5	AS_NODATA(0x0)	RCS(0xD2)	*ErrorFlag*	*0*	*0*	*0*	*0*

5.2.4 Frame NOD

The following Normal Operation Data frames have been implemented:

PITCH

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
BODY_PITCH_ANGLE_ID *dec:311* *hex:0x137*	*Node ID*	*Data type*	*Service code*	*Message code*	*Variable name: Pitch*			
	5	AS_FLOAT_2 (0x02)	0	*0*	*FloatMSB*	*Float2*	*Float3*	*FloatLSB*

ROLL

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
BODY_ROLL_ANGLE_ID *dec:312* *hex:0x138*	*Node ID*	*Data type*	*Service code*	*Message code*	*Variable name: Roll*			
	5	AS_FLOAT_2 (0x02)	0	*0*	*FloatMSB*	*Float2*	*Float3*	*FloatLSB*

YAW

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
HEADING_ANGLE_ID *dec:321*	*Node ID*	*Data type*	*Service code*	*Message code*	*Variable name: Heading*			

hex:0x141	5	AS_FLOAT_2 (0x02)	0	*0*	*FloatMSB*	*Float2*	*Float3*	*FloatLSB*

Longitudinal Acceleration (X axis)

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
BODY_LONG_ACC_ID *dec:300* *hex:0x12C*	*Node ID*	*Data type*	*Service code*	*Message code*	*Variable name: acc_X*			
	5	AS_FLOAT_2 (0x02)	0	*0*	*FloatMSB*	*Float2*	*Float3*	*FloatLSB*

Lateral Acceleration (Y axis)

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
BODY_LAT_ACC_ID *dec:301* *hex:0x12D*	*Node ID*	*Data type*	*Service code*	*Message code*	*Variable name: acc_Y*			
	5	AS_FLOAT_2 (0x02)	0	*0*	*FloatMSB*	*Float2*	*Float3*	*FloatLSB*

Normal Acceleration (Z axis)

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
BODY_NORM_ACC_ID *dec:302* *hex:0x12E*	*Node ID*	*Data type*	*Service code*	*Message code*	*Variable name: acc_Z*			
	5	AS_FLOAT_2 (0x02)	0	*0*	*FloatMSB*	*Float2*	*Float3*	*FloatLSB*

Unit Quaternion component Q0

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
BODY_QUAT_Q0_ID *dec:1500* *hex:0x5DC*	*Node ID*	*Data type*	*Service code*	*Message code*	*Variable name: Quat_q0*			
	5	AS_FLOAT_2 (0x02)	0	*0*	*FloatMSB*	*Float2*	*Float3*	*FloatLSB*

Unit Quaternion component Q1

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
BODY_QUAT_Q1_ID *dec:1501* *hex:0x5DD*	*Node ID*	*Data type*	*Service code*	*Message code*	*Variable name: Quat_q1*			
	5	AS_FLOAT_2 (0x02)	0	*0*	*FloatMSB*	*Float2*	*Float3*	*FloatLSB*

Unit Quaternion component Q2

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
BODY_QUAT_Q2_ID *dec:1500 hex:0x5DE*	*Node ID*	*Data type*	*Service code*	*Message code*	*Variable name: Quat_q2*			
	5	AS_FLOAT_2 (0x02)	0	*0*	*FloatMSB*	*Float2*	*Float3*	*FloatLSB*

Unit Quaternion component Q3

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
BODY_QUAT_Q3_ID *dec:1503 hex:0x5DF*	*Node ID*	*Data type*	*Service code*	*Message code*	*Variable name: Quat_q3*			
	5	AS_FLOAT_2 (0x02)	0	*0*	*FloatMSB*	*Float2*	*Float3*	*FloatLSB*

Gyroscope X

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
BODY_GYR_X_ID *dec:1504 hex:0x5E0*	*Node ID*	*Data type*	*Service code*	*Message code*	*Variable name: gyr_X*			
	5	AS_FLOAT_2 (0x02)	0	*0*	*FloatMSB*	*Float2*	*Float3*	*FloatLSB*

Gyroscope Y

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
BODY_GYR_X_ID *dec:1505 hex:0x5E1*	*Node ID*	*Data type*	*Service code*	*Message code*	*Variable name: gyr_Y*			
	5	AS_FLOAT_2 (0x02)	0	*0*	*FloatMSB*	*Float2*	*Float3*	*FloatLSB*

Gyroscope Z

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
BODY_GYR_X_ID *dec:1506 hex:0x5E2*	*Node ID*	*Data type*	*Service code*	*Message code*	*Variable name: gyr_Z*			
	5	AS_FLOAT_2 (0x02)	0	*0*	*FloatMSB*	*Float2*	*Float3*	*FloatLSB*

Magnetometer X

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
BODY_MAGN_X_ID *dec:1507* *hex:0x5E3*	*Node ID*	*Data type*	*Service code*	*Message code*	*Variable name: Magn _X*			
	5	AS_FLOAT_2 (0x02)	0	*0*	*FloatMSB*	*Float2*	*Float3*	*FloatLSB*

Magnetometer Y

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
BODY_MAGN_Y_ID *dec:1508* *hex:0x5E4*	*Node ID*	*Data type*	*Service code*	*Message code*	*Variable name: Magn _Y*			
	5	AS_FLOAT_2 (0x02)	0	*0*	*FloatMSB*	*Float2*	*Float3*	*FloatLSB*

Magnetometer Z

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
BODY_MAGN_Y_ID *dec:1509* *hex:0x5E5*	*Node ID*	*Data type*	*Service code*	*Message code*	*Variable name: Magn _Z*			
	5	AS_FLOAT_2 (0x02)	0	*0*	*FloatMSB*	*Float2*	*Float3*	*FloatLSB*

5.2.5 Frame DSD

The following Debug Service frames have been implemented:

Degub Frame 1 – accelerometer x and y raw data

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
DEBUG_FRAME_1_ID *dec:1920* *hex:0x780*	*Node ID*	*Data type*	*Service code*	*Message code*	*raw_Acc_X*		*raw_Acc_Y*	
	5	AS_USHORT_2 (0x0D)	0	*0*	*MSB*	*LSB*	*MSB*	*LSB*

Degub Frame 2 – accelerometer z and gyroscope y raw data

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
DEBUG_FRAME_2_ID *dec:1921* *hex:0x781*	*Node ID*	*Data type*	*Service code*	*Message code*	*raw_Acc_Z*		*raw_Gyr_X*	
	5	AS_USHORT_2 (0x0D)	0	*0*	*MSB*	*LSB*	*MSB*	*LSB*

Degub Frame 3 – gyroscope y and z raw data

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
DEBUG_FRAME_3_ID *dec:1922* *hex:0x782*	*Node ID*	*Data type*	*Service code*	*Message code*	*raw_Gyr_Y*		*raw_Gyr_Z*	
	5	AS_USHORT_2 (0x0D)	0	*0*	*MSB*	*LSB*	*MSB*	*LSB*

Degub Frame 4 – magnetomer x and y raw data

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
DEBUG_FRAME_4_ID *dec:1923* *hex:0x783*	*Node ID*	*Data type*	*Service code*	*Message code*	*raw_Mag_X*		*raw_Mag_Y*	
	5	AS_USHORT_2 (0x0D)	0	*0*	*MSB*	*LSB*	*MSB*	*LSB*

Degub Frame 4 – magnetomer z raw data and temperature

M1 ->HMI

ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
DEBUG_FRAME_5_ID *dec:1924* *hex:0x784*	*Node ID*	*Data type*	*Service code*	*Message code*			*raw_Mag_Z*	
	5	AS_USHORT_2 (0x0D)	0	*0*	*MSB*	*LSB*	*MSB*	*LSB*

5.2.6 Errors Management

The management of the errors refers only the NSH frames. The errors are managed by means the flag ErrorFlag in the Message Code field of the frames that the iNEMO M1 sends to the HMI:

ErrorFlag = 0 no errors, message received correctly.
ErrorFlag = -1 service code is wrong.
ErrorFlag = -2 the data of the frame is in a wrong form or is out of range.

5.3 EKF UAV

The Attitude and Heading Reference System is a combined hard- and software system that provides an estimation of the attitude and the orientation of an aircraft.

From the hardware point of view, AHRS consist of either solid-state or MEMS gyroscopes, accelerometers and magnetometers on all three axes and of an on-board processing system (MCU).

In order to estimate the attitude and the orientation of the UAV we developed an extended Kalman filter for the microcontroller of the AHRS.

The attitude and the orientation of an UAV are fully described by the Euler angles roll, pitch and yaw that are a minimal representation of the spatial orientation of a rigid body, usually adopted in aerospace, navigation, and robotics.

