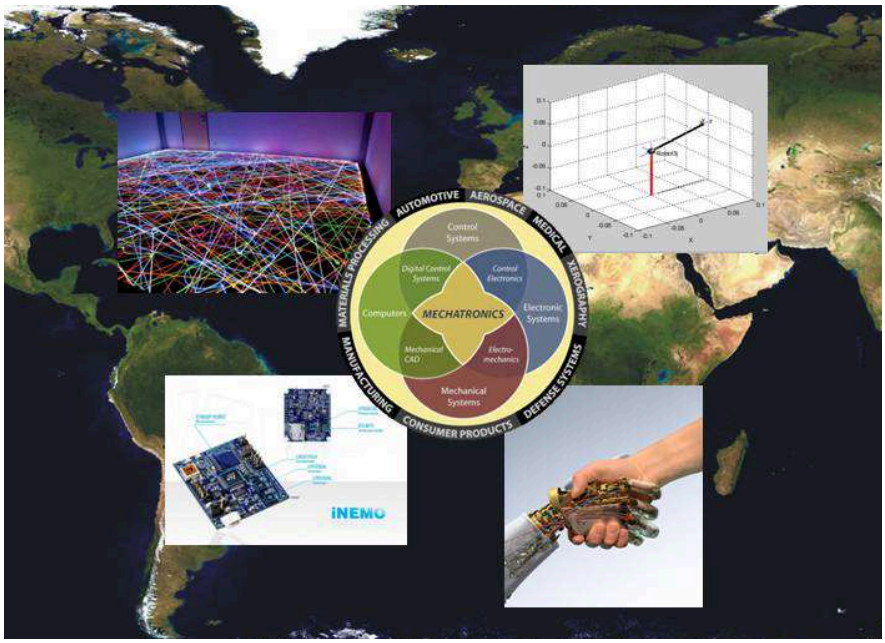
# University of Catania

DIEEI

ELECTRONIC ENGINEERING, AUTOMATION AND CONTROL OF
COMPLEX SYSTEMS ENGINEERING - INTERNATIONAL DOCTORATE XXIV CYCLE

# Developing methods and algorithms of sensor fusion by IMUs applied to service robotics




Tutor

Prof. Giovanni Muscato

Coordinator

Prof. Luigi Fortuna

# University of Catania

DIEEI

ELECTRONIC ENGINEERING, AUTOMATION AND CONTROL OF
COMPLEX SYSTEMS ENGINEERING - INTERNATIONAL DOCTORATE XXIV CYCLE

# Developing methods and algorithms of sensor fusion by IMUs applied to service robotics


Tutor
Prof. Giovanni Muscato

Coordinator
Prof. Luigi Fortuna

INDEX

ABSTRACT
OVERVIEW
CHAPTER I – STATE OF THE ART AND INTRODUCTION
1.1 – Inertial Measurement Unit
1.2 – Kalman Filter and Particle Filter
1.3 – Data Filtering
1.4 – Main Hardware: ST-iNEMO IMU Board
[Gyroscopes; Geomagnetic Unit: Accelerometer and Magnetometer; Barometer; Thermometer]
CHAPTER II – ORIENTATION ESTIMATION
2.1 – Introduction
2.2 – Accelerometer Filtering
2.3 – Compass Calibration
2.4 – Method of Orientation Estimation
2.5 – Experiment Result
2.6 – Robot-iNemo interaction results
2.7 – Statistical analysis
CHAPTER III – NAVIGATION AND TRAYECTORY ESTIMATION
3.1 – Sensor fusion algorithm for dead reckoning
3.2 – 3D Navigation: pressure sensor
3.3– Trajectories reconstruction
3.4 – Reconstruction of planar trajectory per vehicle: test
3.5 – Bump detection
CONCLUSION
ACKNOWLEDGMENTS
BIBLIOGRAPHY
PUBLICATIONS
COURSE/CONFERENCE/PARTNERSHIP

## ABSTRACT

The importance of research in Inertial Navigation Systems (INS) has been growing in recent years. Usually the IMU is used in inertial navigation, such as UAV, AGV, AUV, but it is also used in games, human movement reconstruction (the use of sensors in the studies of human movement is now quickly gaining importance as a promising alternative to video capture systems laboratories), entertainment, etc. Often IMU is used in association with GPS or other sensors to estimate trajectory or for navigation as well as localization. In literature, there are many examples using Kalman Filter or EKF for this aim. It can also be considered to use the implementation of an algorithm in the use of the robot. Moreover, when speaking about an inertial platform there is also a Kalman Filter as a good algorithm providing good results.

The work described concerned the development of systems and algorithms, or new approaches to existing systems to bring robotics to everyday life and to lower costs of implementation of certain devices in industrial processes, or to review some progresses in the light of improvement of technology.

We used the IMUs (Inertial Measurement Unit) and MEMS devices such as accelerometers, gyroscopes, but also temperature and pressure sensors for localization and navigation. Through the use of more accurate sensors and to the growing potential of the new microcontrollers, we have been able to implement algorithms to process and filter data the more quickly and with fewer steps and in some cases to be able to find good solutions at the expense of precision, but in the interest of processing speed.

These sensors have been designed as an aid to existing sensors or for new applications such as three-dimensional localization in a building using

the pressure or for safety, in industry, eg for the monitoring of movements of a robotic arm.

Finally, since usually the inertial navigation uses GPS data to correct inertial data, this excludes the GPS spoofing; in other words that someone deliberately alters the signal in such a way that it provides the same values specifically wrong to hack satellite systems installed in the cars.

The IMU used in this work is the iNEMO™ board, an inertial measurement unit developed by STMicroelectronics. It runs a sophisticated sensor fusion algorithm (attitude heading reference system) to provide static and dynamic orientation and inertial measurements. This 10-DOF inertial system integrates five different sensors and has a size of 4x4 cm.

# OVERVIEW

The purpose of this work of research has been to develop algorithms, methods or new applications for the use of inertial sensors or MEMS (Micro Electro Mechanical System), in order to make less expensive some applications and/or support the traditional systems both for the field of the final consumer - in other words the common man -, and in the field of industrial applications. The importance of research in the inertial navigation systems (INS) has grown in recent years; in fact, usually the IMUs (Inertial Measurement Unit) are used for inertial navigation systems, such as UAV, AGV, AUV; but also for games, for human movement reconstruction, entertainment, etc… IMUs are often used in conjunction with GPS or other sensors to assess the trajectory or the navigation and localization. IMUs usually use MEMS devices, such as accelerometers, gyroscopes, but also Temperature and Pressure Sensors which integrate on-board a firmware that manages the input data. Moreover, each IMU has a suitable algorithm that can be programmed to provide a different output according to the application.

Technological progress has allowed to have sensors more and more at low cost and at a higher level of integration; this progress, combined with the growing possibility of microprocessor calculation - typically integrated directly into the same sensor board - has given the ability to create undefined scenarios for their use (even in areas where previously was unthinkable their use, as for example in the industrial robotics).

Starting from the sensors that were kindly provided by the STMicroelectronics worldwide company - with whom there has been a cooperative relationship, and who I thank for the support - several

approaches have, therefore, developed whose common theme has been the use of inertial platform to know location, position and orientation of an object, regardless of whether it is a vehicle, a mobile robot or an industrial robot, with particular focus to the application to be implemented.

Undoubtedly basic has been the study of the state of art, and technological advancement considered as initial step, along with the processing of sensor data that could not be taken as raw as they were, but that they had to be treated appropriately, and possibly filtered. The first chapter also highlights that, because the filtering and data analysis introduced the problem of delay, as when using a filter there is always a delay due to filtering and processing. In some cases it has been tried to directly use the raw-data and develop an algorithm that, under certain parameters, could lead to results approximate but fine for our goal.

The work is certainly not directed to the pure research, but to an industrial/business research which could have had an outlet, a job, a policy of continuous cooperation development between research and industry, such as the current Partnership University and ST, which has produced these results. One of these is the development of bump detection that can be useful both for industrial robots and for domestic robots. An example is the robotic vacuum cleaners, now widely used in homes, using the mechanical touch sensors to detect collisions with objects and walls of the room. In place of these accelerometers can be used, as shown in the third chapter, which makes the robot more powerful and reliable both from the structural point of view - because they are less subject to break than the mechanical contact sensors - and also allow implementing on the robot board control other functions by

using the same sensor. At the same time in an industrial environment IMUs can be used for estimate and orientation of the robot end-effector, to handle it in an intuitive and simple way.

The first chapter is a brief overview of the techniques studied and tested and the material and hardware used in the development of the algorithms presented. It also refers to the importance of filtering and data analysis and their correction.

The second chapter examines in detail the algorithm developed to guide an industrial robot using inertial platform. Usually to move such a robot it is necessary to define a system of coordinates, the end point of the robot tool, and in which trajectory we want to move it. The robot is programmed in the control unit by qualified personnel and then started up, the robot will do the same thing a million times in the same way, with a very high degree of accuracy in repeatability. Thinking now to replace the encoders with the IMUs on the links at the current state of MEMS technology is unthinkable because we would have not the same degree of precision or repeatability with the degree of tenths of a millimetre that they have (even if there is already a patent in this sense). What has been done instead, in collaboration with Eng. Giacomo Spampinato at Malarden University in Sweden, is to have an easier use of the robot; in fact we can use the IMU the same way as a joystick, so that any worker not in possess of an advanced training could use it easily and intuitively. Another use of the IMU in industrial robotics may be directed to safety in the workplace and in workings. Using statistical analysis we can see if there is an abnormal function of the robot. In fact, by placing one or more inertial platforms on the links and on the end-effector, it can be determined whether during their movement there is a different behavior

than expected. The implementation of this feature would allow, by using a safety routine, to stop the robot immediately in the event that it impacts an object, or for a technical problem, or for the failure of some component. So we tried to have a multilateral vision of the types of problems to be addressed to: with similar algorithms or methodologies we can face and solve different problems in different fields and applications, and also find new ones.

Finally in the third chapter we go back to use the GPS with inertial platforms. Also in this case alternatives for the correction of errors develop independently and we attempt to assess the reliability of the GPS itself. Again, we look at the applications and industrial developments, such as many insurance companies now use the GPS to quantify the number of kilometres per year, and thereby allowing to know the risks associated with that customer and how much has to pay, or it is often installed as anti-theft tracking device. Unfortunately there is already a countermeasure, there are devices that simulate a GPS signal of excellent quality, shielding the correct signal to the receiver and making believe that the car stopped at one point or it is on a trajectory that is actually different from the real one. In this sense we can understand the usefulness of the reconstruction of the path travelled by a vehicle using only the inertial platform to a level that ensures also how measured by GPS.

In conclusion we tried to use an inertial platform and MEMS sensors, developing new approaches to solve old problems, and new methods to face current issues. Moreover, new algorithms and solutions and/or simplifications of the old ones were designed, so as to permit their use economically, efficiently and functionally in the domestic and industrial

applications, in the perspective of interaction and integration between research and industry from which both could receive benefit.

# CHAPTER I - STATE OF ART AND INTRODUCTION

## 1.1 INERTIAL MEASUREMENT UNIT

The localization of an object inside a space, internal or external, is a problem under continuous evolution which, during the last years has known an exponential development of its applications and used methodologies. The growing availability of low-cost sensors has made easier the approach to this issue, driving research towards innovation of methodologies of sensor fusion. Being able to make use of information deriving from different sensors shows the value of any more precise and accurate magnitude. This combined with the increasing reduction in the size of the available sensors has led to the development of applications designed to be integrated into any mobile device. Devices such as accelerometers or gyroscopes have become part of common language and have aggressively invaded any device on the market, finding space even on platforms whose use until a few years ago was unthinkable. We can clearly see, therefore, how important is to maximize in parallel these devices' performance and potentialities. On the other hand, the use of more sensors for the valuation of any physical quantity, as well as being complex in the computational and theoretical point of view, it results to have not few difficulties with the their integration into a single platform. It would be sufficient to note that each sensor has a consumption of energy which, as much as we could optimize it, it shall affect the duration of an independent power system embedded proportionally to the number of actual sensors. Moreover, the chance to obtain accurate valuations through low cost sensors without the aid of more sophisticated sensors

(and therefore more expensive) is undoubtedly attractive and has provided a great boost to the optimization of applications to this sense. On this point of view the developed work lies, and what will be outlined below. Through special mathematical tricks we tried to rebuild at the best the information deriving from an inertial sensor platform provided by ST Microelectronics, refining them as much as possible, in order to derive an estimate of position using the fewest possible sensors. For example using only the accelerometer data it is possible to rebuild the position of objects and / or obstacles in a room, using it as a "bump sensor" and then doing SLAM.

An IMU applied to a body provides output data related to the various dynamic measured magnitudes: linear and angular accelerations, magnetometer data, but also temperature, pressure, etc. The development of these quantities, joint with the knowledge of GPS coordinates allows us to reconstruct the movement of the body, with a better accuracy respect to that given by the single GPS, even when the fix of this is not always available.

The use of inertial sensors into aircraft or land vehicles is nowadays a great used resource, especially for purposes of direct georeferencing or mobile mapping in the LIDAR and, more generally in the field of geodetic survey. The growing use of these sensors, combined with the development of existing satellite constellations (GPS and GLONASS), to the next appearance of the European Galileo system, and to the possible application of modern means of treatment and estimation of signals (eg the Kalman filter), all them get to guide industrial research on their use in personal and low cost navigation applications.

In the past years along with the research activity an integrated software environment has been designed and developed for processing the output data from the IMU ST, iNEMO$^{TM}$ with the obtained information from the navigation sensor GPS TESEO STA8058 EVB Rev. 03; in order to obtain accurate information on the positioning of the vehicle where the system is installed on. The above said vehicle we can imagine being able to rebuild the path flown without human aid. Many of the activities here described have been carried out at the IMS Systems Labs of STMicroelectronics in Catania as well as in the laboratories of the University of Catania DIEE.

## 1.2 KALMAN FILTER AND PARTICLE FILTER

The Kalman filter is an efficient recursive filter that evaluates the state of a dynamic system from a series of measures subjected to noise. For its intrinsic characteristics it is a great filter for noise and disturbances acting on the system zero-mean Gaussian. It is used as an observer state, such as loop transfer recovery (LTR) and as a parameter identification system.
The filter gets its name from Rudolf E. Kalman, although Thorvald Nicolai Thiele and Peter Swerling actually developed a similar algorithm before. Stanley Schmidt is generally recognized to have been the first to develop a practical implementation of a Kalman filter. This was during a visit by Kalman to the NASA Ames Research Center, during which Schmidt saw the applicability of the Kalman's ideas to the problem of estimation of trajectories for the Apollo program, ending with including the program on the Apollo on-board computer. The filter was developed in scientific articles by Swerling (1958), Kalman (1960) and Kalman and Bucy (1961).

Several types of Kalman filter were subsequently developed, starting from the original formulation of Kalman, now called the Kalman easy filter; some examples are the extended filter by Schmidt, the information filter and various square root filters developed by Bierman, Thornton and many others. It can be considered a Kalman filter also the phase-locked loop (or PLL), an electronic circuit now used in countless applications, from radio broadcast to computers, from data transmission to sensor fusion.

The Kalman filter uses the dynamic model of a system (for example, the physical laws of motion), known control inputs and system measures (eg from sensors), to make an assess of the variable quantities of the system (its State) which is better than the assess obtained by making a single measurement.

All measurements and calculations based on models are to a certain extent, estimates. Noise in sensor data, approximations in the equations that describe how to change the system and external factors that are not taken into account, introduce some uncertainty on the deduction of the values of the state of a system. The Kalman filter mediates the state of a system with a new measure using a weighted average. The purpose of the introduction of the weights is that the values with a better (ie smaller) uncertainty estimated are "trusted". The weights are calculated from the covariance, a measure of uncertainty (estimated) on the prediction of the system state. The weighted average result is a the new estimate of the state that lies between prediction and measurement of the state, and, therefore, it is a better estimate. This process is repeated for each step, with the new estimate (and its covariance) used for the next iteration of the prediction. Then the Kalman filter operates recursively and considers

only the "best guess" - not the whole story – of the state of a system to calculate the next state

When the actual computation for the filter is performed, the state estimate and covariance are coded on matrices to manage the multiple dimensions involved using mathematics known on matrix calculus. Generally it is preferred writing the Kalman filter distinguishing two distinct phases: prediction and update.

The prediction step uses the state estimate of the previous time step to produce a state estimate at the current time step. This state estimate is also known as the a priori estimate because, although it is a state estimate at the current time step, it does not include information for measuring the current time step. During the upgrade, the current a priori forecast is combined with information of the current measure to refine the assessment of the state. This best estimate is called the a posteriori state estimate.

In general, the two phases are alternate, with the expectation of state advancement to the next planned measurement and the update incorporating this measure. However, this is not necessary, if the measure is not available for some reason, the update phase may be ignored and only predictions can be run. Likewise, if several independent measurements are available at the same time, several phases of renovation can be made (usually with different observation matrices $H_k$).

Prediction phase

State estimation a priori

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k\mathbf{u}_k$$

Covariance estimation a priori

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^{\mathrm{T}} + \mathbf{Q}_k$$

Update

Innovation or measurement residual

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1}$$

Innovation or covariance residual

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k$$

Kalman excellent gain

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^{\mathrm{T}} \mathbf{S}_k^{-1}$$

Updated estimation of the a posteriori state

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k$$

Updated estimation of the a posteriori covariance

$$\mathbf{P}_{k|k} = (I - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1}$$

This formula for the estimation of the state and covariance update only applies for the Kalman gain.

Within the representation of the localization, the particle representation has a set of advantages: we can use the sensory characteristics, motor dynamics, and arbitrary distributions of noise. These are universal functions which approximate the equation for the probability density, which can overcome very restrictive assumptions than other approaches, focus computational resources in most relevant areas by sampling in proportion to a posteriori density.

Creating a software algorithm implementing them is quite simple. The use of particle filters is due to the difficulty of effectively representing relationships describing the system in all its features. The use of pure probabilistic filters requires the knowledge of the probability density function form, which are often extremely complex and defined on a

multidimensional domain. In order that the particle approach results in a viable option, we should consider that: the choice of the proposal function is crucial for the generation of probability functions; a large number of samples causes more computational power absorption; the samples can degenerate if one of them assumes a much greater weight than others and this would cause a loss of diversity that would weaken the algorithm.