Roll, pitch and yaw angles in the aircraft convention are:

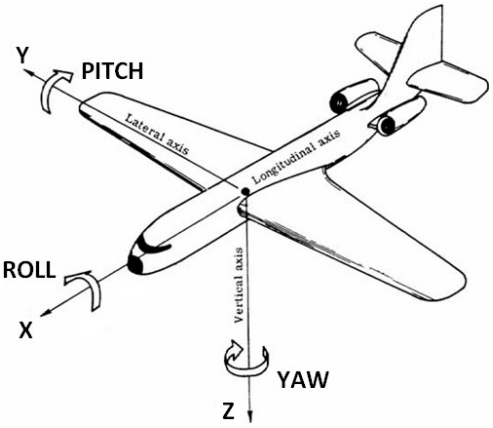

Figure 54: Euler angles aircraft convention

The overall rotation represented by RPY angles respects the following order of rotations (fixed axis) [27]:

- $R_z(\varphi)$ rotation through an angle φ (yaw) about z axis;
- $R_y(\theta)$ rotation through an angle $\theta(pitch)$ about y axis;
- $R_x(\gamma)$ rotation through an angle $\gamma(roll)$ about x axis;

$$R = R_x(\gamma)R_y(\theta)R_z(\varphi) = \begin{bmatrix} c_\theta c_\varphi & c_\theta s_\varphi & -s_\theta \\ s_\gamma s_\theta c_\varphi - c_\gamma s_\varphi & s_\gamma s_\theta s_\varphi + c_\gamma c_\varphi & s_\gamma c_\theta \\ c_\gamma s_\theta c_\varphi + s_\gamma s_\varphi & c_\gamma s_\theta c_\varphi - s_\gamma c_\varphi & c_\gamma c_\theta \end{bmatrix} \quad (72)$$

In the range $\left(-\frac{\pi}{2} < angle < \frac{\pi}{2}\right)$, the Euler angles are:

$$\left\{\begin{array}{c} \gamma = atan2(r_{23}, r_{33}); \\ \theta = atan2\left(-r_{13}, \sqrt{r_{12}^2 + r_{11}^2}\right) \\ \varphi = atan2(r_{12}, r_{11}); \end{array}\right\} \tag{73}$$

While in the range $\left(\frac{\pi}{2} < angle < 3\pi/2\right)$, the Euler angles are:

$$\left\{\begin{array}{c} \gamma = atan2(-r_{23}, -r_{33}); \\ \theta = atan2\left(-r_{13}, -\sqrt{r_{12}^2 + r_{11}^2}\right) \\ \varphi = atan2(-r_{12}, -r_{11}); \end{array}\right\} \tag{74}$$

5.3.1 QUATERNIONS

The Euler angles suffer from some weaknesses such as:

- **Gimbal lock**: the loss of one *dof* (degree of freedom) in a 3D space that occurs when 2 axes are parallel.
- **Mathematical singularities**: caused by the trigonometric function atan2 when both argument are zero.

In order to overcome these problems of the Euler angles, another representation of spatial orientation of a rigid body has been used, the unit quaternion.

Unit quaternions are an efficient and non-singular description of spatial orientation used in particular for calculations involving three-dimensional rotations such as in three-dimensional computer graphics and computer vision.

Quaternion Algebra is a number system that extends the complex numbers [30]:

$$q = (q_0, q_1, q_2, q_3) \tag{75}$$

$$q = q_0 + \boldsymbol{q} = q_0 + iq_1 + jq_2 + kq_3 \tag{76}$$

Multiplication of the basis elements is illustrated in the following table:

Table 24: multiplication of quaternion basis elements

×	1	i	j	k
1	1	i	j	k
i	i	-1	k	$-j$
j	j	$-k$	-1	i
k	k	j	$-i$	-1

5.3.1.1 Main properties

Conjugate:

$$\bar{q} = q_0 - \boldsymbol{q} \tag{77}$$

Norm:

$$N(q) = |q| = \sqrt{q\bar{q}} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \tag{78}$$

Reciprocal:

$$q^{-1} = \frac{\bar{q}}{|q|^2} \tag{79}$$

Addition:

$$q + p = (q_0 + p_0, -\boldsymbol{q} + \boldsymbol{p}) \tag{80}$$

Multiplication:

$$q * p = \left(q_0 p_0 - \boldsymbol{q} \cdot \boldsymbol{p}, q_0 \boldsymbol{p} + p_0 \boldsymbol{q} + \boldsymbol{q} \times \boldsymbol{p}\right) \tag{81}$$

Matrix representation [31]:

$$Q(q) = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \tag{82}$$

From which follows a more useful quaternions multiplication form:

$$q * p = Q(q)p = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} = \bar{Q}(p)q = \\ = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & p_3 & -p_2 \\ p_2 & -p_3 & p_0 & p_1 \\ p_3 & p_2 & -p_1 & p_0 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} \tag{83}$$

5.3.1.2 Unit Quaternions

Quaternions with norm 1 are called **unit quaternions**:

$$|q| = |q_0 + \boldsymbol{q}| = 1 \tag{84}$$

Starting from the fundamental trigonometric identity:

$$cos^2\frac{\theta}{2} + sin^2\frac{\theta}{2} = 1 \tag{85}$$

it is possible to write:

$$q_0^2 + |\boldsymbol{q}|^2 = 1 \tag{86}$$

$$cos^2\frac{\theta}{2} = q_0^2, \quad sin^2\frac{\theta}{2} = |\boldsymbol{q}|^2 \tag{87}$$

$$q = \boldsymbol{cos}\frac{\boldsymbol{\theta}}{\boldsymbol{2}} + \boldsymbol{u sin}\frac{\boldsymbol{\theta}}{\boldsymbol{2}},\ with\ \boldsymbol{u} = \frac{\boldsymbol{q}}{|\boldsymbol{q}|} \tag{88}$$

For unit quaternions the following condition must always be satisfied:

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1 \tag{89}$$

Using the unit quaternion q we can define the following operator [32]:

$$L_q(v) = q * \boldsymbol{v} * \bar{q} = (q_0^2 - \boldsymbol{q}\cdot\boldsymbol{q})\boldsymbol{v} + 2q_0(\boldsymbol{q}\times\boldsymbol{v}) + 2(\boldsymbol{q}\cdot\boldsymbol{v})\boldsymbol{q} = \boldsymbol{R}(\theta)\boldsymbol{v} \tag{90}$$

that represents a rotation of a vector $\boldsymbol{v}$ of an angle θ about q.

From the unit quaternion it is easy to get the corresponding Euler angles [31]:

$$roll = atan2\big(2 * (q0 * q1 + q2 * q3), 1 - 2 * (q1 * q1 + q2 * q2)\big) \quad (91)$$

$$pitch = \text{asin}\big(-2 * (q1 * q3 - q0 * q2)\big) \quad (92)$$

$$yaw = atan2\left(2 * (q1 * q2 + q0 * q3), 1 - 2 * (q2 * q2 + q3 * q3)\right) \quad (93)$$

From the Euler angles to the corresponding unit quaternion exist the following relationships:

$$\begin{aligned}
q0 &= \cos\left(\frac{roll}{2}\right) * \cos\left(\frac{pitch}{2}\right) * \cos\left(\frac{yaw}{2}\right) + \sin\left(\frac{roll}{2}\right) * \sin\left(\frac{pitch}{2}\right) * \sin\left(\frac{yaw}{2}\right); \\
q1 &= \sin\left(\frac{roll}{2}\right) * \cos\left(\frac{pitch}{2}\right) * \cos\left(\frac{yaw}{2}\right) - \cos\left(\frac{roll}{2}\right) * \sin\left(\frac{pitch}{2}\right) * \sin\left(\frac{yaw}{2}\right); \\
q2 &= \cos\left(\frac{roll}{2}\right) * \sin\left(\frac{pitch}{2}\right) * \cos\left(\frac{yaw}{2}\right) + \sin\left(\frac{roll}{2}\right) * \cos\left(\frac{pitch}{2}\right) * \sin\left(\frac{yaw}{2}\right); \\
q3 &= \cos\left(\frac{roll}{2}\right) * \cos\left(\frac{pitch}{2}\right) * \sin\left(\frac{yaw}{2}\right) - \sin\left(\frac{roll}{2}\right) * \sin\left(\frac{pitch}{2}\right) * \cos\left(\frac{yaw}{2}\right);
\end{aligned} \quad (94)$$

The rotation matrix associated to a unit quaternion is:

$$R(q) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_2q_1 + q_0q_3) & q_0^1 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_3q_1 - q_0q_2) & 2(q_3q_2 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \quad (95)$$

The time derivative of the unit quaternion is related to the angular velocity by the following function:

$$\dot{q} = \frac{1}{2} Q(q) \begin{bmatrix} 0 \\ \omega \end{bmatrix} = \frac{1}{2} \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{96}$$

5.3.2 UAV MODEL AND STATE PREDICTION

In order to estimate the dynamic behaviour of a system, the extended Kalman filter needs the complete knowledge of the model of the system:

$$\begin{cases} x_k = f(x_{k-1}, u_k, w_{k-1}) & (97) \\ z_k = h(x_k, v_k) & (98) \end{cases}$$

In this application we want to estimate the attitude and the orientation of the UAV, then we chose the unit quaternion that represents the spatial orientation of the UAV as state vector.

Furthermore, in order to overcome the gyroscope drift problem, we chose the biases of the gyroscope as state variables as well:

$$x = [q_0 \quad q_1 \quad q_2 \quad q_3 \quad b\omega_x \quad b\omega_y \quad b\omega_z]^T \tag{99}$$

In the discrete time and using the Euler integration method it is possible to write:

$$x_k = x_{k-1} + \dot{x}_{k-1} \Delta t \tag{100}$$

Considering the gyroscope measurements ω as inputs u and using the relationship that exists between the derivative of the unit quaternion $\dot{q}$ and the angular velocity ω, it is possible to write:

$$\dot{x}_{k-1} = \frac{1}{2} Q(q_{k-1}) \begin{bmatrix} 0 \\ \omega_{k-1} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} q_{0_{k-1}} & -q_{1_{k-1}} & -q_{2_{k-1}} & -q_{3_{k-1}} \\ q_{1_{k-1}} & q_{0_{k-1}} & -q_{3_{k-1}} & q_{2_{k-1}} \\ q_{2_{k-1}} & q_{3_{k-1}} & q_{0_{k-1}} & -q_{1_{k-1}} \\ q_{3_{k-1}} & -q_{2_{k-1}} & q_{1_{k-1}} & q_{0_{k-1}} \end{bmatrix} \begin{bmatrix} 0 \\ \omega_{x_{k-1}} - b\omega_{x_{k-1}} \\ \omega_{y_{k-1}} - b\omega_{y_{k-1}} \\ \omega_{z_{k-1}} - b\omega_{z_{k-1}} \end{bmatrix} \tag{101}$$

And substituting in the Euler integration formula, we obtain:

$$x_k = \begin{bmatrix} q_{0_k} \\ q_{1_k} \\ q_{2_k} \\ q_{3_k} \\ b\omega_{x_k} \\ b\omega_{y_k} \\ b\omega_{z_k} \end{bmatrix} = \begin{bmatrix} q_{0_{k-1}} \\ q_{1_{k-1}} \\ q_{2_{k-1}} \\ q_{3_{k-1}} \\ b\omega_{x_{k-1}} \\ b\omega_{y_{k-1}} \\ b\omega_{z_{k-1}} \end{bmatrix} + \frac{1}{2} \begin{bmatrix} q_{0_{k-1}} & -q_{1_{k-1}} & -q_{2_{k-1}} & -q_{3_{k-1}} \\ q_{1_{k-1}} & q_{0_{k-1}} & -q_{3_{k-1}} & q_{2_{k-1}} \\ q_{2_{k-1}} & q_{3_{k-1}} & q_{0_{k-1}} & -q_{1_{k-1}} \\ q_{3_{k-1}} & -q_{2_{k-1}} & q_{1_{k-1}} & q_{0_{k-1}} \end{bmatrix} \begin{bmatrix} 0 \\ \omega_{x_{k-1}} - b\omega_{x_{k-1}} \\ \omega_{y_{k-1}} - b\omega_{y_{k-1}} \\ \omega_{z_{k-1}} - b\omega_{z_{k-1}} \end{bmatrix} \Delta t \tag{102}$$

that in extended form become:

$$\begin{cases} q_{0_k} = q_{0_{k-1}} - \frac{1}{2}(\omega_{x_{k-1}} - b\omega_{x_{k-1}}) q_{1_{k-1}} \Delta t - \frac{1}{2}\left(\omega_{y_{k-1}} - b\omega_{y_{k-1}}\right) q_{2_{k-1}} \Delta t - \frac{1}{2}(\omega_{z_{k-1}} - b\omega_{z_{k-1}}) q_{3_{k-1}} \Delta t; \\ q_{1_k} = q_{1_{k-1}} + \frac{1}{2}(\omega_{x_{k-1}} - b\omega_{x_{k-1}}) q_{0_{k-1}} \Delta t - \frac{1}{2}\left(\omega_{y_{k-1}} - b\omega_{y_{k-1}}\right) q_{3_{k-1}} \Delta t + \frac{1}{2}(\omega_{z_{k-1}} - b\omega_{z_{k-1}}) q_{2_{k-1}} \Delta t; \\ q_{2_k} = q_{2_{k-1}} + \frac{1}{2}(\omega_{x_{k-1}} - b\omega_{x_{k-1}}) q_{3_{k-1}} \Delta t + \frac{1}{2}\left(\omega_{y_{k-1}} - b\omega_{y_{k-1}}\right) q_{0_{k-1}} \Delta t - \frac{1}{2}(\omega_{z_{k-1}} - b\omega_{z_{k-1}}) q_{1_{k-1}} \Delta t; \\ q_{3_k} = q_{3_{k-1}} - \frac{1}{2}(\omega_{x_{k-1}} - b\omega_{x_{k-1}}) q_{2_{k-1}} \Delta t + \frac{1}{2}\left(\omega_{y_{k-1}} - b\omega_{y_{k-1}}\right) q_{1_{k-1}} \Delta t + \frac{1}{2}(\omega_{z_{k-1}} - b\omega_{z_{k-1}}) q_{0_{k-1}} \Delta t; \\ b\omega_{x_k} = b\omega_{x_{k-1}}; \\ b\omega_{y_k} = b\omega_{y_{k-1}}; \\ b\omega_{z_k} = b\omega_{z_{k-1}}; \end{cases}$$

(103)

These are the equations for the prediction of the estimation of the extended Kalman filter.

In order to project the error covariance *P_k* ahead it is necessary calculate the Jacobian matrix A of partial derivatives of the state transition function with respect to x and the Jacobian matrix W of partial derivatives of the state transition function with respect to the noise w:

$$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T \tag{104}$$

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}_k, u_k, 0) \tag{105}$$

$$W_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}}(\hat{x}_k, u_k, 0) \tag{106}$$

Where Q is the diagonal matrix of the variance of the gyroscopes.

5.3.3 STATE CORRECTION

After the prediction phase, it is possible to use the accelerometers and magnetometers measurements to correct the predicted estimate.

The correction phase can be split in two independent steps, one step in which only the accelerometers measurements are used and one in which the correction is performed using only the magnetometers measurements.

The measurement function that links together the accelerometers measurements with the state vector, is:

$$h_{acc} = R(q) * \overline{Acc} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_2q_1 + q_0q_3) & q_0^1 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_3q_1 - q_0q_2) & 2(q_3q_2 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \begin{bmatrix} Acc_x \\ Acc_y \\ Acc_z \end{bmatrix} \tag{107}$$

The measurement function that links together the magnetometers measurements with the state vector is:

$$h_{mag} = R(q) * \overline{Mag} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_2q_1 + q_0q_3) & q_0^1 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_3q_1 - q_0q_2) & 2(q_3q_2 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \begin{bmatrix} Mag_x \\ Mag_y \\ Mag_z \end{bmatrix} \tag{108}$$

In order to evaluate the Kalman gain K_k it is necessary calculate the Jacobian matrix H of partial derivatives of the measurement function with respect to x:

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \tag{109}$$

$$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}} (\hat{x}_k, 0) \tag{110}$$

Finally, it is possible to correct the estimate with the sensors measurements:

$$x_k = x_{k-1} + K_k \big(z_k - h(x_k^-, 0)\big) \tag{111}$$

and to update the error covariance:

$$P_k = (I - K_k H) P_k^- \tag{112}$$

5.4 Chebyshev Filtering

One of the main difficulties found the test phase, was represented by the vibrations introduced by the brushless engine to the whole airframe. In particular, the measures of the accelerometers were completely distorted, causing a malfunction of the EKF algorithm. A comparison between the Z component of the accelerometer raw measures, when the motor is off (red) and when the motor is running (blue), is shown in figure 68 . This comparison test has been performed maintaining the airframe locked to the workbench.

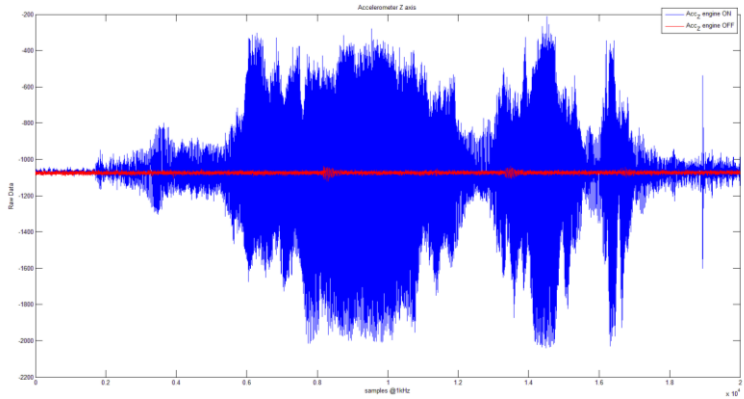

Figure 55: accelerometer measures comparison

In order to identify the frequency region of the noise caused by the engine vibrations, an analysis of the vibrations in the frequency domain has been performed.

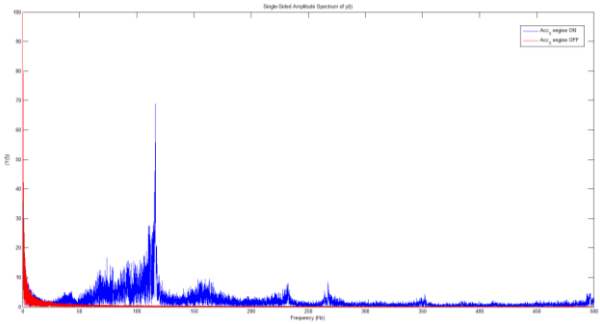

Figure 56: frequency analysis of accelerometer measures

Once the noise region has been identified, the noise has been removed by means of a filter.