After building the system and perceptual models, it becomes possible making the special particle filter for the application we should implement. Particularly the distribution to be made must allow predicting the next state by starting from the current state. In this case having an estimate which takes into account, not only controls, but also observations, would increase our chances of success. This kind of feature is as powerful as difficult to create: finding a mathematical function that intersects the observable and controllable system parts is very complex.

The solution, as for the perception model, might be algorithmic. In this case, however, the development could not be off-line, as the position estimates are one of the steps of the localization cycle, so they must be done frequently and in real time.

While the values of the sensors can be queried only when you must assign a weight to the particle filter samples, it is essential to access the sensor values with a very high frequency: the predictor step in the tracking algorithm is based on the robot translational and rotational speed. If these values are not correct, particle can follow completely wrong trajectories.

The particle filter has been introduced for the application of spoofing GPS, where an attempt was made to have a reliability of the GPS through the use of the inertial platform.

## 1.3 DATA FILTERING

An INS, Inertial Navigation System, is basically a calculator estimating the position of a vehicle compared to a known system, based on measurements of a set of sensors installed on the vehicle. Independence from external structures (eg. Satellites) ensures continuity on position tracking. The used sensors as inertial navigation system are accelerometers, acceleration detectors along a specific direction, and gyroscopes, measuring the rotation speed. The measuring unit consists of a triad of accelerometers and one of gyroscopes arranged in according to an orthogonal system. By integrating the acceleration and rotation components, position and current layout of the body are calculated.

The INS systems require knowledge of data such as position and layout starting and proceed by integrating the differential equations of motion (Equations of navigation).

However, this approach is affected by a large error component due to sensors drift, which, accumulated over time, lead to an overall degradation of the position estimate. For this reason, inertial navigation systems have been integrated with external sources, such ass the GPS through the use of various sensor fusion techniques.

The most effective and best known of these is the one which integrates information from the GPS with the equations of motion characterized by sensorial data through the use of a special Kalman filter.

This approach, as much as accurate and satisfactory is, however, it leaves unresolved the issues related to navigation in environments where the signal is not available. Hence the need to develop new techniques that make the estimation of position in these cases more accurate and fast.

For the estimate of the position of an object in space starting from measurements provided by the inertial sensors, it is necessary to digress on position body estimate subject to a one-way motion from information on accelerations impressed on it.

This seemingly trivial problem is complicated significantly by the presence of any error which can not be compensated directly or indirectly through information from other sensors.

The worst case is the use of a single sensor for the estimation of a position in the space because it is subject to large and growing percentage of error, which increasing the time cause increasing uncertainty to spatial location of the object that you want to estimate the position. In a very general way, we can say that the speed of an object subject to an acceleration a(t) in the interval ($t$-$t_0$) and having an initial speed v0 is given by:

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

Once obtained the speed, the position at time t is given by:

$$p(t) = p_0 + \int_{t_0}^{t} v(t)\,dt$$

by replacing the speed value obtained in the first equation, in the expression on the position we get:

$$p(t) = p_0 + \int_{t_0}^{t} v_0\,dt + \iint_{t_0}^{t} a(t)\,dt$$

Which may be rewritten as:

$$p(t) = p_0 + v_0(t - t_0) + \iint_{t_0}^{t} a(t)dt$$

Looking at the above expression we note how the position from moment to moment depends on an integral double of acceleration. This simple but effective observation highlights how each error in measuring acceleration from any sensor is integrated twice and propagated in time greatly and irreparably impairing the final estimate of the position of the object. The errors that are subject to the acceleration measurements carried out with MEMS-type accelerometers are well known in the literature. These are mainly due to the following types:

1. Dynamic and highly variable bias over time
2. White noise overlapping the output signal
3. Juxtaposing of components of the gravitational acceleration with the signal
4. Oscillations in the absence of external stimuli, too

It seems so obvious how the integration of raw data from the accelerometer, gyroscope and other MEMS is totally inadequate for the estimation of the location or development of any navigation algorithm. The adoption of appropriate mathematical tools can solve this problem in part by offering some alternatives to sensor fusion.

To adequately understand the extent of drift that is subject the estimate of the position performed through a signal subject to a random type of inconvenience (white noise with Gaussian distribution), it is sufficient to make a proper numerical simulation in Matlab® - a program of numerical calculation produced by Mathworks which can provide a wide range of functions and tools for the representation and manipulation of time series, and any type of system. Such step, however apparently superfluous, allows a clear view of what should be the response of our

system to estimate a specific impulse and what is actually the reply received.

First you need to model the dynamics of the system, this can be easily done using the canonical form of state of a discrete linear system of the type:

$$\dot{X} = AX + BU$$

In the particular case just start from the physical laws that govern the dynamics of the body in the event of uniformly accelerated motion:

$$\begin{cases} p(t) = p_0 + v_0 t + \frac{1}{2} a t^2 \\ v(t) = v_0 + at \end{cases}$$

Bringing all in matrix form and rewriting the system in discrete form we get:

$$\begin{bmatrix} p_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_k \\ v_k \end{bmatrix} + \begin{bmatrix} 1/2\, T^2 \\ T \end{bmatrix} a_k$$

This simple form represents our system composed by state variables coinciding with position and speed at time k +1 commanded from the input U coinciding with the acceleration at time k. Transcribing the system under consideration in Matlab and providing undisturbed acceleration we can see the correct behavior of the system. For example, by imposing an acceleration with a sinusoidal pattern described in figure:

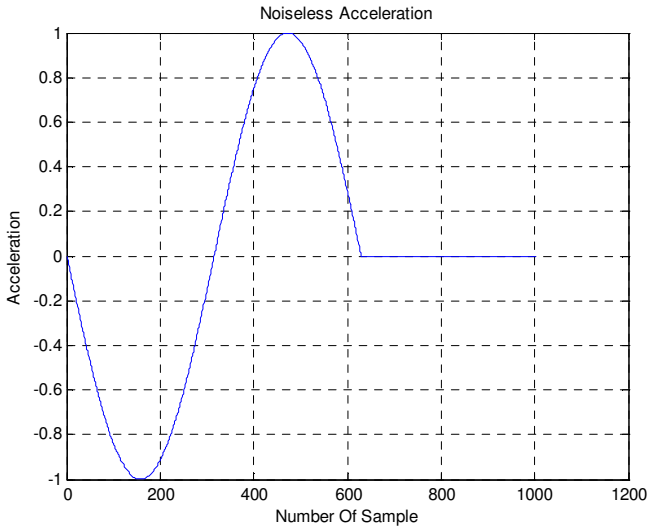



Figure 1. Example of unperturbed input

A similar input produces the following output:

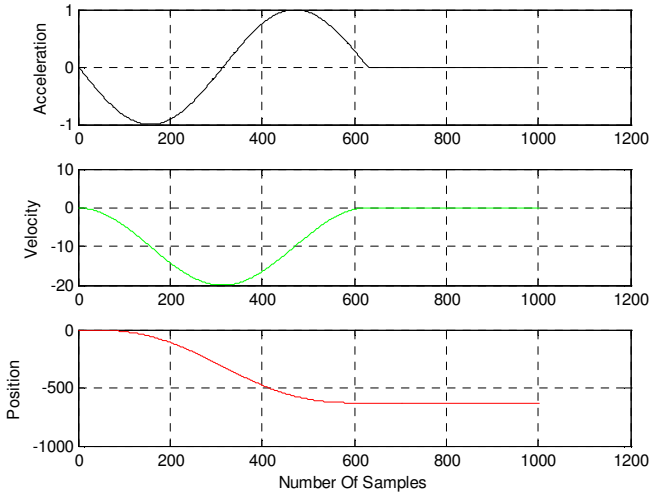



Figure 2. Response of the system to the unperturbed input

By a brief visual analysis is shown how the application of a sinusoidal acceleration produces a variation of speed to be brought back to zero after a whole period of the input sinusoid; this implies a total body move

that may be certified to a constant value over time at the end of the application of the acceleration signal of sinusoidal.

To understand how a signal affected by noise during acceleration goes to affect the estimate of the position in the system under investigation, we must produce a signal similar to the previous one, which is overlapped from a white undisrupted noise, and use it as input, as done with the undisrupted signal. If we proceed thus disrupting the sinusoid input lightly with a white noise variance 0.2m/s2, we obtain the following result:

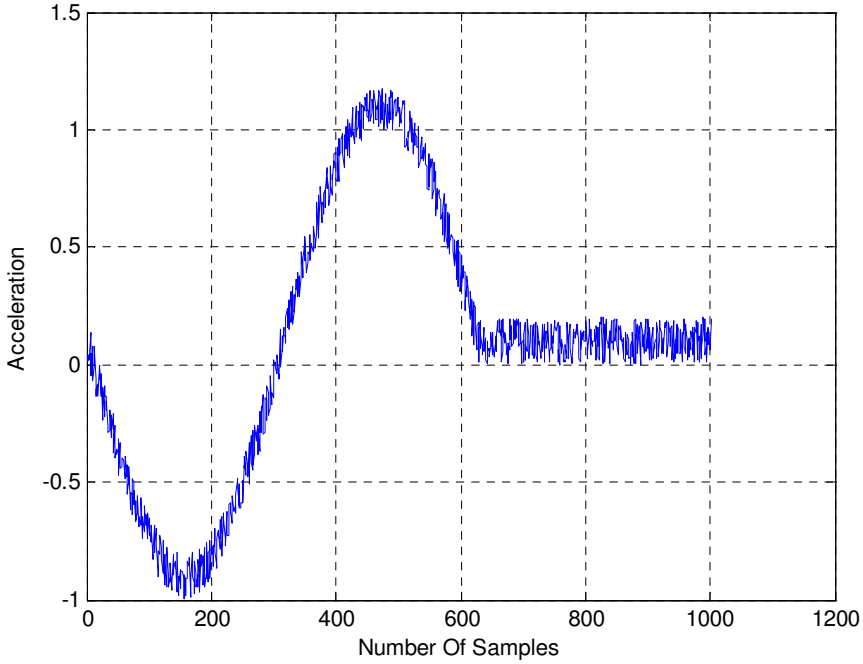



Figure 3. Example of input perturbed by Gaussian white noise

This produces in output:

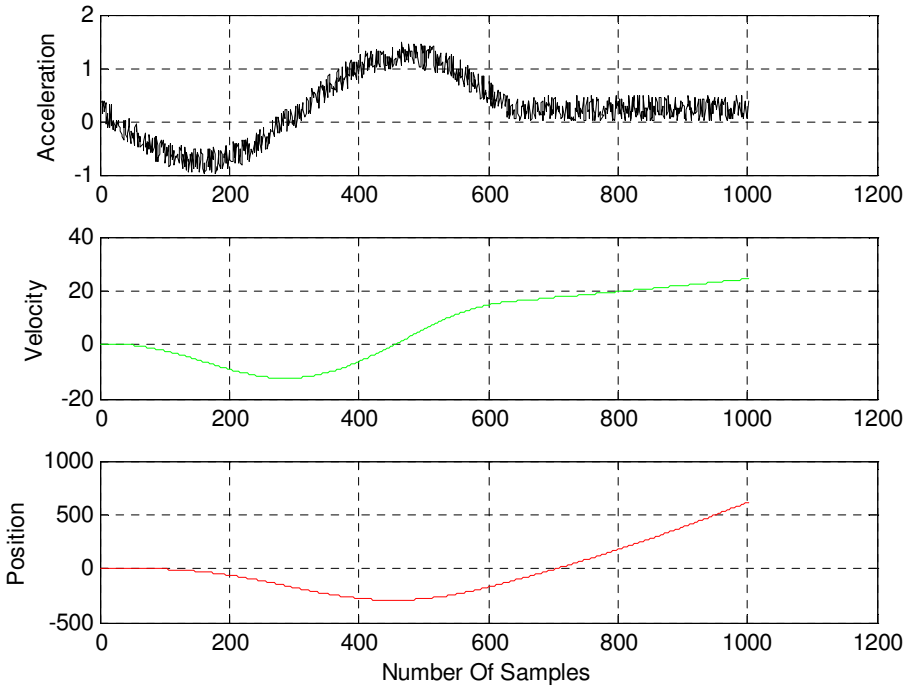



Figure 4. Response of the system to the perturbed input

As we can easily imagine the noise introduced by the previous figure has produced a drift speed the system which leads to the detection of a shift that does not exist. This is even clearer by comparing individual velocities and positions in absence of noise (true velocity) and in presence of noise (Measured velocity):

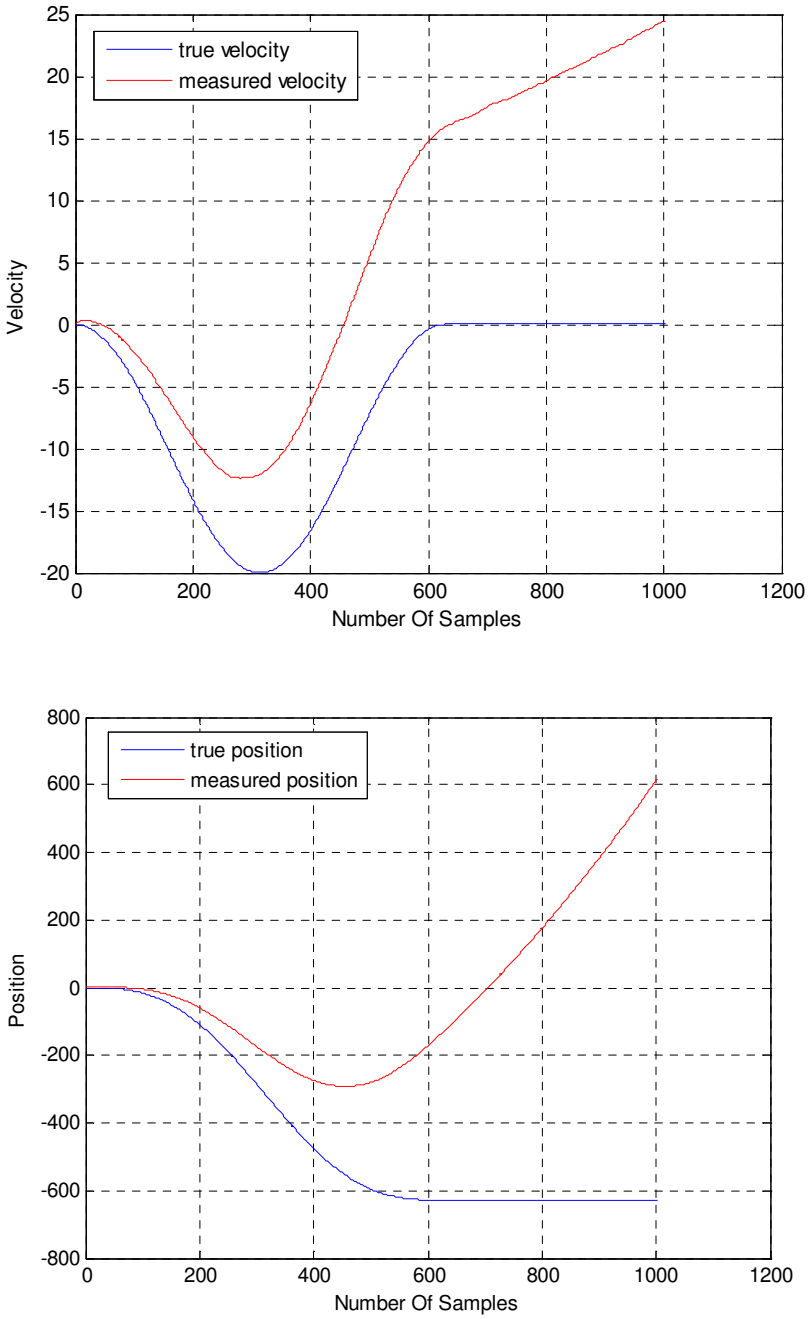



Figure 5. Comparison between the system outputs at the presence of noisy and not noisy input

It is, therefore, needed a digital filtering of the input data allowing to remove more accurately as possible the noise overlapping the signal. In this sense, there are several possibilities producing significantly different results in terms of computational burden and, hence, the time taken to reconstruct the signal. The time parameter in applications - as real-time - becomes a key discriminator in order to make the correct choice and the more appropriate filtering method. Below are some types of filters that can be applied to this examined case in order to reconstruct the original signal. As we shall see later digital filtering is closely related to the methodology for data integration.

The first type of filter under exam belongs to the family of FIR filters, they are also named moving average since their output is simply a sort of weighted average of input values.

These filters present the major drawback of approximating the ideal filter perfectly only for very large values of N, tending infinite. This means they have a very high computational cost because for each sample input, N multiplications and N additions must be calculated, which makes unattractive the use of FIR filters in real-time applications.

In its simplest form a moving average filter is the following relation:

$$y(n) = \frac{1}{N}\sum_{i=0}^{N} x(n-i)$$

By applying this simple iteration to a perturbed signal we get a result with accuracy increasing proportionally to the number of samples used.

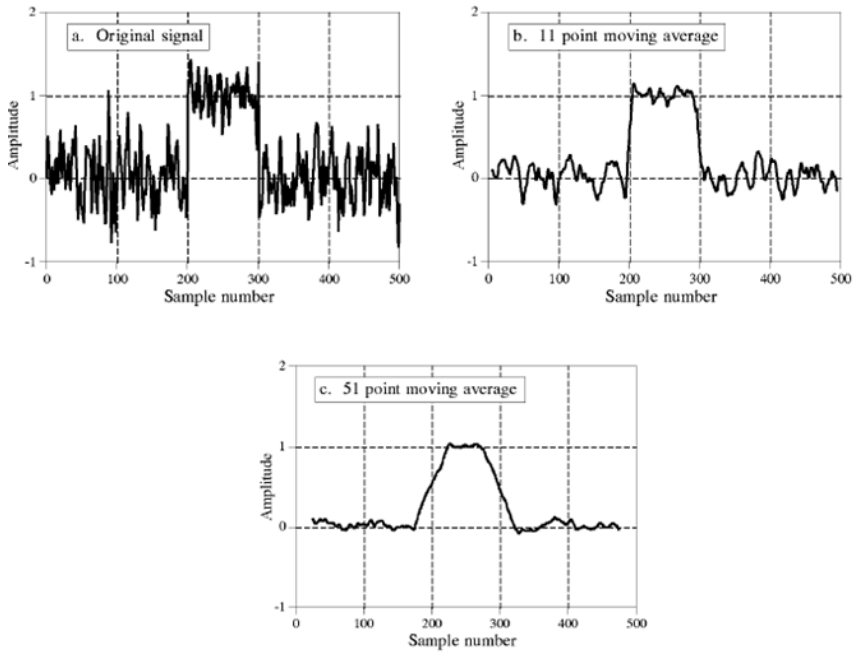



Figure 6 Examples of moving average digital filtering

This method, no matter how simple, has many drawbacks due to the increasing phase delay output of the filter at the growing of the used samples and to the extensive computational operations necessary to perform all the filtering. An excellent alternative to the methodology above is provided by the SMOOTHING of the data in a progressive manner. It is also in this case a moving average filter, but reformulated in order to clear the phase delay and reduce the computational order. The smoothing, however, must be performed several times in order to be effective, but on the other hand, the number of steps necessary for its "convergence" appear to be relatively few.