In order to catch the widest range of frequencies, the Output Data Rate (ODR) of the accelerometer has been set to 1344Hz, whereas the sample rate is 1kHz.

In Figure 69 the same comparison is shown in the frequency domain.

It is clear that the main harmonic corresponds to approximately 110Hz. Therefore, to cut out the noise, the accelerometer measures were filtered with a low pass fourth-order Chebyshev filter, with a cutoff frequency of 25Hz.

The results are shown in Figure 70 . In order to ensure that the Chebyshev filter does not remove part of the dynamics of the aircraft, in this test the airframe is moving.

Moreover, in addition to the comparison between the filtered (black) and unfiltered data (green), a comparison is executed with a commercial IMU (red), in particular an MTi by Xsens[34].

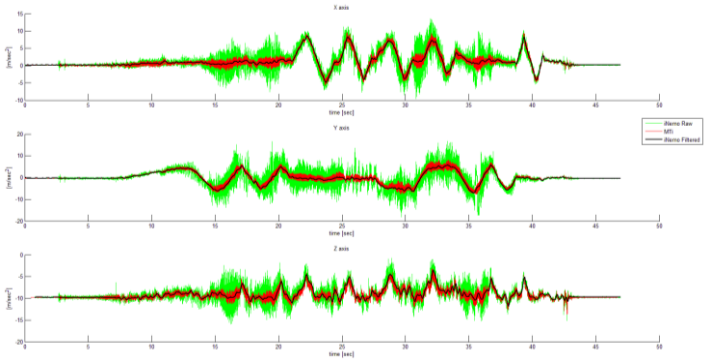

Figure 68: Euler angles comparison

5.5 Dynamic compensation

So far, only the gravitational acceleration has been considered, but when the aircraft is flying, the accelerometer measures also its dynamic acceleration that affects the performance of the EKF as well.

Without a dynamic compensation the EKF algorithm for the IMU works properly only when the system rotates and performs slow linear movements.

For the EKF algorithm only the gravity vector is useful.

Considering that to identify separately the static and dynamic component is not possible without other devices, the only way to compensate the dynamic acceleration is represented by trusting less on the accelerometer measures.

As previously described, the measures of the accelerometer act in the correction phase of the EKF algorithm.

The Kalman gain for the correction by means of the measures of the accelerometer is:

$$K_k = P_K H^T (H P_K H^T + R)^{-1} \tag{113}$$

The *R* matrix, is a diagonal matrix that contains the variances of the X,Y and Z axis of the accelerometer.

$$R = \begin{bmatrix} var\_AccX & 0 & 0 \\ 0 & var\_AccY & 0 \\ 0 & 0 & var\_AccZ \end{bmatrix} \tag{114}$$

The lower is the value of such variance, the more reliable are the measures of the accelerometers. In order to relate the *R* matrix to the dynamic acceleration, the following modification is made to the matrix:

$$R_{dyn} = \begin{bmatrix} var\_AccX & 0 & 0 \\ 0 & var\_AccY & 0 \\ 0 & 0 & var\_AccZ \end{bmatrix} \cdot Dyn_{comp} \tag{115}$$

Where *Dyn_comp* is a variable related to the dynamic acceleration, as it is shown in the following formula:

$$Dyn_{comp} = sat_{0.1}\left\{\left|\frac{norm(AccX, AccY, AccZ)}{9.81} - 1\right|\right\} \qquad (116)$$

In absence of dynamic acceleration, *Dyn_comp* is zero, conversely the value of *Dyn_comp* (saturated to 0.1) increases the unreliability of the accelerometers.

5.6 Firmware Development and Optimization

In order to make the UAV more suitable to work in unstructured environments, it is important to compensate the disturbances as quickly as possible. The IMU is used to make the feedback for the low level control of the aircraft, in order to ensure its stability and therefore it is important to maximize the frequency of the EKF. However, the EKF is a very complex algorithm, and to increase its frequency inside a 32bit microcontroller represents a hard challenge. A solution to this problem consists in optimizing the code, replacing the libraries for the matrix calculation with normal sums of products.

5.6.1 Firmware Block Scheme

The EKF timing is managed by an interrupt of a timer that runs with a frequency equal to 500 Hz. At each interrupt the accelerometer measurements are acquired, calibrated and filtered. The acquisition and calibration of the gyroscope and the magnetometer is executed every two interrupts, i.e. every 4ms.

The yellow wave in Figure 70 indicates the time necessary to execute the timer and it is possible to see that there are positive half-waves longer (where the sensors measures are acquired), alternating with positive half-waves shorter (where only accelerometer is acquired).

The reason to acquire the accelerometer at 500Hz is because in this way a better filtering can be performed.

Once all sensors data are ready, a flag is set and the EKF algorithm is executed in the main loop (Fig. 69). According to the tests, once executed the prediction phase (green wave of Figure 70), the best performance are obtained executing the correction phase (blue wave of Figure 70) one time with the accelerometer measures and one time with the magnetometer measures.

The last step are represented by a rotation of the resulting quaternion, if a ROS (reset orientation service) or a RHS (reset heading service) request is received, and the conversion of the unit quaternion in Euler angles.

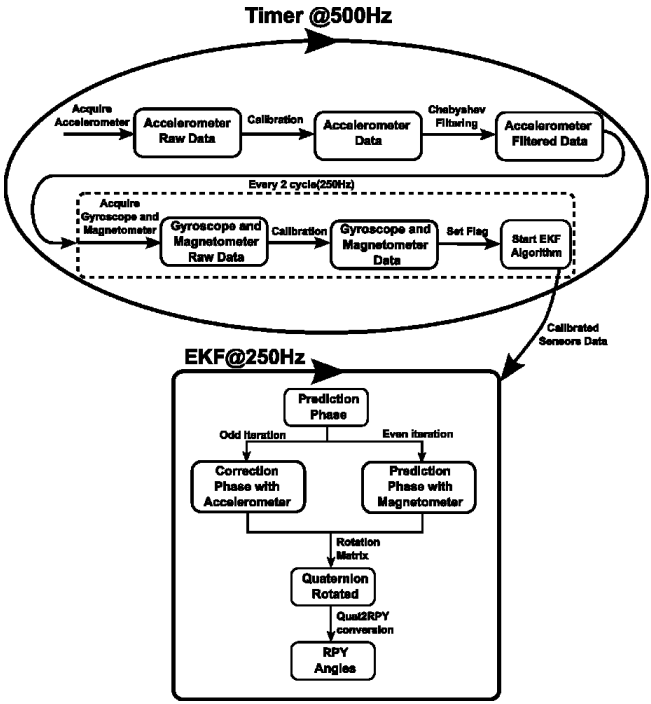

Figure 69: AHRS firmware scheme

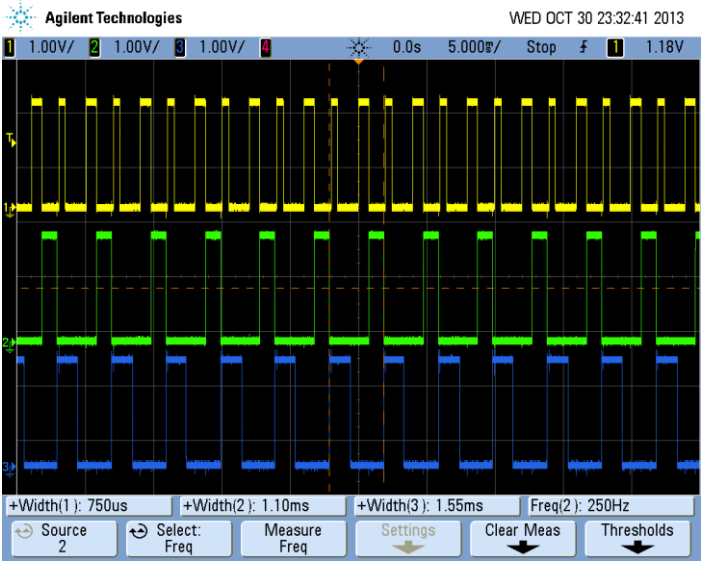

Figure 57: EKF timing analysis

5.7 Results

In order to assess the performances of the extended Kalman filter developed for estimating the Euler angles of an UAV, a comparison with a high quality AHRS is presented in this chapter.

The main features [33] [34] of the iNEMO M1 and the other AHRS used for the comparison are illustrated in the following table:

Table 25: main features AHRS comparison

AHRS	STMicroelectronics iNEMO M1	Xsens MTi
Gyroscopes	3-axis digital gyroscope, ±250°/s, ±500°/s, ±2000°/s full scales, 0.03 deg/s/√Hz Noise, 100 Hz Bandwidth 760 Hz max update rate	3-axis gyroscope, ±300°/s Full Scale, 0.05 deg/s/√Hz Noise, 40 Hz Bandwidth, 512 Hz max update rate
Accelerometers	3-axis digital accelerometer, ±2g, ±4g, ±8g, ±16g full scales, 220 ug/√Hz, 149,3 Hz Bandwidth, 1.344 kHz max update rate	3-axis accelerometer, ±50 m/s^2 Full Scale, 0.002 m/s^2/√Hz, 30 Hz Bandwidth, 512 Hz max update rate
Magnetometers	3-axis digital magnetometer, from ±1.3 gauss to ±8.1 gauss, 0.05 mGauss, 220 Hz max update rate	3-axis accelerometer, ±750 mGauss Full Scale, 0.05 mGauss, 10 Hz Bandwidth, 512 Hz max update rate
Maximum update rate processing	250 Hz	512 Hz

To perform the comparison the two AHRS boards were aligned and fixed in a rigid support as shown in the fig. 71.

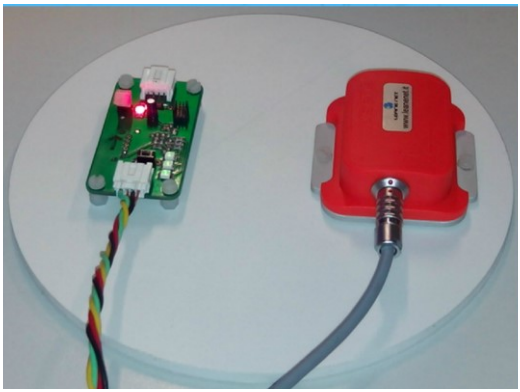

Figure 58: AHRS comparison test

Single and combined movements with slow and fast dynamics along each axis were performed and the data from the two boards were collected.

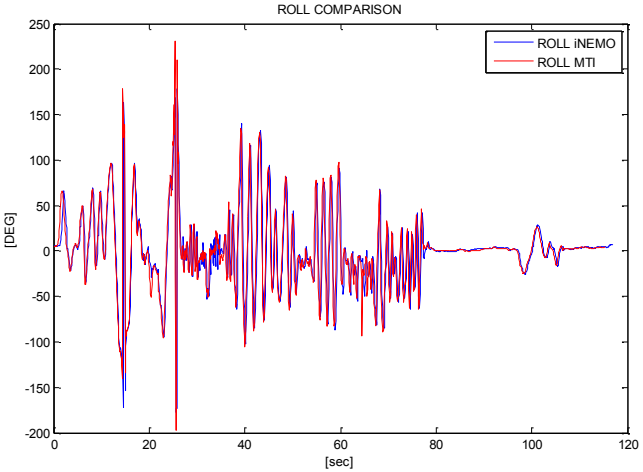

Figure 59: roll angle comparison

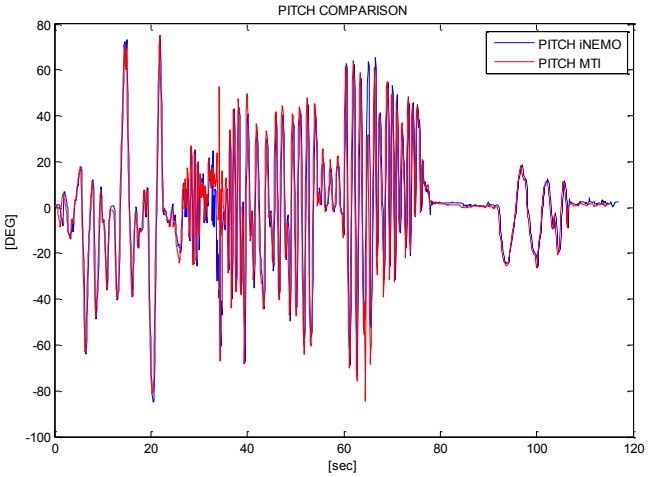

Figure 60: pitch angle comparison

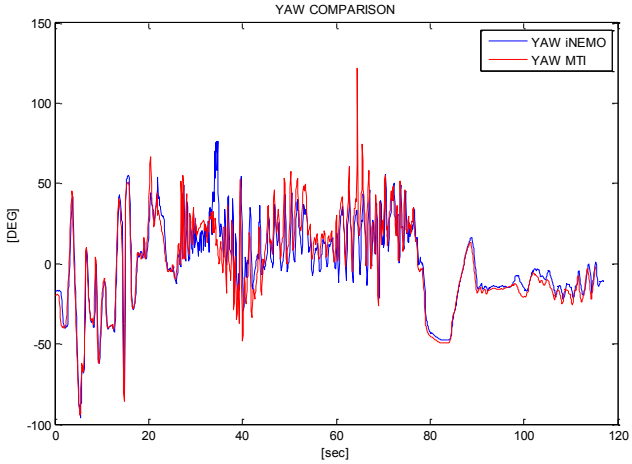

Figure 61: yaw angle comparison

As it is possible to notice in the figures of the comparison , the performances of the two AHRS boards are comparable.

The performances of the Xsens MTi AHRS [34] are shown in the following table:

AHRS	Xsens MTi
Static accuracy (roll/pitch)	<0.5 deg
Static accuracy (heading)	<1 deg
Dynamic accuracy	2 deg RMS
Angular resolution	0.05 deg

6 UGV Localization

6.1 Introduction

In this chapter a multi-sensor data fusion algorithm for the localization and self-calibration of a tracked mobile robot is described.

In order to calculate the absolute position of the robot and to estimate the values of the parameters used by the odometry (wheels radii and wheelbase), data from a GPS, two optical encoders and an inertial platform are fused together by an Extended Kalman Filter. The inertial platform allows the mobile robot to be localized even when the quality of the GPS signal is low for example near high buildings or trees.

The EKF algorithm has been tested both on a simulator and on a tracked mobile robot.

6.2 The model of the robot

Odometry is the classical technique to calculate the position of a wheeled robot. It allows the robot to be tracked through integration of the measurements of wheels movements.

The tracked mobile robot can be schematically represented as in Fig. 75:

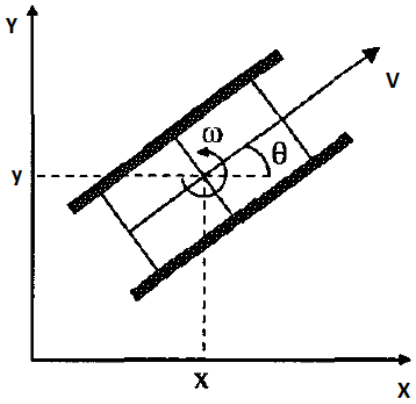

Figure 62: tracked robot representation

The robot is characterized by a position *(x,y)* and an orientation *θ(t)*. The kinematic model of the robot is:

$$\begin{cases} \dot{x}(t) = V(t)\cos\theta(t) & (117) \\ \dot{y}(t) = V(t)\sin\theta(t) & (118) \\ \dot{\theta}(t) = \omega(t) & (119) \end{cases}$$

Where *V(t)* is the linear velocity of the mobile robot while *ω(t)* is its angular velocity.

The kinematic model of the mobile robot transforms the measurements of the wheels movements into measures of the vehicle motion.

Unfortunately odometry is affected by many inaccuracies, such as wheel slipping, uncertainty in the odometry parameters (wheel radii and wheelbase), misalignment of the wheels and finite encoder resolution and sampling frequency [35].

The errors caused by the wheels slipping cannot be estimated by the encoders and hence cannot be compensated. On the other hand the error caused by the uncertainty on the odometry parameters could be reduced if a good calibration is performed.

Calibration is the problem of estimating the parameters of the robot model from sensors measures. This estimation can be very difficult because it depends on the conditions and the environment where the robot has to move; therefore, to perform the calibration of the robot parameters periodically, is useful.

If the measures from the GPS are not available, for example when a big rock or a building reduce the number of visible satellites of the GPS, and the robot is using wrong parameters, the robot can get lost very quickly.

The EKF can concurrently localize the robot and estimate the odometry parameters-.

The discrete-time kinematic model is:

$$\begin{cases} x(k) = x(k-1) + V(k-1)\cos\theta(k-1) & (120) \\ y(k) = y(k-1) + V(k-1)\sin\theta(k-1) & (121) \\ \theta(k) = \theta(k-1) + T\omega(k-1) & (122) \end{cases}$$

Where T is the sampling time.

Two optical encoders are used to measure the angular displacement of the robot wheels.

V(k) and *ω(k)* are calculated by the following equations:

$$\begin{cases} V(k-1) = \dfrac{R_1\Delta\theta_1(k-1) + R_2\Delta\theta_2(k-1)}{2T} & (123) \\ \omega(k-1) = \dfrac{R_1\Delta\theta_1(k-1) - R_2\Delta\theta_2(k-1)}{LT} & (124) \end{cases}$$

Where $\Delta\theta_1(k-1)$ and $\Delta\theta_2(k-1)$ are respectively the angular displacements of the right and left wheels during the interval [T(k-1),T(k)]; R_1 and R_2 represent the radii of the right and left wheels; L denotes the wheelbase.

Replacing the equations (123) (124) in (120-122), the mathematical model of the robot becomes:

$$\begin{cases} x(k) = x(k-1) + \dfrac{R_1\Delta\theta_1(k-1) + R_2\Delta\theta_2(k-1)}{2T}\cos\theta(k-1) & (125) \\ y(k) = y(k-1) + \dfrac{R_1\Delta\theta_1(k-1) + R_2\Delta\theta_2(k-1)}{2T}\sin\theta(k-1) & (126) \\ \theta(k) = \theta(k-1) + T\dfrac{R_1\Delta\theta_1(k-1) - R_2\Delta\theta_2(k-1)}{LT} & (127) \end{cases}$$

6.3 The Localization algorithm

In order to estimates the position and orientation of the mobile robot the extended Kalman filter algorithm uses the encoder measurements in the predictive phase and the GPS and IMU measurements in the corrective phase.