In Matlab has already implemented a smoothing function that allows filtering the data with the following methods:

| method | Description |
|---|---|
| 'moving' | Moving average (default). A lowpass filter with filter coefficients equal to the reciprocal of the span. |
| 'lowess' | Local regression using weighted linear least squares and a 1st degree polynomial model |
| 'loess' | Local regression using weighted linear least squares and a 2nd degree polynomial model |
| 'sgolay' | Savitzky-Golay filter. A generalized moving average with filter coefficients determined by an unweighted linear least-squares regression and a polynomial model of specified degree (default is 2). The method can accept nonuniform predictor data. |
| 'rlowess' | A robust version of 'lowess' that assigns lower weight to outliers in the regression. The method assigns zero weight to data outside six mean absolute deviations. |
| 'rloess' | A robust version of 'loess' that assigns lower weight to outliers in the regression. The method assigns zero weight to data outside six mean absolute deviations. |

Figure 7. Summary table of the types of filtering in MATLAB

The various methods differ depending on the weights assigned to the different coefficients of regression and shall be selected depending on the nature of the disorder. By applying a filter with a window of samples of variable amplitude at 51 '*lowess*' mode we obtain the following result, which already has a good filtration on the first iteration:

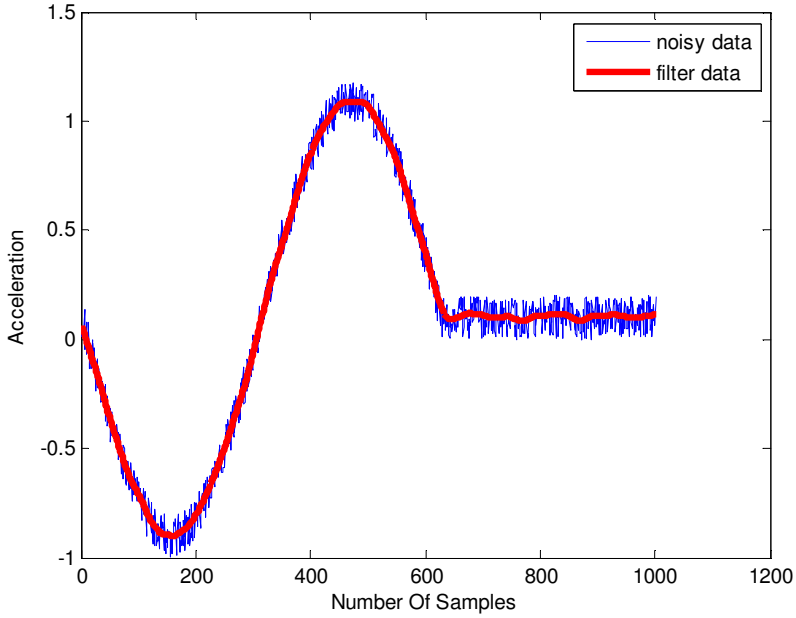



Figure 8. Example of digital filtering applied to a perturbed signal

Once chosen the method of filtering of the data it necessary to proceed to choose an appropriate integration methodology.

In the presence of discrete data such as those provided by a sensor after each sampling interval T, to proceed to their integration you need to apply the concepts and tools described in the mathematical theory of numerical integration. The numerical integration is strongly bound to the type of interpolating polynomial, from which it can not be separated. In general, using numerical integration algorithms which allow us to integrate functions whose are not known the analytical expression but only a few values at certain moments. The formulas of numerical integration (or quadrature) are approximation algorithms of the integral obtained by the integration approximation of the integrand.

The problem then is to approximate as accurately as possible the integral:

$$\Im = \int_a^b f(x)dx$$

where f(x) is a function continuing in [a,b]. The integration formula will be such:

$$I_{n+1} = \sum_{i=0}^{n} w_i f(x_i)$$

where $x_0, x_1, \ldots, x_n$ are named bonds and $w_0, w_1, \ldots, w_n$ weights.

The difference between the integral in analytical form and its numerical approximation is called the rest and represents the analytical error in the approximation of the integral that is committed:

$$r_{n+1} = \Im - I_{n+1}$$

This error depends strictly on the number of knots chosen to evaluate the integral. In relation to this the numerical approximation of the integral is called of level of precision p if for f(x) = $x^i$, $i = 0,1,\ldots,p$ the rest is null, while for f(x) = $x^{p+1}$ it is different from zero.

Considering the polynomial interpolation of f (x) written in the form of Lagrange:

$$L_n(x) = \sum_{i=0}^{n} l_{n,i}(x) f(x_i)$$

and hence considering the integral of Ln(x) instead of f(x), we obtain:

$$I_{n+1} = \sum_{i=0}^{n} w_i f(x_i)$$

Where $w_i$ are the weights for the approximation formula given by:

$$w_i = \int_a^b l_{n,i}(x)\,dx\,, \qquad i = 0,1,\ldots n.$$

These weights can be determined by requiring that the $L_n(x) = \sum_{i=0}^{n} l_{n,i}(x) f(x_i)$ has no level of precision n. In other words, they can be determined by solving the following system of equations:

$$\begin{cases} w_0 + w_1 + \ldots + w_n = b - a \\ w_0 x_0 + w_1 x_1 + \ldots + w_n x_n = \dfrac{b^2 - a^2}{2} \\ \vdots \\ w_0 x_0^n + w_1 x_1^n + \ldots + w_n x_n^n = \dfrac{b^{n+1} - a^{n+1}}{n+1} \end{cases}.$$

The matrix of the previous system, called Vandermonde, is not degenerate, so the system has one and only one solution. If $\overline{w}_0, \overline{w}_1, \ldots, \overline{w}_n$ is the solution of the system, the quadrature formula:

$$I_{n+1} = \sum_{i=0}^{n} \overline{w}_i \, f( x_i )$$

has a degree of precision almost n.

There are several types of numerical integration all turned to minimize the analytical error of the integral calculus, these can be attributed to two main categories:

Newton-Cotes formulas: in which knots are set in the interval [a, b] and are equally spaced. These formulas have no level of precision n or n +1 and their weights are easily obtainable and can be expressed by simple rational numbers. This approach has the handicap of presenting weights of variable sign for n> 9.

Gaussian formulas: knots are not prefixed in advance, but with the weights they are derived to maximize the degree of accuracy that results 2n +1. These formulas, as compared to those of Newton-cotes, have the advantage, in addition to the high degree of accuracy, to have the weights always positive but at the price that the expression of the knots and the weights, is often not rational.

We have chosen to use the Gaussian formulas since they maximize the precision of the integral, and being the interpolating function fixed a priori, it is possible to calculate the off-line weights and knots and putting them in real-time in the approximation formula without further resource waste. In particular the algorithm of integration to which we will refer is

the Gauss-Legendre. The following definitions and theorems are the basis of the algorithm chosen, and allow to understand the mathematical concepts adopted in the development of the integration method of Gauss-Legendre.

Definition: A vector space Γ of the real field R, which is called a scalar product, is called Hilbert space if every Cauchy sequence of elements of Γ is convergent.

Definition: Given a weight function w(x) not negative in the interval (finite or infinite) (a, b) and not identically null and a set of polynomials $\{p_i(x)\}_{i\in N}$ , where $p_i(x)$ is of degree i, it is a Hilbert space respect to the scalar product:

$$\langle p_i, p_j \rangle = \int_a^b w(x)\, p_i(x)\, p_j(x) dx$$

Definition: A polynomial system is called orthogonal respect to a weight function w(x) and to the scalar product defined above if:

$$\langle p_i, p_j \rangle = 0 \quad per\ i \neq j$$

The interval (a,b) and the weight function w(x) univocally define polynomials pi less than non-null constant factors.

Theorem: For every n>0, the orthogonal polynomial $p_n(x)$ has n zeros real or distinct and all contained in (a,b). Furthermore, the zeros of $p_n(x)$ alternate with those of $p_{n-1}(x)$ ie between two consecutive zeros $p_n(x)$ only one zero exists of $p_{n-1}(x)$ .

Theorem: A quadrature formula of interpolatory type built on n+1 points has degree of precision at least n and at most 2n+1. The maximum level

is reached if, and only if, the knots are the zeros of (n+1)-th orthogonal polynomial to the weight function w(x) = 1.

The orthogonal polynomials that are used in the case of Gauss-Legendre formulas are precisely the Legendre polynomials. They are defined as solutions of the differential equation of Legendre:

$$\frac{d}{dx}\left[(1-x^2)\frac{d}{dx}P(x)\right] + n(n+1)P(x) = 0$$

The Legendre differential equation can be solved by standard methods of power series. We have solutions given by convergent series for |x| <1.

We have also converged solutions for x = ± 1 provided that n is a natural whole (n = 0, 1, 2, ...) in this case the solution of changing of n form a polynomial sequence called "Succession of Legendre polynomials". They can therefore be defined for recurrence according to the formula:

$$\begin{cases} P_0(x) = 0 \\ P_1(x) = x \\ P_{n+1}(x) = \frac{2n+1}{n+1}xP_n(x) - \frac{n}{n+1}P_{n-1}(x) \end{cases}$$

For n ≤ 5 we can see them in the following figure:

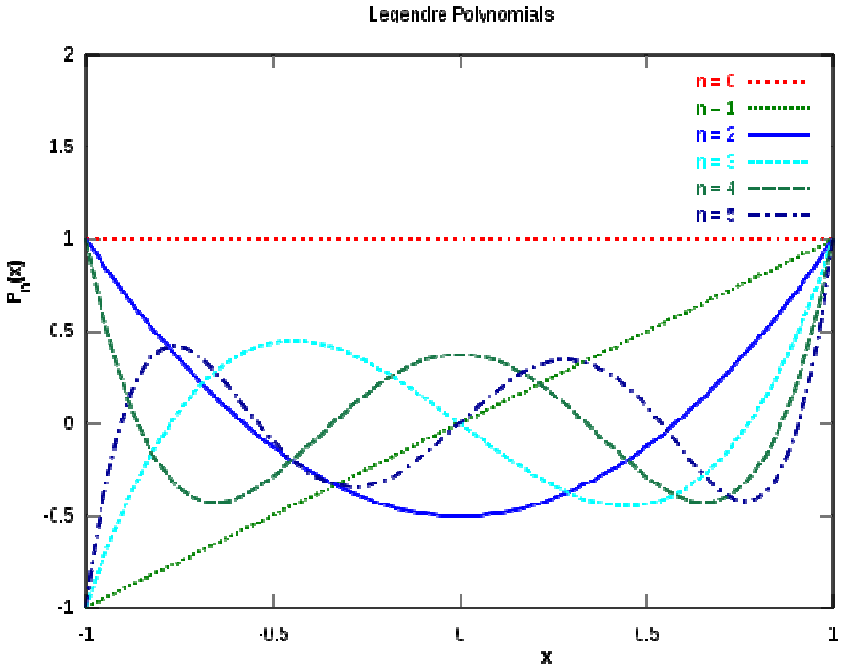



Figure 9. Performance of Legendre polynomials up to the fifth degree

These polynomials are also useful for very precise calculation of a polynomial of discrete values of the accelerations.

After defining the Legendre polynomials the Gauss quadrature formula can be easily obtained, assuming we want to integrate the function with n+1 knots, performing the following steps:

Calculating the Legendre polynomial p of degree n+1, to compute $x_n$ nodes.

Calculation of Legendre polynomial q of degree n+2, to calculate the weights using the formula:

$$w_i = \frac{2(1 - x_i)^2}{[(n+2)q(x_i)]^2}$$

The so calculated weights and nodes will be defined in the interval [-1,1], then I carry them to the interval [a b], with the transformation:

$$\begin{cases} x_n' = \frac{b-a}{2} x_n + \frac{b+a}{2} \\ w_i' = \frac{b-a}{2} w_i \end{cases}$$

Finally, we apply:

$$I_{n+1} = \sum_{i=0}^{n} w_i f(x_i)$$

Where, we remind, f(x) is the interpolating function to be determined.
A crucial step for a successful integration of any discrete data is a precise and consistent interpolation. There are again various types of interpolation, the use of which is discriminated by the type of precision that we want to get.
The problem, quite generally, can be defined as follows:
Given a sequence of n distinct set of real numbers xk called knots, and for each of these xk gave a second number yk. We aim to find a function f such that the relation is satisfied:

$$f(x_k) = y_k \text{ per } k = 1 \dots n$$

Where n represents the number of samples available to us.
We therefore speak of: known some data pairs (x, y) interpretable as points of a plan, we aim to build a function, called interpolating function, which is able to describe the relationship between the set of values x and the y values ensemble.

A first and obvious example of interpolation is the linear one, in which for every pair of consecutive points called (x, y) and (xb, yb) is defined interpolating function in the range xa, xb the following function:

$$f(x) = \frac{x - x_b}{x_a - x_b} y_a - \frac{x - x_a}{x_a - x_b} y_b$$

This interpolation produces a response like:

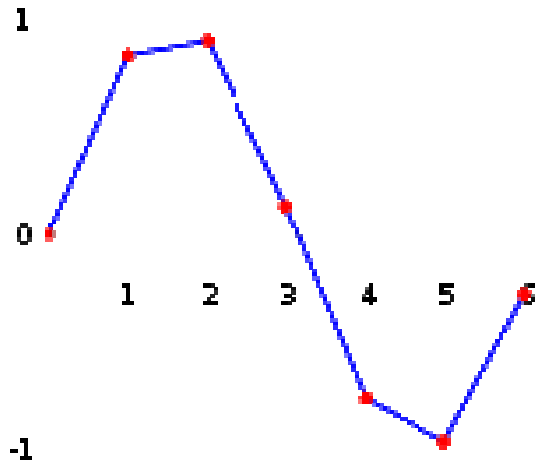



Figure 10. Example of linear interpolation

This type of interpolation is very fast and basic, but it has several handicaps, such as the discontinuity in the transition points and an error increasing with the square of the distance of the points to be interpolated. More precise and complex respect to the linear interpolation is the polynominal, which can be regarded as a generalization of the previous one, and that is to approach the interpolating function with a polynomial of degree n by using the method of least squares. Since, for example, a

set of discrete data that you want to interpolate with a polynomial of degree n such as:

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

And given a set of m samples, with $m \geq n$, we will interpolate the data as:

$$\begin{bmatrix} 1 & x_0 & \cdots & x_0^n \\ 1 & \vdots & \ddots & \vdots \\ 1 & x_m & \cdots & x_m^n \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ \vdots \\ y_m \end{bmatrix}$$

By naming with B the matrix constituted by samples $x_m$, with $\vec{a}$ the coefficient vector and Y the sample vector corresponding to each $x_m$, we will get that the coefficient vector is given by:

$$\vec{a} = (B^T B)^{-1} B^T Y$$

To make the most accurate polynomial approximation can be thought of dividing the interval into subintervals of interpolation to be interpolated separately, this will make more accurate the result but it will introduce points of discontinuity between a set of samples and the other. By applying this method to a set of points where a white noise is superimposed, we obtain for example:

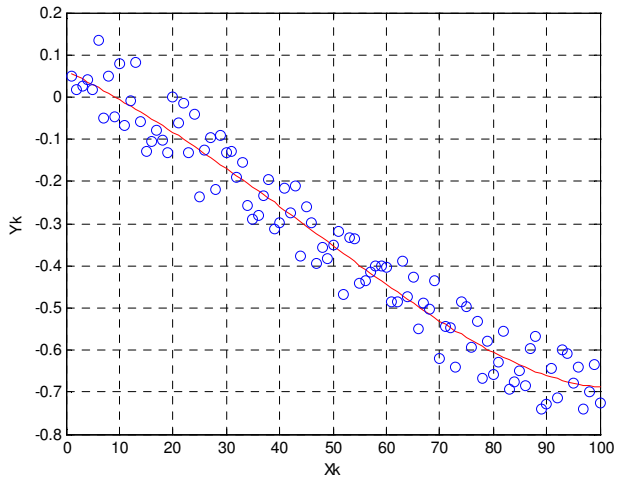



Figure 11. Example of polynomial interpolation

Using a fourth-degree polynomial interpolation. A big problem with this approximation is that it suffers from Runge's phenomenon. The Runge's phenomenon is a problem related to the polynomial interpolation of high degree polymonials, it is the progressive increasing of the error near the ends of the range as shown in the below figure:

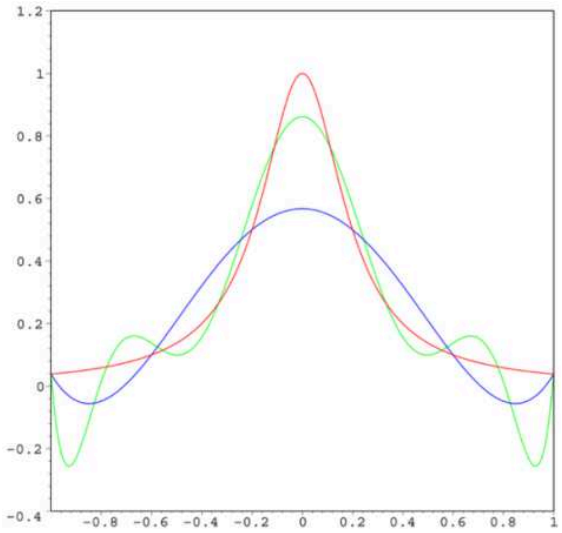



Figure 12. Performance of the error due to Runge's phenomenon

The red curve is the Runge function, the blue curve is a polynomial of the fifth degree, and the green curve is a polynomial of ninth degree. Ordinate we have the error, as you can see the approximation deteriorates with increasing degree. In particular, interpolation by a polynomial Pn(x) of degree n, it gets worse according to the relation:

$$\lim_{n\to+\infty}\left(\max_{[-1,1]}|f(x)-P_n(x)|\right)^n=+\infty$$

To overcome this problem and does not affect further the integration of data with additional errors caused by interpolation it has been chosen to use another technique called cubic spline interpolation. This methodology is a particularization of the method of spline function interpolation.