The odometry parameters are considered as state variables in the EKF and are estimated together with the localization variables. Therefore the state vector is defined as it follows:

$$X(k) = [x(k)y(k)\theta(k)R_1(k)R_2(k)\ L(k)]^T \tag{128}$$

The mathematical model becomes:

$$\begin{cases} x(k) = x(k-1) + \dfrac{R_1(k-1)\Delta\theta_1(k-1) + R_2(k-1)\Delta\theta_2(k-1)}{2T}\cos\theta(k-1) \\ y(k) = y(k-1) + \dfrac{R_1(k-1)\Delta\theta_1(k-1) + R_2(k-1)\Delta\theta_2(k-1)}{2T}\sin\theta(k-1) \\ \theta(k) = \theta(k-1) + T\dfrac{R_1(k-1)\Delta\theta_1(k-1) - R_2(k-1)\Delta\theta_2(k-1)}{L(k-1)T} \\ R_1(k) = R_1(k-1) \\ R_2(k) = R_2(k-1) \\ L(k) = L(k-1) \end{cases} \tag{129}$$

The encoder measurements are considered as input variables of the kinematic model:

$$U(k) = [\Delta\theta_1(k)\ \Delta\theta_2(k)]^T \tag{130}$$

while the position and the orientation of the robot represent the output vector:

$$Z(k) = HX(k) = [x(k) \quad y(k) \quad \theta(k)\ ]^T \tag{131}$$

where H is the following matrix:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{132}$$

Finally, the kinematic model of the robot can be simply written as:

$$\begin{cases} X(k) = f\big(X(k-1), U(k)\big) & (133) \\ Z(k) = HX(k) & (134) \end{cases}$$

After the prediction of the state, the predicted covariance matrix of the state error is calculated as follows:

$$P_k^- = J_{x_k} P_{k-1} J_{x_k}^T + J_{u_k} C_u J_{u_k}^T + Q \quad (135)$$

And represents the uncertainty of the estimated state; C_u is the covariance matrix of the noise that corrupts the input; Q is a diagonal matrix that represents the uncertainty of the model; J_{x_k} and J_{u_k} are the Jacobians of the system:

$$J_{x_{[i,j]}} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}_k, u_k, 0) \quad (136)$$

$$J_{u_{[i,j]}} = \frac{\partial f_{[i]}}{\partial u_{[j]}}(\hat{x}_k, u_k, 0) \quad (137)$$

In the matrix Q the terms related to R_1 and R_2 and L are equal to zero because the estimated parameters have not to change when the GPS is not available.

The Kalman gain is calculated as follows:

$$K_k = P_k^- H^T (H P_k^- H^T + R_k)^{-1} \quad (138)$$

Where R_k is the covariance matrix of the noise of the GPS and the IMU at time k obtained from the datasheet and from the GPS string:

$$R_k = \begin{bmatrix} var\_x\_gps & 0 & 0 \\ 0 & var\_y\_gps & 0 \\ 0 & 0 & var\_yaw\_imu \end{bmatrix} \quad (139)$$

The measurement vector is also obtained directly from the GPS and IMU measurements:

$$Z(k) = \begin{bmatrix} z_1(k) \\ z_2(k) \\ z_3(k) \end{bmatrix} = \begin{bmatrix} x_{GPS}(k) \\ Y_{GPS}(k) \\ \theta_{IMU}(k) \end{bmatrix} \quad (140)$$

The predicted measurement vector is:

$$Z(k) = HX^{-}(k) \quad (141)$$

The measurement vector is compared to the predicted measurement vector in order to obtain the estimated state $X(k)$ using the Kalman Gain:

$$X(k) = X(k-1) + K_k\big(Z(k) - HX^{-}(k)\big) \quad (142)$$

Finally the state covariance matrix P is updated.

6.4 GPS Accuracy and Precision

Accuracy and precision are often used to describe how good is the position acquired by the GPS receiver. A distinction should be made between accuracy and precision [36].

Accuracy is the degree of closeness of an estimate to its true, but unknown value while the precision is the degree of closeness of observations to their means. Figure 76 illustrates various relationships between these two parameters. The true value is located at the intersection of the crosshairs, the center of the

shaded area is the location of the mean estimate, and the radius of the shaded area is a measure of the uncertainty contained in the estimate.

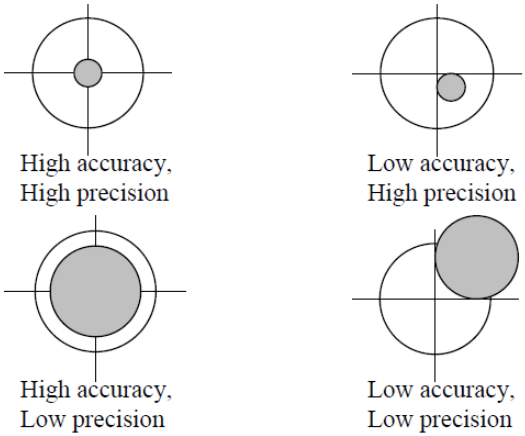

Figure 63: GPS accuracy and precision

The most common measures of 2D accuracy are the Distance Root Mean Squared (DRMS) and the Circular Error Probability CEP.

6.4.1 Distance Root Mean Squared (DRMS)

DRMS is a single number that expresses 2D accuracy. In order to compute the DRMS of horizontal position errors, the standard errors (σ) from the known position in the directions of the coordinate axis are required.

DRMS is the square root of the average of the square errors which is defined as follows:

$$DRMS = \sqrt{\sigma_x^2 + \sigma_y^2} \tag{143}$$

Standard errors (σ) of estimated coordinates (x, y) of each point being positioned can be predicted from the corresponding variances on the diagonal of the covariance matrix.

6.4.2 Circular Error Probability (CEP)

CEP refers to the radius of a circle in which 50% of the values occurs, i.e. if a CEP of 5 meters is quoted then 50% of horizontal point positions should be within 5 meters of the true position. The radius of the 95% is often quoted and the term R95 used. R95 is CEP with the radius of the 95% probability circle.

Table 1 describes the most commonly used position accuracy measures and their probability.

Table 26: common position measures

Accuracy Measures	Formula	Probability	Definition
DRMS	$\sqrt{\sigma_x^2+\sigma_y^2}$	**65%**	The square root of the average of the squared horizontal position errors.
2DRMS	$2\sqrt{\sigma_x^2+\sigma_y^2}$	**95%**	Twice the DRMS of the horizontal position errors.
CEP	$0.62\sigma_y+0.56\sigma_x$ (Accurate when $\sigma_y/\sigma_x>0.3$)	**50%**	The radius of circle centered at the true position, containing the position estimate with probability of 50%.
R95	R($0.62\sigma_y+0.56\sigma_x$) (R=2.08, when $\sigma_y/\sigma_x=1$)	**95%**	The radius of circle centered at the true position, containing the position estimate with probability of 95%.

6.4.3 Dilution of Precision (DOP)

The DOPs provide a simple characterization of the user-satellite geometry. DOP is related to the volume formed by the intersection points of the user-satellite vectors, with the unit sphere centered on the user. Larger volumes give smaller DOPs. Lower DOP values generally represent better position accuracy. The role of DOP in GPS positioning, however, is often misunderstood. A lower DOP value does not automatically mean a low position error. The quality of a GPS-derived position estimate depends upon both the measurement geometry as represented by DOP values, and range errors caused by signal strength, ionospheric effects, multipath etc.

DOP is the ratio of the positioning accuracy to the measurement accuracy:

$$\sigma = DOP \cdot \sigma_0 \tag{144}$$

Where:

σ_0 is the measurement accuracy, and
σ is the position accuracy.

6.4.4 Variance of the GPS noise

Manuals of receivers include static and RTK specifications. The datasheet of the Leica Viva GNSS - GS10 Receiver reports [37]:

Accuracy (rms) with Real-Time-Kinematic (RTK)[1]	
Standard of compliance	Compliance with ISO17123-8
Single Baseline (< 30 km)	Horizontal: 8 mm + 1 ppm (rms) Vertical: 15 mm + 1 ppm (rms)
Network RTK	Horizontal: 8 mm + 0.5 ppm (rms) Vertical: 15 mm + 0.5 ppm (rms)
Accuracy (rms) with Post Processing[1]	
Static (phase) with long observations	Horizontal: 3 mm + 0.1 ppm (rms) Vertical: 3.5 mm + 0.4 ppm (rms)
Static and rapid static (phase)	Horizontal: 3 mm + 0.5 ppm (rms) / Vertical: 5 mm + 0.5 ppm (rms)
Kinematic (phase)	Horizontal: 8 mm + 1 ppm (rms) / Vertical: 15 mm + 1 ppm (rms)

Figure 64: datasheet of GPS Leica

Baseline is the distance from the rover to the reference station and "ppm" indicates part-per-million (1E-6) of the baseline.