A spline function is defined in the following way:

$\Lambda \equiv \{a = x_0 < x_1 < \cdots < x_n = b\}$ given as a closed interval subdivision [a,b]. A spline function of degree p with knots at the points xi with i=1,2, ... n is a function on [a,b] denoted by sp(x) such that, in the interval [a,b] we have:

in each subinterval $[x_i,x_{i+1}]$ where i=1,2,…n the function sp(x) is a polynomial p degree the function sp(x) and its first derived p-1 are continuous in each sub-interval.

Particularly, the natural cubic spline function is nothing more than a spline function of third degree. The idea behind this interpolation is to divide the samples interval into smaller intervals interpolated by an appropriate spline.

Use a third-degree polynomial with continuous derivative in each sub-interval can correctly interpolate any signal getting significantly better results compared to a polynomial interpolation.

The application of this type of interpolation to the previous case we obtain:

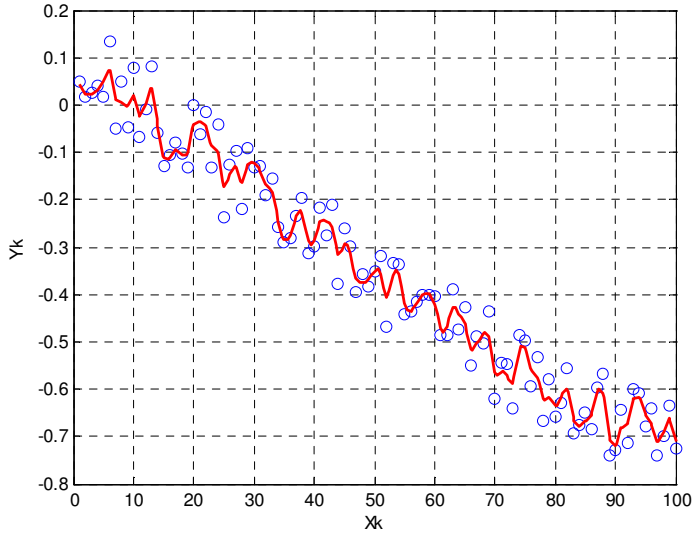



Figure 13. Example of cubic spline interpolation

You can see how, even in presence of a highly variable signal, this solution produces much more satisfactory results than those of the linear interpolation.

The interpolation polynomials of Legendre polynomial is similar to that based on the method of least squares, but due to the nature of the interpolating polynomials it can approximate to a finely discrete signal even in presence of a strong variation of that.

Unfortunately a large computational burden is required due to the high number of Legendre polynomials to use; but it does not suffers from

Runge's phenomenon and can be used in off-line data processing. The interpolation formula in this case is the following:

$$p(x) = a_0 + a_1 P_1(x) + a_2 P(x)_2 + \cdots + a_n P_n(x)$$

Where the various P(x) in this case are the Legendre polynomials previously defined and which can be tabulated before the interpolation itself. In relation to the previous case, this interpolation provides a slightly better (but takes longer) result than the natural cubic spline:

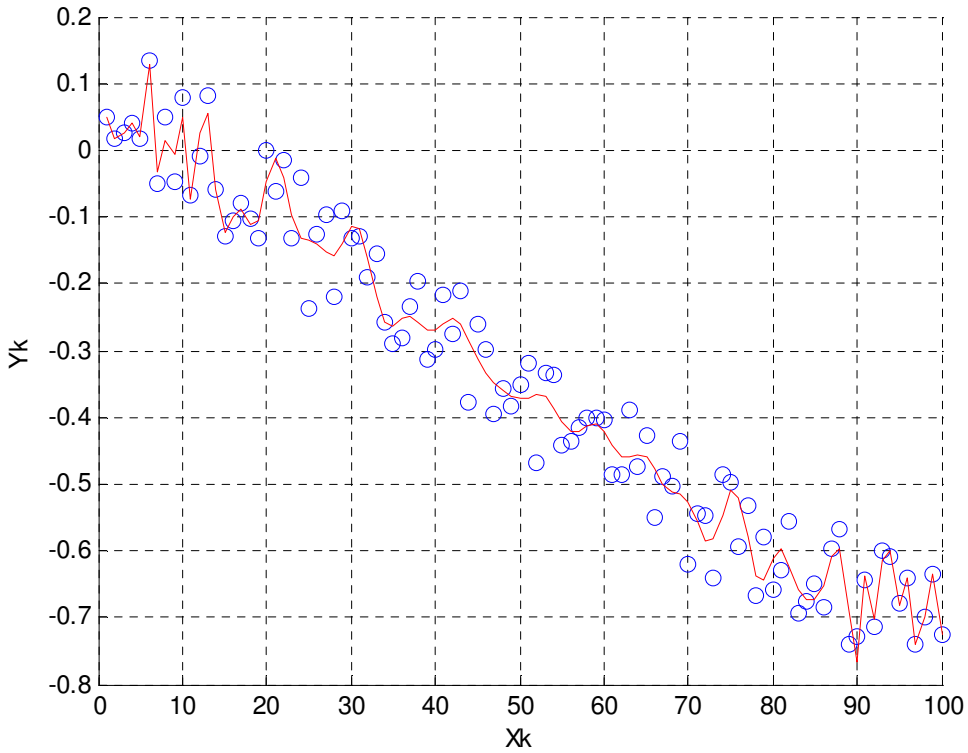



Figure 14. Example of interpolation with polynomials of Legendre

## 1.4 MAIN HARDWARE: ST-iNEMO IMU BOARD

The sensor platform used primarily for the development and the work presented is a prototype provided by ST (STEVAL-MKI062V1) called the iNEMO™; that is, iNErtial MOviment unit. Actually to this, time to time, it has been developed alongside other boards such as CRM003 GPS board, equipped with Theseus chip, connected to the iNEMO gate UART2 through a level translator ST3232, or the rover I.A.F.F.I.E.D.D.U. developed on control cards based on microprocessor STM32 which is of ARM model.

The iNEMO inertial platform, developed by STMicroelectronics, is a 10-axis Inertial Measurement Unit (dof, degrees of freedom), made with:

- 6-axis geomagnetic sensor, LSM303DLH: 3-axis accelerometer and 3 axis magnetometer. Digital output with protocol $I^2C$
- 2-axis gyroscope (roll, pitch), LPR430AL, analog output
- Axis yaw gyroscope: LY330ALH, analog output
- Pressure sensor LPS001DL, a digital output with protocol $I^2C$
- STLM75 temperature sensor, a digital output with protocol $I^2C$
- MCU Microcontroller Unit STM32F103RE

The board shows various options for interfacing: MicroSD card connector (data logging), USB connector, serial connector and a connector for an optional wireless interface.

It is an inertial system to 10 degrees of freedom which can be used in many applications such as, for example, virtual reality, augmented reality, image stabilization in a camera, man-machine interaction and robotics.

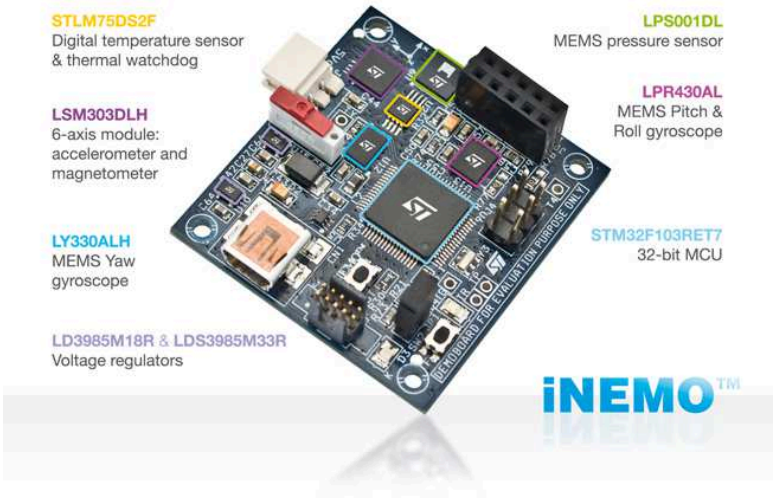



Figure 15. STEVAL-MKI062V2, iNEMO, realized by STMicroelectronics, Catania

The firmware supplied with the board has an integrated sensor fusion algorithm AHRS (Attitude and Heading Reference System) that provides an output roll, pitch and yaw (roll, pitch and yaw, in the notation of Euler angles, explained graphically in the below figure) and the parameters $q_0$, $q_1$, $q_2$, $q_3$ on the notation with quaternions


.

Figure 16. Graphical explanation of the angles of roll (rotation around the X axis), pitch (rotation about the Y axis) and yaw (rotation around the Z axis).

Gyroscopes

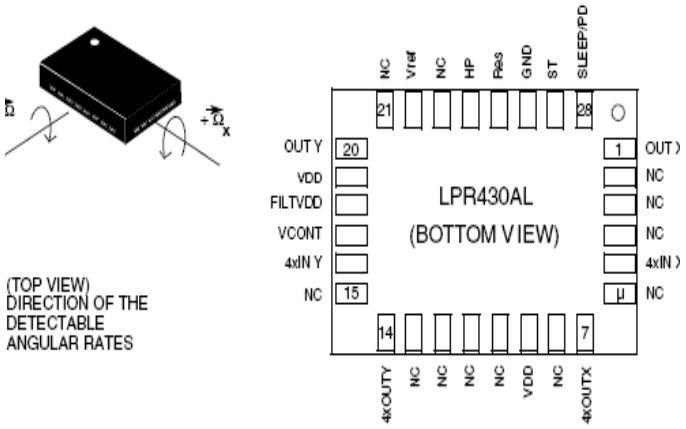



Figure 17. Roll and pitch gyroscope: LPR430AL: 2-axis gyro (roll, pitch) 300°/s full-scale

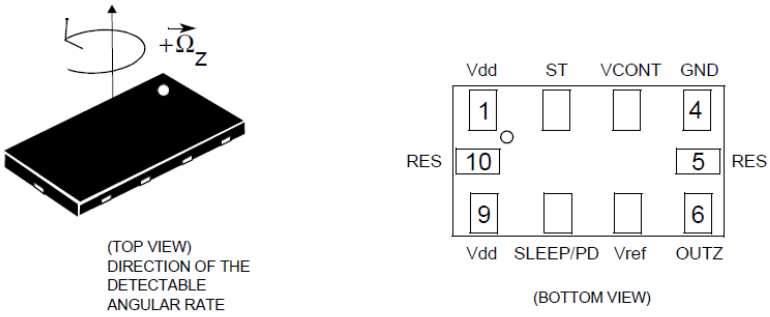



Figure 18. Giroscope yaw: LY330ALH: yaw-axis gyro 300°/s full scale

iNEMO uses two devices to detect the angular velocities around the three axes: LPR430AL in the same package contains two gyroscopes that detect the speed of rotation around the axis of rolling and pitching; LY330ALH detects the speed of rotation around the yaw axis. The output of the gyroscopes is analog, connected to an ADC, present in the same microcontroller.

Geomagnetic Unit: accelerometer and magnetometer

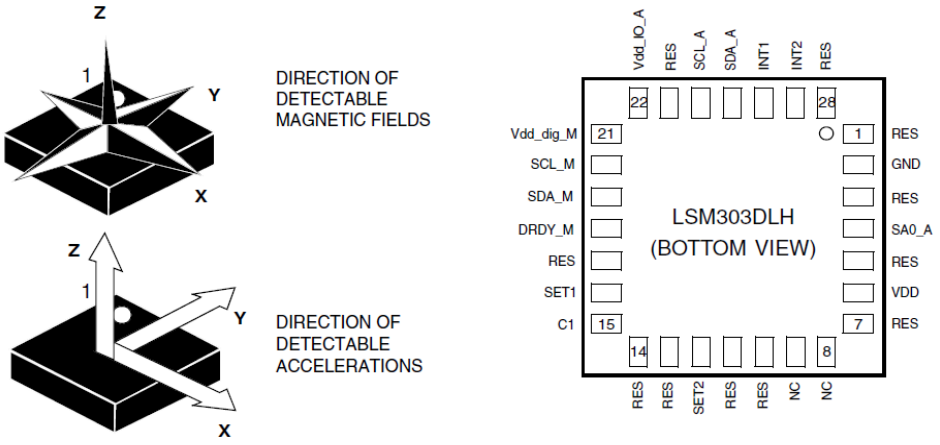



Figure 19. Geomagnetic module LSM303DLH: 6-axis geomagnetic module

The geomagnetic module LSM303DLH in the same package integrates a three-axis accelerometer and a three-axis magnetometer. It is interfaced with the microcontroller via I2C interface.

Barometer

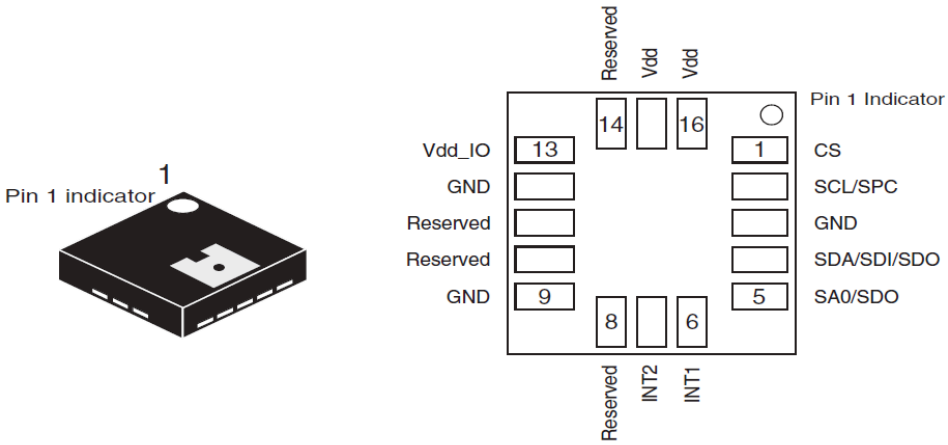



Figure 20. LPS001DL: pressure sensor 300-1100 mbar absolute full scale

The pressure sensor measures the air pressure LPS001DL in the range 300-1100 mbar. It is interfaced with the microcontroller via I2C

interface, it is a high-resolution sensor. The sensing element is based on a membrane with a piezoresistive approach and proper conditioning electronics based on a Wheatstone bridge and amplifiers, it converts pressure into an electrical signal. The sensor also features two programmable interrupts to recalibrate and adapt it to different pressure events, not only, we have also a DRS interrupt (data ready signal) which is also managed by micro, which provides better data synchronization. The following circuit shows the micro outputs to manage INT1 and INT2.

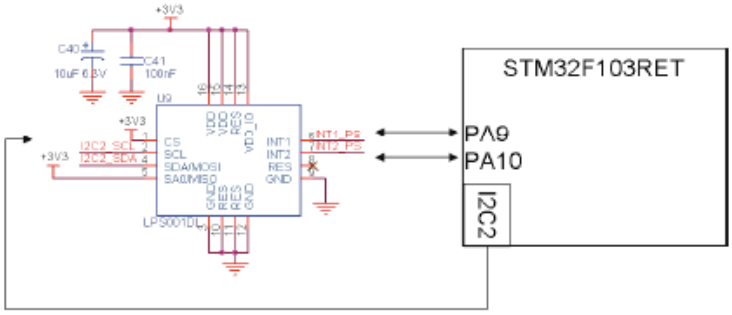



Figure 21. Pressure sensor schematic

Thermometer

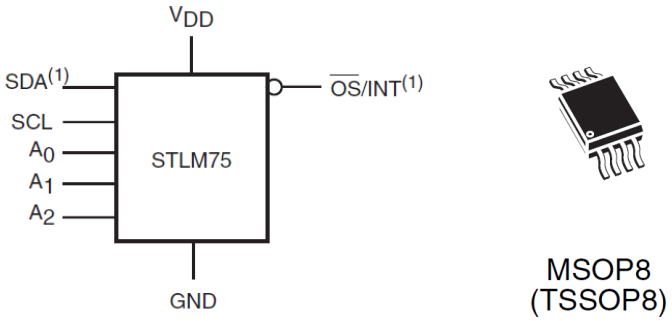



Figure 22. STLM75: temperature sensor with –55 to +125°C range

The temperature sensor STLM75 detects the environmental temperature in the range from -55°C to +125°C. It is interfaced with the microcontroller via I2C interface.

Along with the Hardware, the DLL libraries have been developed and made available by ST where the necessary functions are implemented to interface of the board with Matlab or other development environments.

DLLs also provide communication with GUI iNEMO, an intuitive graphical interface shown in the following figure; it is useful to be able to become familiar with the signals from the sensors and display them on screen in a simple and immediate way. This aspect is not of secondary importance in understanding the nature of the signals work, the noise to which they are affected is an important first step to begin to understand which could be the first steps of developing an algorithm, for example: the filtering data, the correction of off-set, the correction of over-elongations and settling times typical of MEMS structures.

Figure 23. Graphical Interface iNEMO.

The interface allows to display information from sensors either continuously or for a finite number of samples, but also to save them as files "**.tsv*", enabling you to load data into calculation software such as Matlab and set so the development of algorithms off-Line.

The connection between the board GPS and iNEMO has been achieved through the iNEMO UART (UART2 of the STM32 microcontroller) and the serial port labeled NMEA of the GPS. However, the GPS transmits the standard RS232, while the serial iNEMO is a TTL/CMOS. The RS232 features a single-bit transmission voltage levels ranging from ± 5V to ± 15V. Normally we use a voltage of ± 12V.

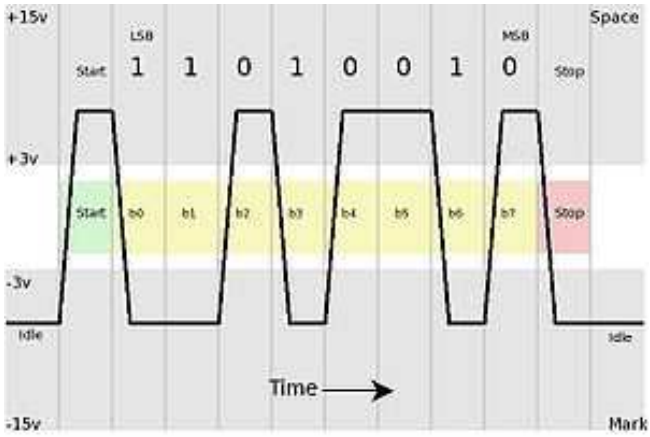



Figure 24. Port signal RS232

The transmission TTL/CMOS - typical of digital integrated circuits - occurs in voltage levels in the 0V-5V range: typically about 1.3V for low logic level, and 4.3V for high logic level. It must therefore include a transceiver adapting TTL signal to RS232 levels and vice versa.