If, for example, the baseline is 1.500 meters, as it is likely at the Cittadella Universitaria di Catania, the static accuracy σ_0 is:

$$\boldsymbol{Static\ Phase\ with\ Long\ observations\ Accuracy}\ (rms) = 0.003 + 0.1E(-6) * 1{,}500 = 0.0032\ meters \tag{145}$$

$$\boldsymbol{Rapid\ Static\ Accuracy}\ (rms) = 0.003 + 0.5E(-6) * 1{,}500 = 0.0037\ meters \tag{146}$$

$$\boldsymbol{Kinematic\ Phase\ Accuracy}\ (rms) = 0.008 + 1E(-6) * 1{,}500 = 0.0095\ meters \tag{147}$$

$$\boldsymbol{Single\ Baseline\ Accuracy}\ (rms) = 0.008 + 1E(-6) * 1{,}500 = 0.0095\ meters \tag{148}$$

$$\boldsymbol{RTK\ Accuracy}\ (rms) = 0.008 + 0.5E(-6) * 1{,}500 = 0.0088\ meters \tag{149}$$

The accuracy with which positions can be determined is not just a function of the measurement precision, and the appropriate modelling of biases. It is also a function of the satellite(s) - receiver(s) geometry.

Finally the variance of the noise of the GPS in the extended Kalman filter is chosen in the following way:

$$
\begin{aligned}
&\textbf{\textit{switch}}\ fix\_quality\_GPS \\
&case\ 0: \\
&\qquad variance = 1000; \\
&case\ 1: \\
&\qquad variance = \sigma_0 * GPSDOP(k); \\
&case\ 2: \\
&\qquad variance = \sigma_0 * GPSDOP(k); \\
&case\ 4: \\
&\qquad variance = \sigma_0 * GPSDOP(k);
\end{aligned}
$$

6.5 Results

In order to assess the performance of the localization algorithm developed, the results of an experiment with the tracked robot are presented.

In the two first figures both the two coordinates of the position of the robot during the experiment are illustrated, the measured one in red and the estimated one in blue respectively:

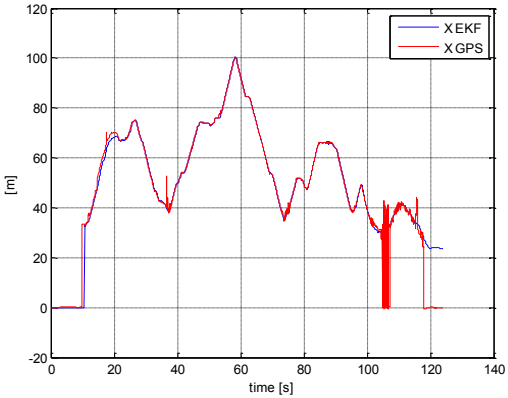

Figure 78: X position comparison

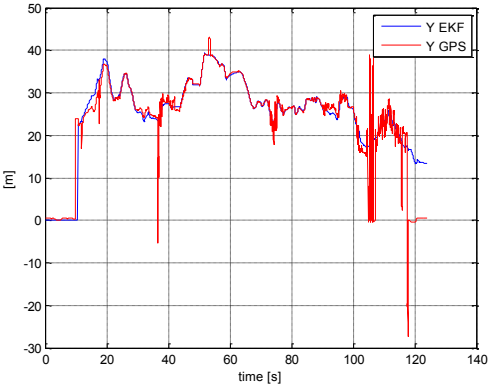

Figure 79: Y position comparison

As it is possible to notice in the figures 78 79, the extended Kalman filter is able to localize the robot pretty well, apart from some samples in which the quality of the GPS signal (Fig. 80) is low.

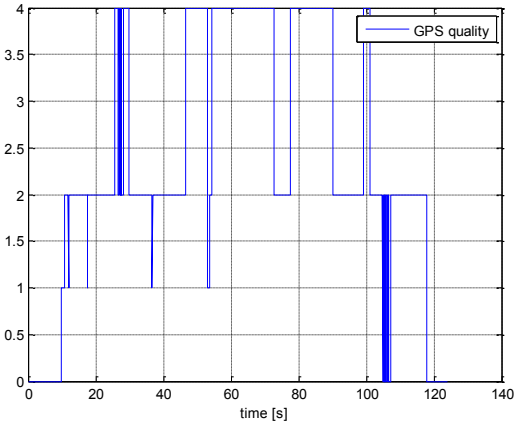

Figure 65: GPS signal quality

The difference between the position measured by the GPS and the position estimated by the EKF is:

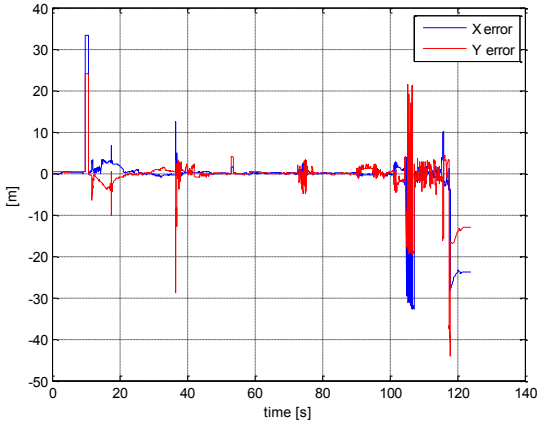

Figure 66: position error

It is possible to notice that in the range time in which the quality of the GPS signal is good the error is in the order of a few meters.

The comparison between the estimated heading angle and the one measured by means of the IMU, is illustrated in the following figure:

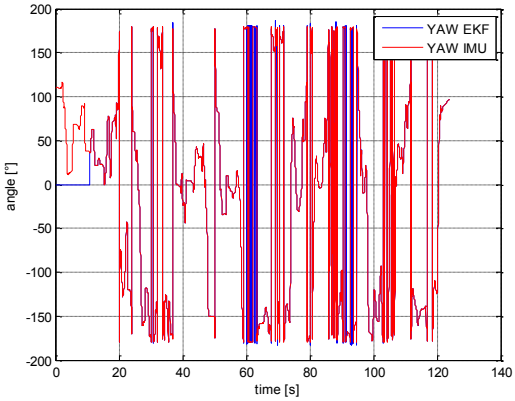

Figure 67: YAW angle comparison

The difference between the yaw angle measured by the IMU and the yaw angle estimated by the EKF, is:

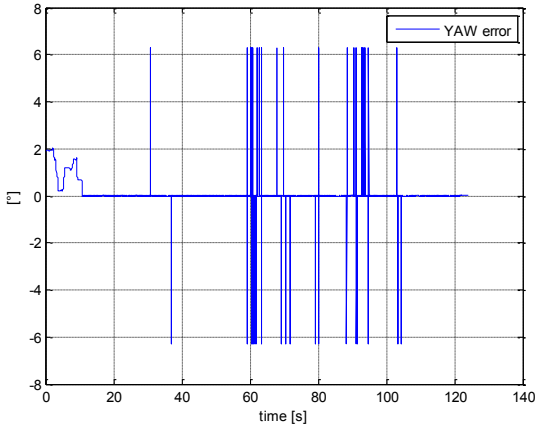

Figure 68: YAW error

It is possible to notice that in the range time in which the quality of the GPS signal is good the error is in the order of a few degrees.

The extended Kalman filter developed estimates also some parameters such as the lengths of the radii of the left and the right wheels of the robot.

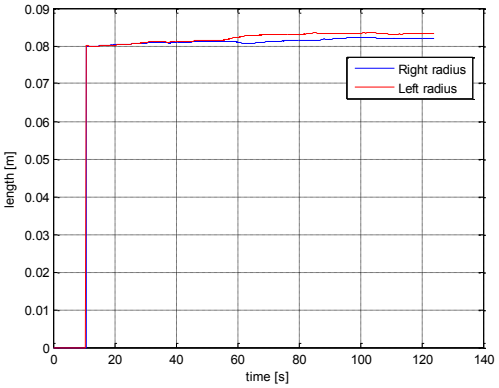

Figure 69: left and right wheel radii

In addition to the radii of the left and the right wheels of the robot, the algorithm estimates also the length of the wheelbase of the robot.

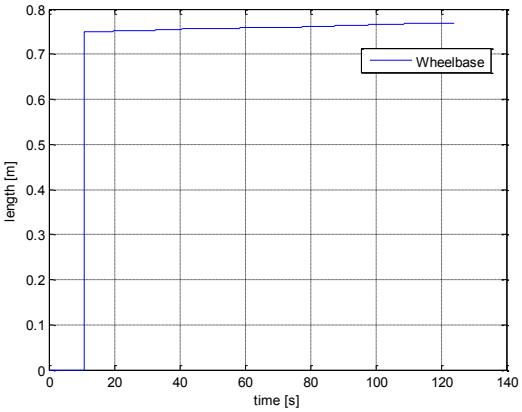

Figure 70: wheelbase

7 Conclusion

In this thesis, multi-sensor data fusion, and in particular its application in robotics, is discussed.

Thanks to technology progresses such as faster and better integrated processing units and increasingly miniaturized and precise sensors, Robotics is experiencing a period of significant development.

Power electronics is also contributing to a transformation in the field of robotics. In order to present Power electronics implications in robotics, its usage is first presented in this thesis. As it is described in the first chapter, power electronics devices allow the use of distributed instead of centralized control in industrial robotics.