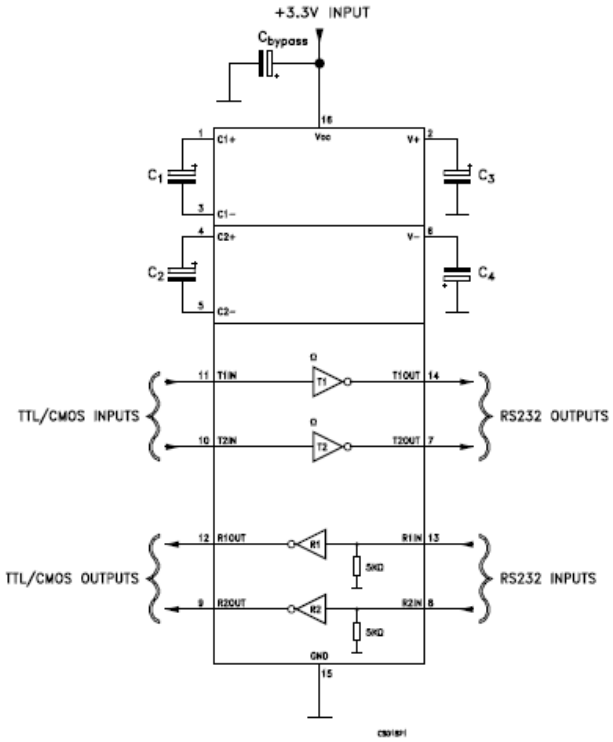



Figure 25. GPS interface circuit - iNEMO

The GPS module transmits every second a train of sentences containing data on location, speed, etc.. Therefore, it has been integrated in the firmware of iNEMO algorithm that deals with:

- Stocking data arrival from UART2
- Extract interest sentences RMC and GGA
- Extract from these last the wished fields

The UART2 of iNEMO is buffered cyclically on a string, the microcontroller DMA (Direct Memory Access) is in charge of copying the bytes (characters) coming directly in this string, the function DMA_GetCurrDataCounter (DMA1_Channel6) returns a pointer to the character of the string inserted last (this means that the previous

characters are the last to be received). After entering the last character, the pointer returns to the beginning of the string. Once loaded the strings RMC and GGA from the cyclic buffer, they are queued to be processed RMC and GGA, respectively. The information to be used are: UTC time - GPS, Present Position Latitude, Longitude of the current position, the quality of the GPS survey, the number of satellites in view, Altitude of the GPS antenna relative to the average sea level. Finally, the module iNEMO connects to the PC via the USB interface. The driver used is Virtual COM, for the management of the data it was used ControlLibrary namespace, supplied by STMicroelectronics in the *SDK* (Software Development Kit).

# CHAPTER II - ORIENTATION ESTIMATION

## 2.1 INTRODUCTION

Having the best estimation possible of board's orientation is the most important thing to do, making a working application starts now. Even considering that MEMS components are very high-quality in terms of noise level, there are still some that need to be filtered. Accelerometer has been filtered with a low pass FIR discrete filter to avoid noise influence and also to get a more clear gravity vector estimation when the board is shacked. Magnetometer has not been filtered, but before its data could be usable, it needs to be calibrated; calibration is an operation which depends on the working environment, because there are always influences producing even little distortion of the Earth's magnetic field. Gyroscope is the only component for which there is no needing of filtering nor calibration, and its output has been integrated to calculate the current board's orientation.

The orientation recognition is based on the estimation of the gravity vector and on a rotation about its axis; in this way we have completely defined the board orientation. As already said, the gravity vector is given by the accelerometer and we worked under the assumption that it's always correct (even though there could be the possibility of some erroneous samples); but just from the accelerometer we cannot have an estimation of the rotation around the G-axis; that comes from magnetometer and gyroscope which jointly compensate the accelerometer's defect. Theoretically, it is possible to get a complete orientation estimation just from accelerometer and magnetometer, as

many inertial platforms in commerce do, because they compensate each other defects: rotation around the G-axis (accelerometer) and rotation around the North-South axis (magnetometer); but as it is known magnetometer is affected by external influences (such as steel or magnetic objects), so by having the gyroscope contribute, there is always another point of reference to avoid to lose orientation around the G-axis in case that the working environment has a high interference's level.
Next paragraphs will describe the steps needed for the orientation representation: starting with the accelerometer filtering and the magnetometer calibration, and then describing the gyroscope data integration's correction by the accelerometer and magnetometer "clean" data. The purpose of this steps is to calculate, from the iNEMO board output, a rotation quaternion in respect to a given start position.

## 2.2 ACCELEROMETER FILTERING

By filtering the accelerometer we can solve two problems: reducing the noise coming from the component, and cleaning the output from any wrong data that can come by moving the iNEMO board too fast. The goal is to extract the three components of the g-vector on the relative reference system in agreement with the board, the gravitational vector should have always norm 1. Since we are talking about discrete signals, we need a discrete filter to manipulate them; to perform it a FIR (Finite Impulse Response) low-pass filter has been used. Using a FIR filter means manipulating a signal to obtain a cleaner output without the noise that we want to remove. The FIR filter formula is:

$$y[n] = \sum_{i=0}^{N} (b_i \cdot x[n-i])$$

Where: $y[n]$ is the filter output at the sample $n$; $b_i$ are the FIR coefficients; and $x[n]$ is the input signal at the sample $n$.

The FIR coefficients come from the Fourier transform of a linear low-pass filter's transfer function; by trying several combination of cut-off frequency and filter's order. The optimal specification that has been found for the accelerometer is a cut-off frequency of 2 Hz, with a filter's order of 20. A FIR filter is just a low-pass filter suitable for discrete signals, which have to be filtered from a finite number of their sample; the accuracy of a FIR filter respect to its analog equivalent depends on the filter's order; in fact higher is the filter's order, more accurate it will be compared to the effect given by the analog one. But if the filter's order is too high, we will introduce a delay on the filtered signal which can compromise the orientation representation accuracy. For this reason a fair number of tests has been done to find the best compromise possible.

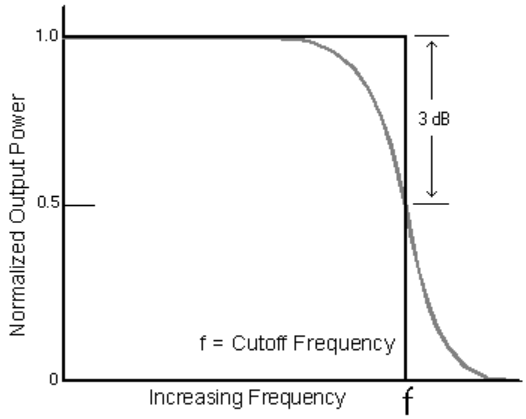



Figure 26. Example of a low pass filter with its finite order equivalent.

For calculating Fourier transform, the FFT (Fast Fourier Transform) algorithm has been implemented due to its rapidity quality, it has, in fact, a computational complexity of $O(n \log n)$ instead of a computational complexity of $O(n^2)$ given by the normal Fourier transform. .

Now the filtered accelerometer data is obtained, this means that it has been cleaned from the white noise given by the sensor and from external contributes that cause loss of precision on the G estimation when the board is shacked.

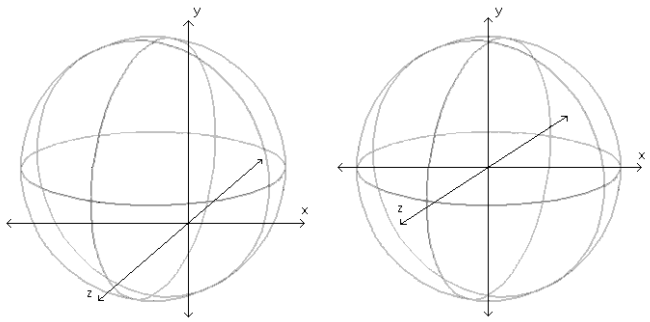
## 2.3 COMPASS CALIBRATION

As it is known, magnetometers are affected by external influences; steel materials or magnetic source are always present in laboratory's environment, and that is the first cause of unreliability for a magnetometer. However there could be the possibility where this magnetic disturbs can only deflect the Earth magnetic field without compromising the reliability of the magnetometer; in these cases a good calibration is necessary to get a good estimation of the Earth magnetic field.

When the board is not moving, the magnetometer works similarly to the accelerometer, but instead of pointing to the Earth center, it points to the direction given by the magnetic field detected; by rotating the board around each possible direction and around each axis, we can see the compass "needle" (meant as the normalized vector built from the three component of the magnetometer) moving within a sphere object. Basically the needle is always pointing a fixed point in the relative reference's system of the board; but since the magnetometer is not calibrated this sphere is not centered on the axis origin of the relative

system of the board. The purpose of the compass calibration algorithm is to align this sphere to the origin; after an initial phase of data acquisition in which all the three axis range of the magnetometer are recorded. These ranges are used to calculate the real center of the not-calibrated sphere and translate this sphere to the axis origin by adding to each sample the coordinates of the calculated centre. In this way we can obtain each time a calibrated sample.

This picture explains the solution given by the compass calibration to the magnetometer's error:




(a) (b)

Figure 27. Difference between magnetometer range of values before (a) and after calibration (b).

The calibration phase, as already said, derives first from the real working phase, but however it needs the data acquisition to be started in order to get all the samples needed for the calibration; to do it. The data acquisition has to be started and, then, the board has to be rotated around each axis in all the possible position, so as to have all the values range from each magnetometer axis.

Once the calibration stage is completed too, data from accelerometer and magnetometer are ready to be used as they are for the orientation representation.

## 2.4 METHOD OF ORIENTATION ESTIMATION

Orientation can be determined by using the direction of the magnetic field of the Earth and the gravity vector's direction. Magnetometers are employed to detect the magnetic field of the Earth. Accelerometers are used to detect the gravity direction. In order to better understand how to combine all information coming from the sensors, we have to know which kind of information we got and in which unit of measure they are expressed.

All information coming from the three main components of the iNEMO board are:

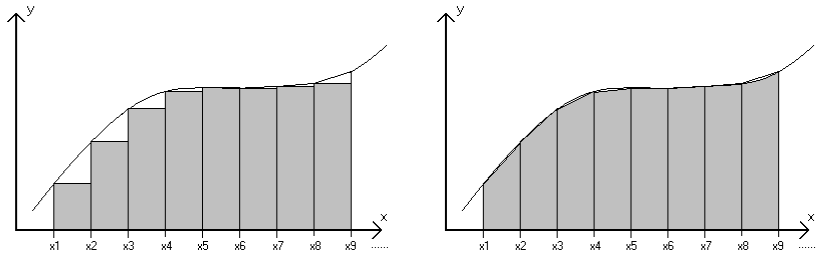
- Gyroscope: angular speed components (in degree/sec) around the three axis of the board related to the reference system;
- Accelerometer: acceleration components (in mg) on the three axis of the board related to the reference system;
- Magnetometer: Earth's magnetic field components (in mG) on the three axis of the board related to the reference system;

All this information must be considered clean and ready to use, since they have already passed the process of calibration (magnetometer) and filtering (accelerometer). Note that the accelerometer measure's unit mg means milli-g, where g is gravity acceleration (9,81 m/s$^2$).

The method of orientation representation consists in combining all these information to get a global orientation in respect to a given start position

expressed as a rotation quaternion. In order to obtain this goal, we first need to make all this information useful; and make them useful means transforming them in rotation quaternion, so as to have a relative "truth" of each component and combing them to have the final rotation estimation.

Having a rotation quaternion from the gyroscope is possible by integrating the angular speed around of each axis, so as to have a relative rotation for each sample; the integration method used for this purpose has been the trapezoidal rule, which is a useful method to reduce error during discrete signal integration. The trapezoidal rule is explained by this picture:




(a) (b)

Figure 28. Error's difference between classic integration (a) and trapezoidal integration (b).

The formula for the trapezoidal integration is:

$$\vec{V} = \frac{\overrightarrow{G_p} + \frac{(\overrightarrow{G_c} - \overrightarrow{G_p})}{2}}{f}$$

Where: $\vec{V}$ is the current angular rotation vector; $\overrightarrow{G_p}$ is the gyroscope data vector for the previous sample; $\overrightarrow{G_c}$ is the gyroscope data vector for the current sample; and $f$ is the sampling frequency.

The first cause of error is just the integration, this because for each sample we integrated with error, we are going to add this error to the next sample's integration, and so on for each sample's integration. In this way we have a drift growing in time that needs to be corrected with a non-divergent information.

Since both accelerometer and magnetometer data are not integrated, we can consider them as non-divergent information; in fact even if there would be some incorrect samples, they'll give their contribute just for a very short time without adding a drift to the orientation estimation.

The algorithm is explained by the following flow chart:

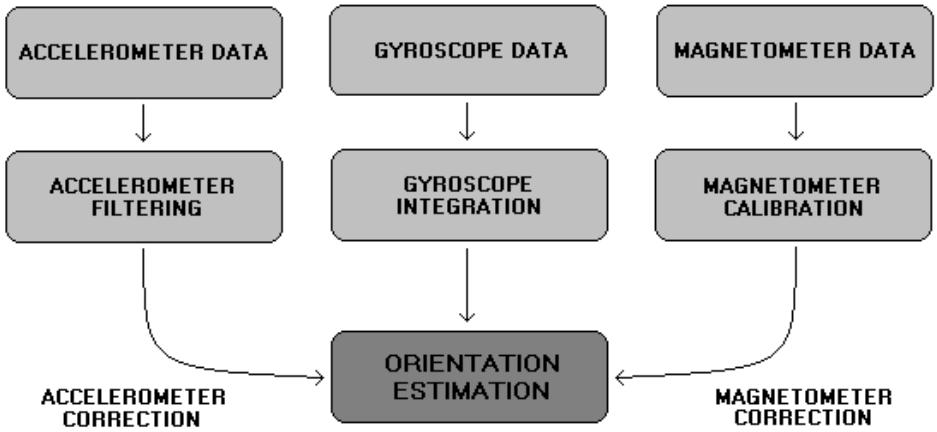



Figure 29. Schema of the orientation estimation's algorithm.

As the flow chart explains, the way to combine all the information coming from all components is to correct them by comparing their each own "truth"; each subjective truth is expressed by a rotation quaternion, which represents a rotation from a global start position to the current position.

Since the way to obtain the quaternion from the gyroscope has been already explained, we need to define how to get a quaternion from the accelerometer and the magnetometer. All this is based on a start position which is different for accelerometer and magnetometer:

- Regarding the accelerometer: extracting a quaternion from its data has been implemented by calculating the rotation from a start position, assumed as the unit vector $v = (0,-1,0)$ , to the current position, assumed as the current normalized data of the accelerometer;
- Regarding the magnetometer, the start position has been assumed as the first sample during the data acquisition after the calibration; this because is not possible to define a global start position which is not dependent on the environment for the magnetometer.

The approach is basically to get the hypothetic current position given by the gyroscope, obtained by rotating the accelerometer start position and the magnetometer start position with the quaternion coming from the gyroscope integration, and comparing it with the effective samples given by the sensors. The formula to rotate a vector w= $(w_x, w_y, w_z)$ by a quaternion $q = (q_x, q_y, q_z, q_w)$ to obtain $w^r = (w_x^r, w_y^r, w_z^r)$ is the following:

$$\begin{bmatrix} 1-2\cdot(q_y^2+q_z^2) & 2\cdot(q_xq_y-q_zq_w) & 2\cdot(q_xq_z+q_yq_w) \\ 2\cdot(q_xq_y+q_zq_w) & 1-2\cdot(q_x^2+q_z^2) & 2\cdot(q_yq_z-q_xq_w) \\ 2\cdot(q_xq_z-q_yq_w) & 2\cdot(q_yq_z+q_xq_w) & 1-2\cdot(q_x^2+q_y^2) \end{bmatrix} \cdot \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} w_x^r \\ w_y^r \\ w_z^r \end{bmatrix}$$

Once we have the hypothetic current positions (from the accelerometer and magnetometer start position) given by the gyroscope quaternion, they can be compared with the real ones (the effective sensors value); since, as already said, the gyroscope has a drift due to its integration; the

hypothetic and the real ones won't be the same, it depends on the drift. As long as they are similar there will be no need of correction; but when they diverge too much then the gyroscope "truth" must be corrected. The correction is performed in the same way for both accelerometer and magnetometer, the principle is to calculate the rotation needed to rotate the hypothetic current position given by the gyroscope to the real current position given by the accelerometer or the magnetometer.

Calculating a rotation between two vectors means calculating a quaternion, and since a rotation quaternion is defined as:

$$q = (\hat{v}_x \sin\frac{\alpha}{2}, \hat{v}_y \sin\frac{\alpha}{2}, \hat{v}_z \sin\frac{\alpha}{2}, \cos\frac{\alpha}{2})$$

Where $\hat{v}$ is the rotation axis and $\alpha$ is the rotation angle; it is possible to define a quaternion between two vectors from their cross and dot product, since for definition the cross product between two unit vectors is the rotation axis that needs to obtain the second by rotating the first, and the angle $\alpha$ is given by the dot product which represents the $\cos\alpha$.

Note that the product between quaternions is intended as the Hamilton product; trying to be clearer, considering $q_1 = (q_{1_x}, q_{1_y}, q_{1_z}, q_{1_w})$ and $q_2 = (q_{2_x}, q_{2_y}, q_{2_z}, q_{2_w})$ their product $q_3 = q_1 \times q_2$ is:

$$\begin{aligned} q_3 = (&q_{1_w} q_{2_x} + q_{2_w} q_{1_x} + q_{1_y} q_{2_z} - q_{2_y} q_{1_z}, \\ &q_{1_w} q_{2_y} - q_{2_z} q_{1_x} + q_{1_y} q_{2_w} + q_{2_x} q_{1_z}, \\ &q_{1_w} q_{2_z} + q_{2_y} q_{1_x} - q_{1_y} q_{2_x} + q_{2_w} q_{1_z}, \\ &q_{1_w} q_{2_w} - q_{1_x} q_{2_x} - q_{1_y} q_{2_y} - q_{1_z} q_{2_z}) \end{aligned}$$

Now, for each sample coming from the board we are able to calculate a quaternion which represents the rotation of the board from the start position, assumed as the position where the accelerometer measures the gravity as for the vector (0,-1,0) in the related reference system of the

board. Once we have this information as a quaternion, it will be pretty simple to show it graphically by a rotating cube.

## 2.5 EXPERIMENTAL RESULTS

Inertial platforms with MEMS sensors are very good in terms of accuracy and noise level; but even considering that, there are some adjustments that have to be done to make them work properly. For each adjustment there are always several possible configurations and part of time of this work has been spent on tests regarding this configuration; except for the compass calibration which has not got any important setting to configure, accelerometer filtering and gyroscope integration give some alternatives to choose with relevant changes in terms of experimental result.

There are also several possibilities in the iNEMO-Robot communication, considering in fact that the kinematic inversion algorithm can calculate the six joints angle from a given point; it is not necessary to give them directly to the robot, but they can only be useful to evaluate if a point is reachable or not.

Recent studies assert that inertial platforms have proven to be very helpful in such many areas, and actually they are; but, however there are at least three problems that must be kept in mind, every time an application with them has to be developed:

- Accelerometer noise;
- Gyroscope integration drift;
- Magnetometer external influence.

In this thesis a solution to each one of these problems has been found, and with very good results: a low-pass FIR filter with cutoff frequency of 2 Hz and filter's order 20 to clean the accelerometer output, allows to have the best estimation possible of the gravitational force; the gyroscope integration's drift has been corrected and reset with the information of accelerometer and magnetometer; magnetometer has been calibrated. However there are still some problems that can happen: for example, as already said magnetometer is always influenced even by little distortions of the Earth magnetic field; when this influence is very strong (such as a metallic material or a magnet at very short distance from the board), this disturb is easily recognizable due to its influence on the magnetometer output, its values in fact can even assume values up to 200% of the normal magnetometer's range of values without disturbs, so they can be recognized and discarded; but there could be situations in which the external influence is very weak. As this causes only little distortion on the magnetometer output not clearly recognizable, such problem is very common in many environment as research laboratories where a fair number of working instruments can typically distort the Earth magnetic field detected by the magnetometer; to avoid incoherent information in these situations the magnetometer has to be calibrated for each side of the room it has to work in, and once it is calibrated it should not be moved too far from the calibrating position. This because all the magnetic field influence are always unpredictable and hard to study. For example the robotic application for this thesis has been tried in a laboratory with a high level of "unrecognizable" magnetic noise, and with the method just explained, the orientation estimation has always been correct.

## 2.6 ROBOT-INEMO INTERACTION RESULTS

The interaction between iNEMO and the Robot is a key point for the working application; in fact, as already said, there are two ways to give to the robot information coming from the board orientation. When we want the robot to execute a movement command, we have to specify the point, the tool object offset and the orientation of the robot around this point.

Difference in robot axis configuration for the same Target Point with different Tool Object Offset and Tool Center Point Orientation.

The application has been executed under simulation with the software, to test the kinematic inversion algorithm and to be sure that everything worked correctly.

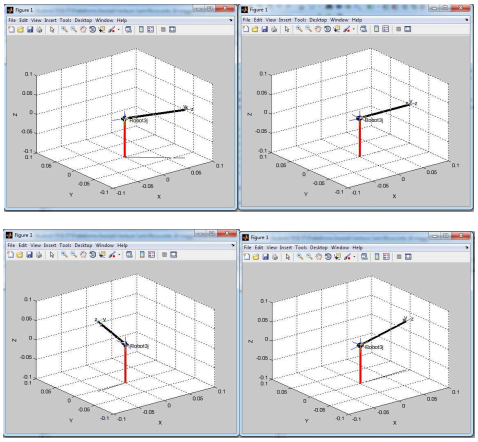


Figure 30. MatLab Simulation of robot movement.

After that the application has been tried on a robotic arm, the robot speed has been always reduced to avoid collision with the environment but even with that, the robot response was very good. The communication has been done through Ethernet with a frequency of data of 50 Hz and robot has never demonstrated signs of overloading.


Figure 31. Block diagram of system.

Next figure show the working application executed in laboratory (with the collaboration of the Automation Center in Västerås Sweden)

Figure 32. Test of robot driving.

Inertial sensors for human motion recognition are now widely diffused to make similar applications and further developments are in course. This results demonstrate that it is possible to communicate commands to robots through inertial sensors making an application with very good results, and it opens the path to new ideas and further applications. This means that there will be a drift on the position estimation that will grow exponentially in time. The solution is to find a fixed point of reference which could be useful to cancel this drift. A possible point of reference can be found by adding to the iNEMO board a vision system to get a fixed reference while the board is moved; in this way it is possible to correct errors from the double integration of accelerations and in particular, the implementation of a mathematical model able to reconstruct a given trajectory through the use of an inertial system.

## 2.7 STATISTICAL ANALYSIS

Through an initial analysis on a statistical study of the signals from the sensors, including within the inertial used platform, it has been possible to obtain the desired information regarding speed, position and orientation of the board with a certain degree of accuracy and precision.

Statistical analysis was performed by mounting the platform on the inertial end-effector of a manipulator and making a considerable number of acquisitions. The trajectory for each acquisition is set the same for all directions and frame orientations of an imaginary end body of the robot.

The inertial systems of medium-high precision (and cost) generally present manufacturing technology and control algorithms contained within the sensor, which limit the effects related to the types of errors typical of the measuring devices.

By performing a study on signals arising from the accelerometer and two gyroscopes of the inertial platform iNEMO, using the Statistics tools.

The board device is mounted on the end-effector of the industrial manipulator KUKA KR3. This makes possible to analyze the behavior of the various signals of inertial sensors when to the robot is set an arbitrary trajectory and performed a number of acquisitions corresponding to it. The aim is to understand whether, from a statistical study of the acceleration, you can get some additional information which allows tracing the reconstruction of the end position of the body every moment of time. The first step is to mount the iNEMO card on the end organ of the robot and to let the manipulator perform and do the same trajectory repeatedly. This will simulate the actual activity that the robot could perform in a factory. At the same time the operation takes place for the acquisition of signals from the inertial unit. For this purpose an algorithm

was implemented to put the computer in communication with the platform, and coordinate the acquisition of the desired data.

Samples taken into account are those relating to the linear acceleration and those to the angular velocity ones. The reason for this is due to the decision to perform statistical analysis on the trend of acceleration during the motion of KR3. And to fulfill this task they need the information contained within these two types of signals. In figure is shown the trend of the three components during the motion of end-effector of the KUKA KR3. 50 signals such as those shown in figure have been acquired!

The axes of the two reference systems fixed to the robot and to the iNEMO are not aligned. Specifically, the x-axis of the reference system iNEMO ($x_{iNEMO}$) coincides with the z-axis of the frame of the end-effector of KR3 ($x_{KR3}$); y-axes iNEMO ($y_{iNEMO}$) and lKR3 y ($y_{KR3}$) match but on opposite direction; and finally the z-axis iNEMO ($z_{iNEMO}$) is aligned with the axis of x KR3 ($x_{KR3}$).

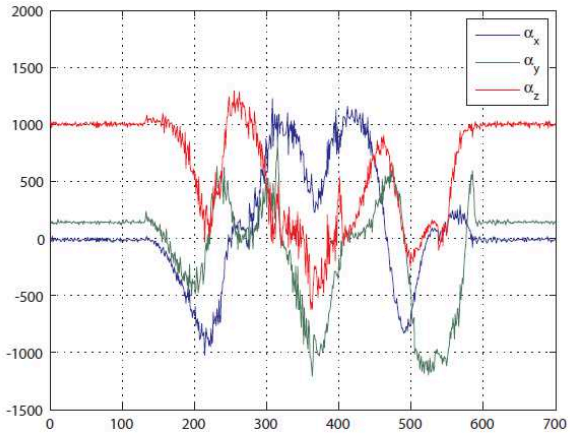



Figure 33. performance of the vector components of linear acceleration during the motion of the robot KUKA KR3

$$x_{iNEMO} \equiv z_{KR3}$$

$$y_{iNEMO} \equiv -y_{KR3}$$

$$z_{iNEMO} \equiv x_{KR3}$$

Of course, even if theoretically the two systems of reference are as shown in the equations there will always be an error, an offset, due to their imperfect alignment. All of this for the purposes of statistical analysis is not an important factor because all the acquisitions were made in succession, without removing the card and then reinstall on the end organ of the manipulator.

In order to achieve the goal of the statistical study of trends, not only for quantitative but also qualitative performances of the various signals, it is necessary to compare all the corresponding components of the 50 acquisitions. The comparison may be feasible and provide the correct conclusions only whether there is synchronization between them. Synchronization is a key factor and will be obtained when each linear acceleration contribution, such as an acquisition, will be aligned from the point of view of time with corresponding signals of the other acquisitions. In fact, some signals will be in advance and still lagging behind others in an acquisition taken as reference. The algorithm that has the function to synchronize the various signals is chosen so that each sample, representing the beginning of the movement of a given acquisition, corresponds to the first sample stored in the time series of that acquisition. The procedure for the realization of the synchronization is based on the use of the information contained within the same acceleration signals. However, a signal from an accelerometer owns, besides the direct information on the linear acceleration, a systematic offset value and, above all, a certain level of noise. This would imply a difficulty. To overcome this obstacle it is necessary that the signal is

properly processed. The block diagram represents the various phases of this development.

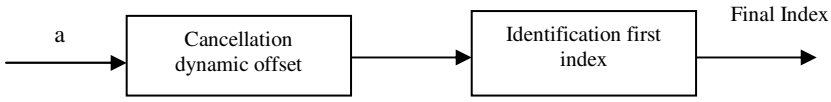



Figure 34. Block diagram of the algorithm of identification of the first index of movement.

Initially we remove the contribute of systematic offsets between the three components of the linear acceleration. Then we run a check on the samples based on the level of noise present in them. After a detailed analysis conducted on this type of signals, we deduct that in stationary conditions the noise level is within the interval [-50mg, 50mg]. It might, therefore, be supposed to implement a so-called control threshold to obtain the exact sample corresponding to the first instant of movement.

$$Ind_{move} : a < t_h \, , \, a > T_H$$

where $Ind_{move}$ represents the first index identified at the check.

This operation, however, is not valid for all linear acceleration contribution signals. In fact, it happens that, for certain elements, we are unable to obtain the true initial sample of movement. To make this control universal, a second one is added.

At the end we get all the samples at the beginning of movements related to various acquisitions, obtaining a vector containing the information regarding the delay or the advance of the corresponding components.

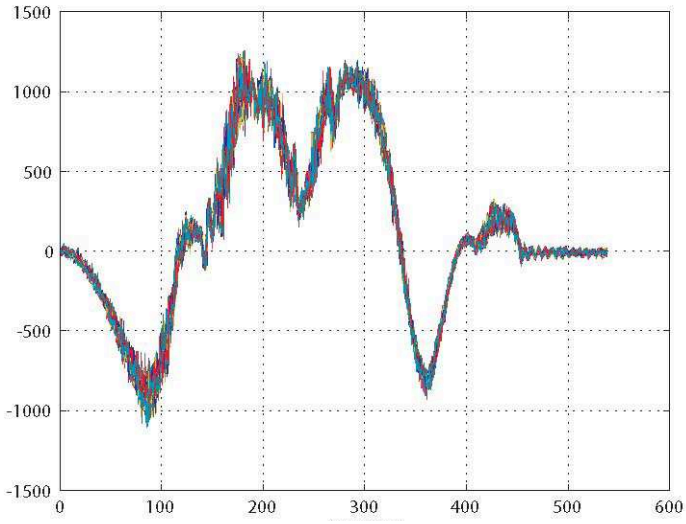



Figure 35. acquisition synchronization

Extracting a sample, at a given instant of time, we can create a distribution function. So there will be as many distributions as the samples contained in a single acquisition. It is evident that the performance of these distributions has a symmetry with respect to a parallel axis to the vertical axis which assumes a nearly bell-shaped. We might assume a *gaussian* performance. For a Gaussian probability curve we can introduce two parameters that uniquely characterize each other. The first parameter is the average value of the normal distribution, denoted by μ, while the second is the average square gap, also known as standard deviation, which is usually indicated with σ. The average value of the distribution is the point where the axis of symmetry passes by. The standard deviation provides a measure of dispersion of data around the expected value.

The normal probability function can be expressed in terms of these two parameters.

$$F(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{z-\mu}{2\sigma^2}}$$

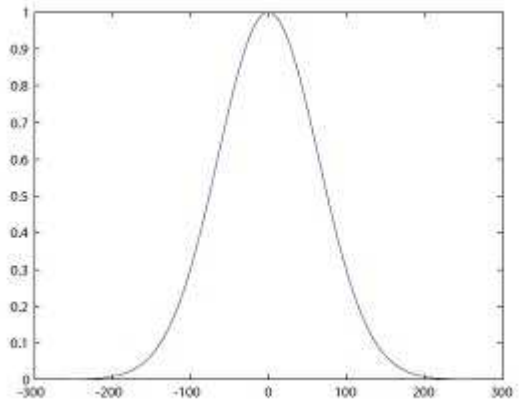



Figure 36. Normal curve of probability

Each distribution of 50 samples, at a given instant of time, gives an average value. By collecting all the average values thus obtained, we build a vector: *the vector of averages* for each of the three components of linear acceleration. The performance quality of these three signals highlights the value of the acceleration during movement by adding the contribution of gravitational acceleration. Consequently, each acquisition is a signal from the accelerometer whose samples are instantaneously or above or below the corresponding sample of the vector of averages and within a certain range of uncertainty. An important factor is attributed to the *vector of residuals* for each component of acceleration and for each acquisition. This vector is calculated from moment to moment by subtracting the average vector to the raw signal. An important element about this signal is to understand whether it is Gaussian white noise, ie if it has a probability function similar to a Gaussian distribution characterized by having the instantaneous values of all unrelated and zero

average. In fact, if this happens for different carrier signals of residuals, it would confirm the hypothesis that the trend of the average quality of the vector represents the acceleration signal of the end organ of the private manipulator of the noise component.

So you might think to use such information for the reconstruction of the trajectory. In order to verify if these signals are Gaussian has been carried out, ie a *test of whiteness*. There are several types of tests of whiteness: the used was the so-called *Quantile plot* (or *Q-Q plot*). This is a graphical method for comparing quantiles of two distributions. Along the abscissa axis we put the quantiles of a distribution; and on the ordinate axis the second distribution quantiles. If the two distributions are similar, the points of the *Q-Q plot* will be found around the line y = x. If the distributions have a linear relationship, then the points will be arranged around a straight line but not necessarily y = x. In our case, the *Q-Q plot* is used to compare the distribution of data with normal distribution. Drawing along the abscissa axis of the sample deployment quantiles, and along the ordinate axis the normal quantiles, if you get the points around the line y = x, then we can say that the data are approximately gaussian. Furthermore, we can say that it is sufficient to calculate and compare empirical quantiles with quantiles of standard normal. If the points representing the pairs of quantiles are arranged along a straight line, the slope and the intercept of this line provides information on the value of $\sigma$ and $\mu$, respectively.

The procedure is applied for all 50 vectors of residuals obtained. Considering a randomly chosen one, we obtain the graph of figure37. The graph shows the Gaussian nature of the various signals from the vector of residuals.

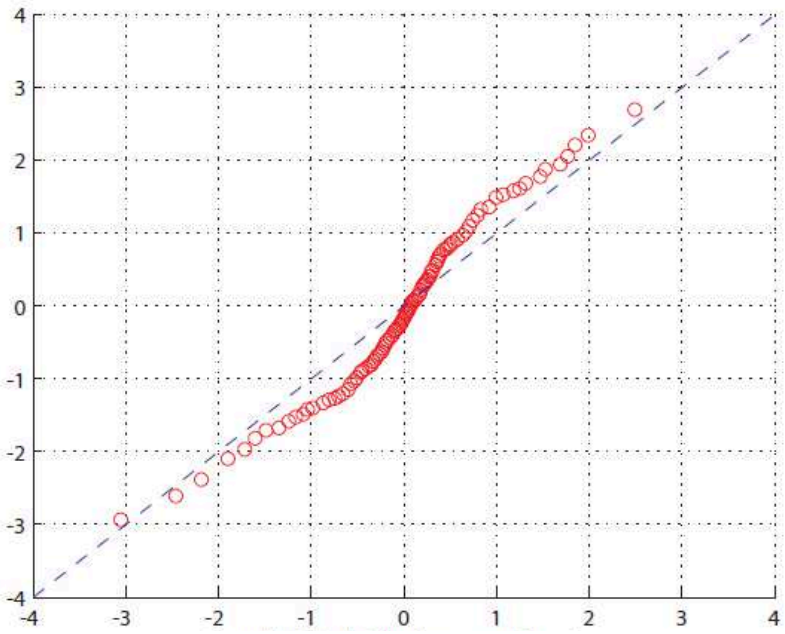



Figure 37. Graph related to Q-Q plot. In blue the reference line.

The carried out statistical analysis shows that, in a certain amount of time, during the motion of the manipulator there is a strong chance that the sample has a value equal to the average of the Gaussian distribution: which is a lower probability that the sample value is within the range, centered on the average value [-3σ, 3σ]. This means that if, at some time during the movement of the end organ manipulator, some unexpected event happens, then the value of the sample at that moment will surely be outside this range. Consequently, with this statistical analysis it is possible to obtain information about the regularity of the movement and then the robot. Unusual situations may occur, for example if the robot accidentally strikes some object, or even human beings. In these cases it is necessary that the activity of the manipulator is stopped almost instantly so not to cause damages. This could cause damage to up the

robot itself and then the company would manage to support unanticipated costs.In order to cope with this kind of undesirable situations it is necessary to understand the exact moment in which the contact took place between the robot and things and / or people. In this respect, it was implemented a control which involves the use of standard deviation and the average moment by moment of the probability distributions derived from the movement of the manipulator. We consider the following conditions:

$$x_i - x < 4\,\sigma_x$$
$$y_i - y < 4\,\sigma_y$$
$$z_i - z < 4\,\sigma_z$$

where $x_i$, $y_i$ and $z_i$ are the samples respectively of linear acceleration along x, y and z at a given point in time of all the acquisitions; x, y and z are the average values of the distributions at the same time: whereas $\sigma_x$, $\sigma_y$ and $\sigma_z$ are the standard deviations distributions in the same instant. To verify the effectiveness of the algorithm implemented during a takeover, the industrial robot has been hit with one hand on a point on the end-effector. This will simulate the conduct of acceleration, obtained from the inertial platform, in case of accidents. As you can see in the figure 38, the quality performance of the three components of linear acceleration has a peak at the moment when the collision occurred.