In the second chapter, multi-sensor data fusion and Recursive Bayesian Filtering techniques such as Kalman Filter and Particle filter are described. These techniques allow the estimation of the state of a dynamic system like a robot.

The multi-sensor data fusion applications on which this thesis is focused on are the estimation of the pose of industrial robots, the attitude and heading reference system of an UAV and the localization of a UGV.

In all the applications presented, inertial sensors are used in combination with other kind of sensors such as magnetometers, GPS and encoders.

In order to convert the sensors raw data to meaningful information, inertial sensors and magnetometers need a calibration procedure, as described in the third chapter.

In the fourth chapter, two new methods, based on low-cost Micro Electro-Mechanical Systems (MEMS) Inertial Sensors, to estimate joint angles of a manipulator are presented.

In one method an Extended Kalman Filter is used, while in the second a Particle Filter. The experimental results presented show that, with both methods, the positioning error of the end effector of manipulator never reaches 2 cm and then it is possible to say that, in non-precision applications, both methods can be used instead or in parallel to the encoder, as system for sensing the joints angles.

Despite some clues of possible advantages of the Particle Filter over the Extended Kalman Filter, such as the possibility to work with not only Gaussian systems, for this application the Extended Kalman filter has always shown better performance than the Particle Filter.

In order to estimate the attitude and the heading of an UAV, a multi-sensor data fusion method based on the Extended Kalman Filter is described in the fifth chapter. The EKF estimates the Unit Quaternion associated to the Euler angles of the aircraft and the biases of the gyroscopes to compensate the drift of the measurement integration. Experimental results show that the performance of the AHRS developed are comparable with the performance of a high quality commercial AHRS.

The last application described in this thesis is the localization of an Unmanned Ground Vehicles when the quality of the GPS signal is low for example near high buildings or trees. The experimental results show that by using an inertial platform it is possible to localize an UGV even when the GPS data are not reliable.

8 Bibliography

[1] http://www.st.com/web/en/resource/technical/document/datasheet/CD00274138.pdf, STMicroelectronics SPIMD20, Datasheet — production data, 2012

[2] Chen, Z., "Bayesian Filtering From Kalman Filters to Particle Filters, and Beyond," technical report, Adaptive System Lab., McMaster University, Hamilton, ON, Canada, 2003.

[3] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," Department of Computer Science at the University of North Carolina at Chapel Hill Tech. report TR 95-041, 2006.

[4] S. Thrun, W. Burgard and D. Fox , "Nonparametrics Filters," in Probabilistic Robotics, MIT Press, 2005, pp. 96–112.

[5] TA0343, STMicroelectronics, Technical article, Everything about STMicroelectronics' 3-axis digital MEMS gyroscopes, 2011

[6] AN3192, STMicroelectronics, Application note, Using LSM303DLH for a tilt compensated electronic compass, 2010

[7] Torgny Brogardh, "Present and future robot control development—An industrial perspective", Annual Reviews in Control 31 (2007) 69–79

[8] Tan, J.T.C.; Feng Duan; Ye Zhang; Kato, R.; Arai, T., "Safety design and development of human-robot collaboration in cellular manufacturing," Automation Science and Engineering, 2009. CASE 2009. IEEE International Conference on , vol., no., pp.537,542, 22-25 Aug. 2009

[9] Bonaccorso, F.; Cantelli, L.; Muscato, Giovanni, "An Arc Welding Robot Control for a Shaped Metal Deposition Plant: Modular Software Interface and

Sensors," Industrial Electronics, IEEE Transactions on , vol.58, no.8, pp.3126,3132, Aug. 2011

[10] Xing-tao Wang; Xing-guang Duan; Qiang Huang; Hong-hua Zhao; Yue Chen; Hua-tao Yu, "Kinematics and trajectory planning of a supporting medical manipulator for vascular interventional surgery," Complex Medical Engineering (CME), 2011 IEEE/ICME International Conference on , vol., no., pp.406,411, 22-25 May 2011

[11] F. Ghassemil, S. Tafazoli, P. D. Lawrence and K. Hashtrudi-Zaad, " Design and Calibration of an Integration-Free Accelerometer-Based Joint-Angle Sensor", IEEE Transactions on Instrumentation and Measurement, VOL. 57, NO. 1, JANUARY 2008

[12] P. Cheng, B. Oelmann, "Joint-Angle Measurement Using Accelerometers and Gyroscopes—A Survey", IEEE Transactions on Instrumentation and Measurement, VOL. 59, NO. 2, FEBRUARY 2010.

13] Daniel Kubus and Friedrich M. Wahl, "A Sensor Fusion Approach to Angle and Angular Rate Estimation", 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, September 25-30, 2011. San Francisco, CA, USA

[14] P. Axelsson, M. Norrlöf, "Estimation of Orientation and Position of an Accelerometer Mounted to an Industrial Manipulator", 50th IEEE Conference on Decision and Control 2011.

[15] C. Wang, W. Chen, and M. Tomizuka, "Robot End-effector Sensing with Position Sensitive Detector and Inertial Sensors", 2012 IEEE International Conference on Robotics and Automation, RiverCentre, Saint Paul, Minnesota, USA May 14-18, 2012.

[16] M. Quigley, R. Brewer, Sai P. Soundararaj, V. Pradeep, Q. Le, and A. Y. Ng," Low-cost Accelerometers for Robotic Manipulator Perception", Intelligent

Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on 18—22 Oct. 2010.

[17] S. P. Won, W. W. Melek, and F. Golnaraghi, "A Kalman/Particle Filter-Based Position and Orientation Estimation Method Using a Position Sensor/Inertial Measurement Unit Hybrid System", IEEE Transactions on Industrial Electronics, VOL. 57, NO. 5, MAY 2010

[18] Guanglong Du; Ping Zhang; Di Li, "Human–Manipulator Interface Based on Multisensory Process via Kalman Filters," Industrial Electronics, IEEE Transactions on , vol.61, no.10, pp.5411,5418, Oct. 2014

[19] G. G. Rigatos, "Particle Filtering for State Estimation in Nonlinear Industrial Systems," IEEE Transactions on Instrumentation and Measurement, VOL. 58, NO. 11, pp. 3885-3900 November 2009.

[20] Du, G.; Zhang, P., "Online Serial Manipulator Calibration based on Multi-sensory Process via Extended Kalman and Particle Filters," Industrial Electronics, IEEE Transactions on , no.99, 2014.

[21] R. Karlsoon, M. Norrlof, "Sensor fusion for position estimation of an industrial robot," Department of Electrical Engineering Linkoping University, SE-581 83 Linkoping, Sweden , Jun 2004 .

[22] R. Karlsoon, M. Norrlof, "Bayesian Position Estimation of an Industrial Robot using Multiple Sensors," IEEE Conference on Control Applications 2004, Taipei, Taiwan.

[23] S. Ying, C. Wenkai, "Particle Filter Applied to a Manipulator with Two Flexible Links," Proceedings of the 27th Chinese Control Conference July 16-18, 2008, Kunming,Yunnan, China, pp. 705-708.

[24] P. Axelsson, R. Karlsoon, M. Norrlof, "Bayesian state estimation of a flexible industrial robot," Control Engineering Practice 20 (2012), pp.1220–1228.

[25] X. Ren, G. Xue, "Position Estimation for Manipulators Based on Multisensor Fusion," Proceedings of the 31st Chinese Control Conference, July 25-27, 2012, Hefei, China, pp. 5082-5087.

[26] P. Roan et al., "Manipulator state estimation with low cost accelerometers and gyroscopes", IEEE/RSJ IROS 2012, pp. 4822 - 4827, Oct.2012.

[27] B. Siciliano, L. Sciavicco, L. Villani and G. Oriolo , "Kinematics," in Robotics: Modelling, Planning and Control, Springer, 2009, pp. 61–64.

[28] CANAerospace,Wikipedia, http://en.wikipedia.org/wiki/CANaerospace

[29] Michael Stock, CANAerospace specifications, Interface specification for airborne CAN applications V 1.7, NASA, 2006

[30] Quaternion, Wikipedia, http://en.wikipedia.org/wiki/Quaternion

[31] James Diebel, Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors, Stanford University, 2006

[32] J. B. Kuipers, Quaternions and Rotation Sequences, Princeton University Press, 1999.

[33] STMicroelectronics iNEMO system-on-board Datasheet, http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00056715.pdf , 2013

[34] xSens MTi AHRS, https://www.xsens.com/wp-content/uploads/2013/11/mti-leaflet1.pdf

[35] Borenstein, J. Liqiang Fag; "Mcasment and carnction of systematic Odometry errors in mobile robots", IEEE Transactions on Robotics and Automation, Volume: 12, Issue: 6, Dec. 1996 pp.: 869 -880

[36] GPS Position Accuracy Measures, NovAtel, http://www.novatel.com/assets/Documents/Bulletins/apn029.pdf , 2003

[37] Leica Viva GNSS - GS10 Receiver datasheet, http://www.leica-geosystems.com/downloads123/zz/gpsgis/Viva%20GS10/brochures-datasheet/Leica_Viva_GS10_DS_en.pdf