Now we perform statistical analysis on acquisitions, including the one where the collision occurs. Considering the distribution of samples in the instant of contact, there is a single value that is outside the range $[-4\sigma, 4\sigma]$.

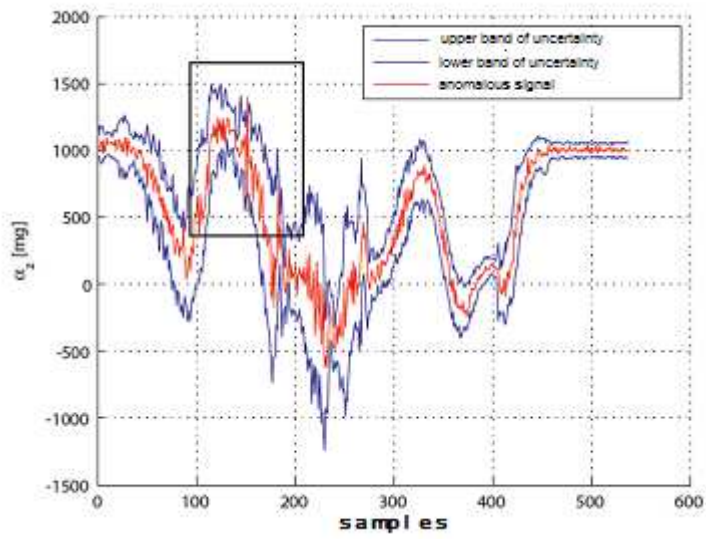



Figure 38. Performance of the component along the z axis of the acceleration in case there is an accidental event. The black rectangle shows such a case.

We make execute to the manipulator a trajectory to simulate a possible behavior within a general industrial automation system. Below we will discuss the *trajectory of reference* for this kind of movement. During this motion the robot is commanded to grasp an object, located in a first point, and place it at a later point. The object in question is a rectangular timber size (5 cm) x (5 cm) x (5.5 cm).

From the foregoing, if during the course of the reference trajectory some unexpected event happens, the statistical analysis on the quality performance of the signals, coming from the accelerometer, will be such as to allow identifying the moment when it has occurred. In fact there will be samples which are beyond the range of uncertainty.

In case of accidental event, samples are obtained at the instant which it occurs that are not within the range of uncertainty.

The so obtained results have shown the validity of the statistical analysis

on the detection of emergency situations that may occur within an industrial automation system.

The development of this approach and the reconstruction of the robot movement may be dependent on the type of movement that has been done to improve the operation of the synchronization, releasing it from the type of trajectory. It has been possible to detect some instants of time, when there is a particularly danger, with high accuracy. Therefore, it might be implemented a check signalizing such accidental situations.

A further step could be accomplished in order to increase the precision and the accuracy by which the iNEMO trajectory is traced.

# CHAPTER III - NAVIGATION AND TRAJECTORY ESTIMATION

## 3.1 SENSOR FUSION ALGORITHM FOR DEAD RECKONING

The Kalman filter used in this application is EKF model (Extended Kalman Filter), the *non linear version* of the Kalman easier filter, because the equations describing the system are non linear. Modeling of the system, the equation of KF is:

$$\dot{\mathbf{x}}_1 = \mathbf{f}_1(\mathbf{x}_1, \boldsymbol{\omega}) + \mathbf{w}_1$$
$$\mathbf{y}_1 = \mathbf{h}_1(\mathbf{x}_1) + \mathbf{v}_1$$

Where:

$$\mathbf{x}_2 = [\mathbf{p}\ \mathbf{v}\ \mathbf{b}_a]^T \in \Re^{9\times 1}$$
$$\mathbf{p} = [\lambda\ \Phi\ h]^T$$
$$\mathbf{v}^t = [v_N\ v_E\ v_D]^T$$
$$\mathbf{b}_a \in \Re^{3\times 1},\ \mathbf{a}^t = [a_N\ a_E\ a_D]^T$$
$$\mathbf{y}_2 = [\mathbf{p}\ \mathbf{v}^t]^T \in \Re^{6\times 1}$$
$$H = [I^{6\times 6} \mid 0^{6\times 3}]^T$$

The ∫ function is:

$$\mathbf{f}_2(\mathbf{x}_2,\mathbf{a}^t)=\begin{bmatrix} \dfrac{v_N}{R_\lambda+h} \\ \dfrac{v_E}{(R_\Phi+h)\cos\lambda} \\ -v_D \\ \dfrac{-v_E^2\sin\lambda}{(R_\Phi+h)\cos\lambda}+\dfrac{v_N v_D}{R_\lambda+h}+a_N-b_{a_N} \\ \dfrac{v_E v_N\sin\lambda}{(R_\Phi+h)\cos\lambda}+\dfrac{v_E v_D}{R_\lambda+h}+a_E-b_{a_E} \\ -\dfrac{v_E^2}{(R_\Phi+h)}-\dfrac{v_N^2}{(R_\lambda+h)}+g+a_D-b_{a_D} \\ \vec{0}\in\Re^{3\times1} \end{bmatrix}$$

The initialization of the system concerns the Q matrix (covariance matrix of errors in the process) and R (covariance matrix of measurement errors). The measures have been performed for a first estimate orientation values of the covariances, which will however be heavily modified during the course of the experiments. Concerning the matrix R, the parameter estimation is based on the behavior of the GPS system. Through the measurements made with the GPS antenna in a static position, the following values have been calculated: values found for PCOV and graphic MatLAB, latitude, longitude and ellipsoidal height. As for the values of the speed covariance, we can determine numerically only for the speed of *0m/s* (antenna in a static position). Therefore, this parameter during experiments has been heavily modified.

The initialization of the state vector of the system is done with a measurement, that is loading the first set of output data from the GPS. Before starting processing, the system waits for a set of GPS data with

fix quality equal to one (GPS) or two (DGPS), which indicates that the fix is reliable, then initializing the system properly. The prediction formulas:

$$\hat{\mathbf{x}}_{\mathbf{i}_k}(-) = \boldsymbol{\Phi}_{\mathbf{i}_{k-1}} \hat{\mathbf{x}}_{\mathbf{i}_{k-1}}(+)$$

$$\mathbf{P}_{\mathbf{i}_k}(-) = \boldsymbol{\Phi}_{\mathbf{i}_{k-1}} \mathbf{P}_{\mathbf{i}_{k-1}}(+) \boldsymbol{\Phi}^{\mathbf{T}}{}_{\mathbf{i}_{k-1}} + \mathbf{Q}_{\mathbf{i}}$$

$$\boldsymbol{\Phi}_{\mathbf{i}_k} = \mathbf{e}^{\left.\frac{\partial \mathbf{f}_{\mathbf{i}}}{\partial \mathbf{x}_{\mathbf{i}}}\right|_k \Delta t}$$

$$\mathbf{w}_{\mathbf{i}} = \mathbf{N}(\mathbf{0}, \mathbf{Q}_{\mathbf{i}}), \qquad \forall \mathbf{i} = 1,2$$

And for correction (update):

$$\hat{\mathbf{x}}_{\mathbf{i}_k}(+) = \hat{\mathbf{x}}_{\mathbf{i}_k}(-) + \mathbf{K}_{\mathbf{i}_k}\left[\mathbf{y}_{\mathbf{i}_k} - \mathbf{h}_{\mathbf{i}}\left(\hat{\mathbf{x}}_{\mathbf{i}_k}(-)\right)\right]$$

$$\mathbf{P}_{\mathbf{i}_k}(+) = \left[\mathbf{I}_{\mathbf{i}} - \mathbf{K}_{\mathbf{i}_k}\mathbf{H}_{\mathbf{i}_k}\right]\mathbf{P}_{\mathbf{i}_k}(-)$$

$$\mathbf{K}_{\mathbf{i}_k} = \mathbf{P}_{\mathbf{i}_k}(-)\mathbf{H}^{\mathbf{T}}{}_{\mathbf{i}_k}\left[\mathbf{H}_{\mathbf{i}_k}\mathbf{P}_{\mathbf{i}_k}(-)\mathbf{H}^{\mathbf{T}}{}_{\mathbf{i}_k} + \mathbf{R}_{\mathbf{i}}\right]^{-1}$$

$$\mathbf{H}_{\mathbf{i}_k} = \left.\frac{\partial \mathbf{h}_{\mathbf{i}}}{\partial \mathbf{x}_{\mathbf{i}}}\right|_k$$

$$\mathbf{v}_{\mathbf{i}} = \mathbf{N}(\mathbf{0}, \mathbf{R}_{\mathbf{i}}), \qquad \forall \mathbf{i} = 1,2$$

At the end of each processing, the updated state of the system is *observed*: the state estimated vector can then be processed or saved on file (datalogging). Two implementations have been made of the filter implementation in real time and post processing (the latter only implemented at MatLAB).

The state vector of the system is initialized if, at the reading of the sample, the field FixQual of the GPS data structure is *not* less than one. In other words it must take the values one or two; otherwise it means that the output data from the GPS are not *valid* (not reliable, the number of satellites is not sufficient to ensure adequate precision or no satellite is

received) and therefore the system restarts the calibration phase. So the *euler* vector is loaded with data from roll, pitch and yaw. Now the direction cosines matrix is built, which will provide to rotate the data of the accelerometers from the body-frame to the navigation-frame, including the rotation in this matrix about 2° 25' due to the magnetic declination. The matrix of direction cosines is then multiplied for *dt*, the time elapsed since the previous run of the filter to the immediate present. In this case 1/50 $Hz^{-1}$, 0,020 s. Then the matrix F of the system is built:

$$F = \begin{bmatrix} \cdot & \cdot & \cdot & \frac{dt}{R_{ns} + Alt} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \frac{dt}{(R_{ew} + Alt)\cos(Lat)} & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & -dt & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -dcm_{11} & -dcm_{12} & -dcm_{13} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -dcm_{21} & -dcm_{22} & -dcm_{23} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -dcm_{31} & -dcm_{32} & -dcm_{33} & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}$$

and the G matrix:

$$G = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ dcm_{11} & dcm_{12} & dcm_{13} & dt & \cdot & \cdot \\ dcm_{21} & dcm_{22} & dcm_{23} & \cdot & dt & \cdot \\ dcm_{31} & dcm_{32} & dcm_{33} & \cdot & \cdot & dt \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}$$

From following lines we see the discretization of the system in the time domain:

$$G_d = \left(\frac{1}{2}F + I\right) * G$$

$$F_d = F + I$$

where the subscript *d* indicates the discrete-time domain.

We construct now the inputs vector:

$$u = \begin{bmatrix} a_x \\ a_y \\ a_z \\ . \\ . \\ 9.81 \end{bmatrix}$$

*Prediction of the state*

$$x_k = F_d * x_{k-1} + G_d * u_{k-1}$$

In order to how the matrices have been built, we have got:

$$x_k = x_{k-1} + F * x_{k-1} + \frac{1}{2}F * G * u_{k-1} + G * u_{k-1}$$

By developing calculations, the first line is:

$$LAT_k = LAT_{k-1} + \frac{1}{R_{NS} + Alt}\left(v_N * dt + \frac{1}{2}\left(DCM_N(a_x, a_y, a_z)\right)dt^2\right)$$

for the component of the latitude (North direction, increasing latitude), and similarly the second and third for East and Down components of the Navigation frame. The multiplication factor: $\frac{1}{R_{NS}+Alt}$ derives from the formula for calculating the angle at the center of a circular sector ($\alpha = \frac{a}{r}$ where $\alpha$ (in radians) is the angle at the center, $a$ the arc length subtended angle $\alpha$ and $r$ the radius of the circle). The endorsement $DCM_N(a_x,a_y,a_z)$ we wanted to mean the resulting component of the acceleration in the direction of North, obtained with the application of the matrix of the direction cosines (DCM, Director Cosine Matrix) to the triplet of accelerations in the *body frame*. For the third, fourth and fifth row, we have:

$$v_{Nk} = v_{Nk-1} + DCM_N\left(ba_x, ba_y, ba_z\right) * dt + DCM_N\left(a_x, a_y, a_z\right) * dt$$

with the same notations used before.

These equations are formally identical to the classical formulas of uniformly accelerated motion:

$$x = x_o + v * dt + \frac{1}{2}a * dt^2$$

$$v = v_0 + a * dt$$

For $dt$ sufficiently small, these can be considered an approximation of the general equations of the movement.

*Prediction of the covariance matrix of the errors:*

$$P_k = F_d * P_{k-1} * F_d{}^T + Q$$

*Update and Calculation of the Kalman gain (optimal)*

$$K_n = P * H^T * (H * P * H^T + R)$$

*Update of the covariance matrix of the errors*

$$P = (I - K * H) * P$$

Update of the state vector

$$x_{a\ posteriori} = x_{a\ priori} + K * (y - H * x)$$

Tests were conducted using a car: the system board iNEMO + GPS has been applied to a vehicle. The board iNEMO is integral with the car, as well as the GPS antenna. The interface software iNEMO integrates a Kalman filter and the output data are recorded in a tsv format file on your PC. The input data to the filter are also saved in order to perform subsequent processing (post processing). The first step is to calibrate the initial asset in general, during an acquisition, it is not known a priori the structure of the board of iNEMO (angles of roll, pitch and yaw). For this reason it has been needed to introduce the following constraint on measures: for the first ten seconds, the board is steady maintained, subject only to gravity, in order to mediate the static measurements of this initial interval, and use these values to realign (computationally) the board in the next instants. The output of the application consists of data logging concerning the various measured and estimated quantities data. A file is saved in "*.*tsv*" format with 28 columns for each row: the results are shown in the following images:

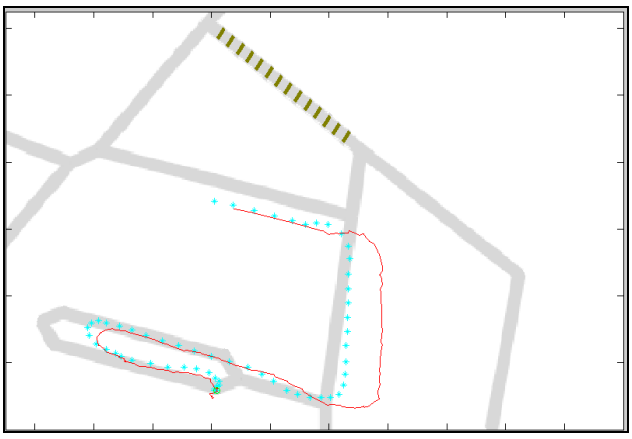


Figure 39. Results of the Dead Reckoning algorithm software implementation

We shall notice that long set of measures have been carried out with acceptable results in terms of *stability* of the algorithm. However, errors in calculation that in some cases have occurred during experiments - likely due to the calculation of the inverse matrix - spread in a disastrous: the next results were therefore not acceptable and made it necessary to repeat the experiment.

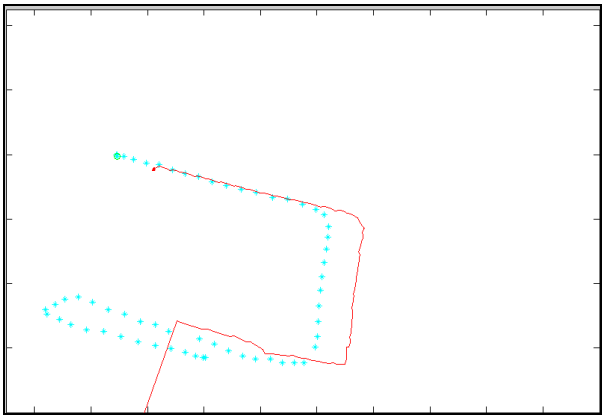


Figure 40. Example of failure of one of the first versions of the real time algorithm

The values of the covariances are chosen ad hoc, depending on the chosen route and its features (bumps in the road, any minor obstacles that cause unwanted solicitations), weather conditions, the positioning of the boards inside the car; this at least in the early stages of the software development.

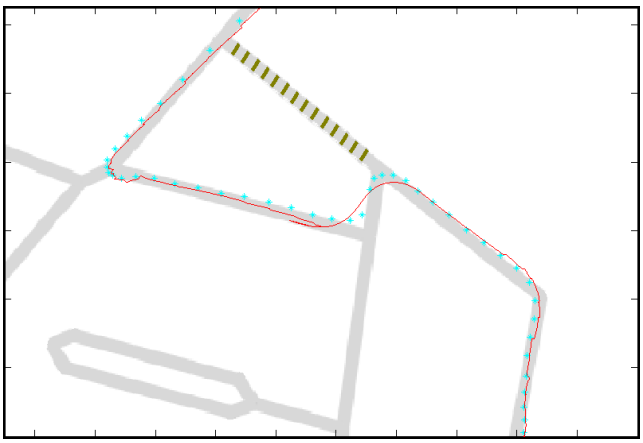


Figure 41. Dead Reckoning

Per le caratteristiche intrinseche di un algoritmo che viene eseguito in real time, il processo di tuning dei valori delle covarianze è, di fatto, complesso da effettuare. Per la rielaborazione dei dati in post processing si è utilizzato MatLAB, in modo da osservare il comportamento del sistema al variare dei parametri relativi alle covarianze (Matrici Q ed R), ma operando sul medesimo treno di valori.

The covariance of the values of speed and position in the Kalman filter can be viewed as the degree of trust that the system gives to the information coming from GPS: small covariance = we assume that mistakes committed by the GPS are small, the system trusts GPS; high covariance = GPS is not very precise (eg because it can receive few satellites), therefore the system gives less weight to the GPS fixes. For

sake of conciseness, we will denote with $PCOV^2$ the covariance of the coordinate values of the position coordinates, latitude and longitude, and instead with $VCOV^2$ the covariance of the values of velocity components, $V_N$ and $V_E$;

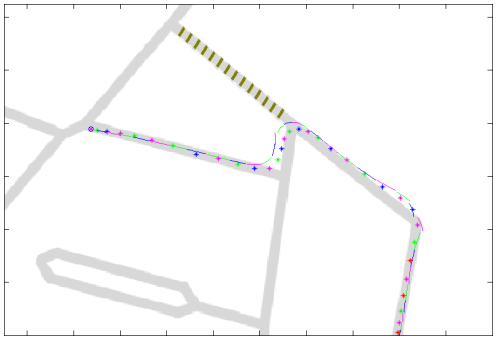


Figure 42. Dead reckoning: implementation at MatLAB.

Incorrect values for the covariance: $PCOV = 2.0e^{-4\frac{\pi}{180}rad}$ equivalent to an uncertainty of almost 22 meters and VCOV = 0.1 m/s.

The optimal value for the covariance matrix has been obtained through successive approximations, as the values, derived from measurements made under static conditions, were inadequate.

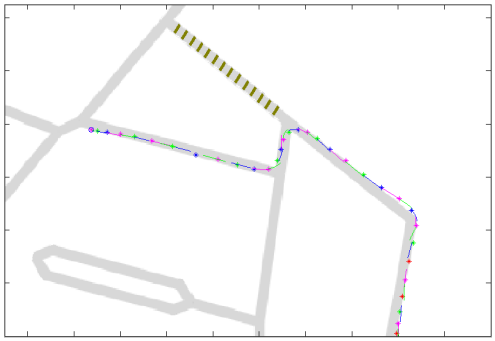


Figure 43. Dead reckoning: implementation at MatLAB.

Improve values for the covariance: PCOV = 2.0e-5* rad, equivalent to incertitude of almost 2.2 meters and VCOV = 2 m/s

Below there are the corrections plots made by the process of updating the Kalman filter. In case of GPS developed regularly, we noticed an increase with the speed error and a discrepancy error after n steps. Another important aspect, highlighted by the analyses carried out at MatLAB, concerns the errors in close proximity to lateral accelerations, so about changes in the trajectory. By reviewing the IMU behavior compared to the corresponding GPS fixes, it was observed that the trajectory shows an advance over time compared to the actual trajectory - clearly due to delays in the acquisition of the GPS data sets. Such delays are due to:

- delay due to the incoming data processing from satellites, processed by the GPS sensor itself;
- delay due to transmission of data from the GPS to the module iNEMO, the transmission of these data through RS232 protocol involves a delay of about 10ms;
- search of the strings in the USART buffer and extrapolation of the values related to the GPS from the RMC and GGA strings;
- sending data to the PC;

This delay has been assessed overall by working a shift of the output values by the IMU compared to those of GPS. These delays could be more restrained by integrating the software part related to data development directly in the microcontroller firmware.

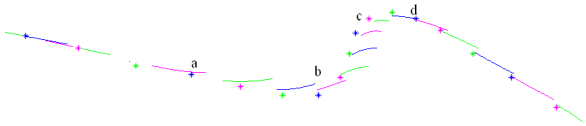


Figure 44. Explanation of the GPS time lag compared to the IMU one: chart without time shift. Travel direction is from left to right.

In correspondence to the fix a, in rectilinear trajectory, the IMU is (already) sending the lateral acceleration on the first corner. At the fix in b, in the middle of the turn, the IMU is sending data to the straight section between the two turns. In the c the IMU provides information on the straight section that follows the d, while the GPS signals locates a position in the middle of the turn.

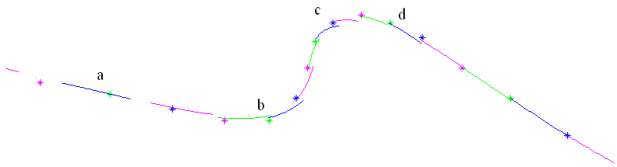



Figure 45. Explan Explanation of the GPS time lag compared to the IMU one: graph with 1.4 s time shift. Travel direction is from left to right.

For visual inspection we can see a correspondence between the curvature of the trajectory reconstructed with the IMU data (continuous dash) and the trajectory indicated by the GPS (starry path).

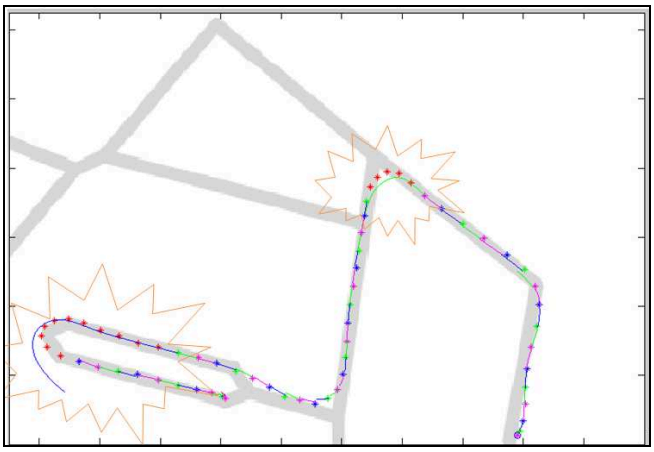


Figure 46. Dead reckoning: two examples, respectively 5 and 11 seconds

The dead reckoning is a computational procedure estimating the path of a car without the help of the navigation sensor. In this case the GPS is present, but not used in the features highlighted in orange, where the fixes are in red. In those sections the GPS fixes are not considered for the update phase of the Kalman filter, thus simulating the absence of GPS signal (while plotting fixes to examine the trajectory actually followed the car.

Figure 47. The plot superimposed on the aerial photo the involved area

The commercial navigation systems are generally composed by a GPS receiver detecting the position on the Earth moment by moment, a database containing the maps of a particular geographical area, and a processor - capable to calculate the path connecting the departure point (the current position) to the point of arrival (the target - the desired destination). The route calculation is typically made using Dijkstra's algorithm. While driving, the computer verifies that the user is actually following the recommended route and if for some reason he/she can not (or will not) follow the recommended route - for eg impediments due to roadwork or for intense traffic - the signal from the GPS receiver detects

the change made to the route and *recalculates* a new route, considering as a new starting point the coordinates received from GPS.

Let's consider the case illustrated in the following figure: we are in the *P,* point of departure, and set the target to reach the end point *T* - the target. The route recommended by the navigator will be *P* - *A* - *T*. Let's suppose that near the point *A* GPS coverage is not available - for example due to a section of covered road - and that for reasons of force majeure - such as a road construction site - we can not go straight, but we are forced to turn in the direction of point *B*. Continuing straight ahead, when near point *C* the GPS signal is received again, the navigator will recalculate and advise you the path *C* - *E* - *F* - *D* - *T*. Whereas, if the navigator is equipped with an inertial platform and a sensor fusion algorithm - such as the type developed in this research - it will be able, at the point *A*, by receiving the data related to the path deviation path (left turn), to make immediate recalculation, advising the user to turn right into *B*, following the path *B* - *D* - *T*, with a significant path optimization.

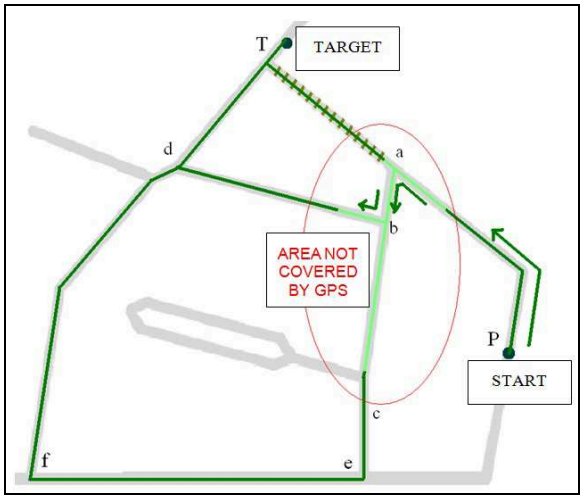



Figure 48. A possible application of the dead reckoning algorithm when integrated into a commercial navigation system

The test results demonstrate the capabilities of these techniques used in an actual field of use: a moving vehicle with an average speed of 25 Km/h in adverse conditions of GPS signal. The algorithm is based on the Extended Kalman Filter, in other words a sensor fusion algorithm capable of predicting and correcting the single measures.

## 3.2 3D NAVIGATION PRESSURE SENSOR

The atmospheric pressure varies with altitude, as well known; in fact the air, as it passes through the various layers of the atmosphere, becomes more and more rarefied, and consequently decreases the pressure it exerts. Obviously, this concept asks for clarifications, in fact, it is not said that the atmospheric pressure remains constant at the same altitude level: turbulences, temperature and humidity variations can contribute to its significant variation, and to unclear altimetry information. Using MEMS membrane, we can make a barometric reading; air particles when depositing on the membrane determine its deflection and appropriate strain-gages will provide electrical reading modulated by instrumentation amplifiers.

Before defining the curves of characterization (pressure-height) it is appropriate a short presentation on the units of measurement of pressure and the related conversions:

$$1Pa = 10^{-5}\,bar = 10^{-6}\,N/mm^2 = 9.87 \cdot 10^{-6}\,atm = 0.0075 torr$$

In this case the sensor is described by a characteristic function which correlates the input magnitude, which is the pressure, with the output

magnitude: height. Assuming, in fact, that the sensor is properly calibrated a first relation could be the following:

$$\Delta h = 44330 \cdot \left( 1 - \left( \frac{P}{P_0} \right)^{0.19} \right) - h_{rif} ;$$

Where $h_{rif}$ is the height reference calculated at pressure $p_{rif}$ and $p_0$=101.325 KPa; obviously the thermal noise and quantization are added to real information making necessary the use of filtering functions such as FIR and *smooth*, described in detail in the following paragraphs.
However, applying the above formula and the only filter functions do not allow accurate altimetric information, the error may even be up to 60%, which is why it has been necessary deriving a functional relationship between pressure and height via a calibration based on mistake minimization:

$$p = f(h)$$

The estimation of functional parameters can be performed through a series of experimental measurements and applying statistical regression.
Calibration results from exposure of the sensor to it; by measuring the environmental conditions, in fact, they may affect the response, so in this case by placing the pressure sensor to a certain height, which in this case is the input - the known information -, and making a series of acquisitions repeated in several intervals of time, we can understand what is the error on pressure, minimizing it with appropriate algorithm.

To be precise, from 1 to 3 acquisitions have been carried out for each step of height, repeated every minute (1 meter for step, since the sensor resolution is 80 cm) with a thousand samples per acquisition.

Pressure and height are two measurable quantities in functional relation between themselves, this report may be linear or a variable degree polynomial expression:

$$p = g(h, k_1, k_2, ....., k_m) = k_1 + k_2 h + k_3 h^2 + ..... + k_m h^{m-1}$$

If no measurements errors occur, the functional relation $p = g(h)$ will be correct, if thus we perform n measurements $h_i$, $p_i$ we obtain:

$$p_i = g(h_i)$$

In an actual case such is this, the functional relationship is not exact; so it is possible to clearly explain a generalized form in which height is affected by error, and parameters k of the polynomial expression are so given:

$$p_i = g((h + e_i), k_1, k_2, ..... k_m)$$

Before proceeding with the calibration algorithm it is necessary to introduce some hypothesis:

- $e_{pi} >> e_{hi}$, the measure error at sensor reply higher than stress;
- the p variable has a gaussian distribution;
- N events are independent among them;
- all measures have same variance;

In order to obtain the calibration curve we shall minimize compared to individual parameters, resulting m equations in m unknown:

$$\begin{cases} \dfrac{\partial}{\partial k_1}\sum_{i=1}^{n}(p_i - g(x_i, k_1, ....., k_m)) \\ ..... \\ \dfrac{\partial}{\partial k_m}\sum_{i=1}^{n}(p_i - g(x_i, k_{1,} .....k_m)) \end{cases}$$

Recalling the expression of the pseudo-inverse matrix:

$$X^{+} = (X^{T}X)^{-1}X^{T}$$

now just apply the Gauss-Markov theorem and solve the problem at least squares expressing, as a result of carried out measurements, the polynomial function on matrix form and obtaining the coefficients by means of MatLAB calculation software.

$$p = a_0 + a_1 \cdot h + a_2 \cdot h^2 + ..... + a_n \cdot h^n$$

The former applies in the event of continuous time, of course if we consider the sampling of sensory information, assuming that we have available m samples then you will get the discrete-time expression:

$$[p_1 .....p_m] = [a_0 .....a_n] \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ h_1 & .. & .. & h_m \\ .. & .. & .. & .. \\ h_1^n & .. & .. & h_m^n \end{bmatrix} + [e_1 .....e_n];$$

Where $p_1 .....p_m$ are the sampled pressures, $h_1 .....h_m$ the heights, $a_1 .....a_n$ are the polynomial expression coefficients, and last $e_1 .....e_n$ are the errors related to pressure, errors that should be minimized. The former can be expressed in compact matrix form - this also to easily understand the algorithm of Gauss-Markov - and in this way is

guaranteed the error minimization and the computation of the polynomial expression coefficients by the product between the vector of the pressures and the pseudo-inverse of height:

$$P_{1\times m} = K_{1\times n+1} \cdot H_{n+1\times m} + E_{1\times m};$$

$$\min\left\|E_{1\times m}\right\| \longrightarrow K_{1\times n+1} = P_{1\times m} \cdot H^{+}_{n+1\times m};$$

The expression found concerns the pressure as a function of height, but, however, the calibration curve should express height depending on the pressure.

However, through the *polyfit* function, giving on input first the vector of pressures (on the abscissa) and then the heights (ordinate), it has been possible to obtain the final polynomial expression, actually to apply the minimization, the heights have been initially hypothesized as inputs having minimum error.

$$h = a_0 + a_1 \cdot p \ + ..... + a_n \cdot p^n;$$

$a_0 = -4.77 \cdot 10^{-23}$

$a_1 = 1.18 \cdot 10^{-17}$

$a_2 = 2.27 \cdot 10^{-13}$

$a_3 = -2.3 \cdot 10^{-7}$

$a_4 = 0.0013$

$a_5 = 3708$

$a_6 = -3.27 \cdot 10^{9}$

$a_7 = 8.66 \cdot 10^{-12}$

In the following figure we may see the output graphic to the MatLAB *polyfit* function*:*

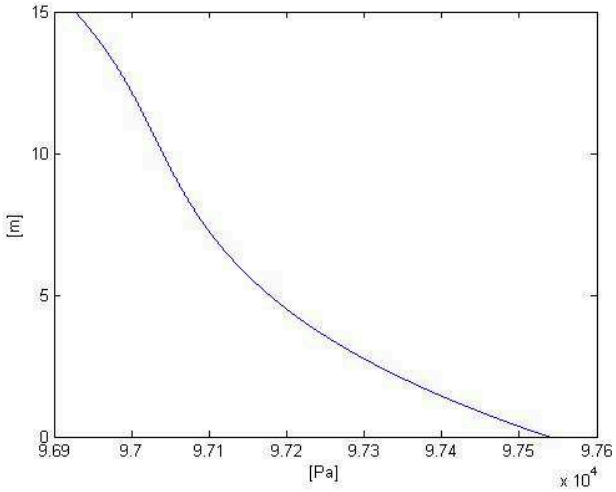



Figure 49. Calibration Curve.

Estimation Height through the Kalman filter: through the Gauss-Markov theorem, by which it was traced to the calibration curve with a polynomial expression allowing to obtain the altitude information.

However, for a proper reconstruction of the heights besides the pressure sensor the accelerometer present on the board can be used; thinking of integrating twice the acceleration along the z axis of the system.

At this point you can think of matching the two pieces of sensory information and of a mathematical model based on prediction and correction of the actual position of the system.

For this reason we applied the Kalman filter, in fact, starting from a system in the form of state, which in this case is derived from the equations of kinematics, and considering statistical information about errors related to state variables and measures, it has been possible to

reconstruct the altitude position with accuracies ranging between 2% and 10%.

We have considered a mathematical model in the form of state based on the equations of kinematics, so as to be able to fix the two Kalman states: speed and position.

$$h(t) = h(t_0) + T \cdot v(t_0) + \frac{1}{2} \cdot a_t(t) \cdot T^2$$
$$v(t) = v(t_0) + T \cdot a_t(t)$$

Moving on to a matrix expression in discrete form we obtain:

$$\begin{bmatrix} h(k) \\ v(k) \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} h(k-1) \\ v(k-1) \end{bmatrix} + \begin{bmatrix} \frac{1}{2} \cdot a(k) \cdot T^2 \\ a(k) \cdot T \end{bmatrix}$$

Obviously in the ideal case we neglect measurement errors related to acceleration and heights, through which is possible to define the status and output uncertainties. We must therefore rearrange the equations by introducing the measurement noise to implement the Kalman filter and initialize the matrices.

We shall begin this section with an overview of the high-level operations of the algorithm for the discrete Kalman filter that we want to be treated. So we will analyze in detail the specific equations and their use in this version of the filter.

The Kalman filter provides an estimate of a process using a form of feedback control: the filter estimates the process, at a certain moment, and get feedback in the form of measurements affected by noise.

The equations of the Kalman filter fall into two groups: *time update* equations and *measurement update* equations. The former are responsible

for the prediction of the current state and covariance of the error, evaluated to obtain an a priori estimate for the next step. The *measurement update* equations are responsible for feedback: they are used to join a new measurement with the a priori estimate to obtain improved estimate a posteriori.

The *time update* equations can also be thought as prediction equations, while the *measurement update* equations represent the equations of correction. Indeed the final estimation algorithm looks like a predictor-corrector algorithm for solving numerical problems.

The time-update equations are the following and allow a prediction on the state:

$$\hat{x}(k+1) = A \cdot x(k) + B \cdot u(k)$$
$$\hat{P}(k+1) = A \cdot P(k) \cdot A^T + Q(k)$$

Where the matrices A and B concern the mathematical model of the system, matrix P is the solution of Riccati equation, while the matrix Q is the covariance matrix relative to the uncertainties about the state modelled as Gaussian distribution stochastic processes.

The next step is the *measurement-update*, the first operation during the update by means of the measurement consists of calculating the gain K from the matrix R(k) which includes measurement error covariance.

$$K(k) = \hat{P}(k)H(k)^T (H(k)\hat{P}(k)H(k)^T + R(k))^{-1};$$

The next step foresees to measure the process and then determining the a posteriori state estimate, obtained by incorporating the measurement.

$$x(k) = \hat{x}(k) + K(z(k) - H(k)\hat{x}(k));$$

Now we update matrix P, the solution of Riccati equation:

$$P(k) = (I - K(k)H(k))\hat{P}(k);$$

At this point we have to adapt to the system under investigation the filter, defining the model with errors and related matrices. Considering that the measured quantities are subject to errors, they can be defined into the system on status form $w_k$ and $\varepsilon_k$, or white noise, independent and with Gaussian distribution: process error is defined, as $\varepsilon_k$ represents the measurement error.

The input acceleration to the system and the output variable have the following expression:

$$u = a + \Delta a;$$
$$y = h + \Delta h;$$

where $\Delta h = \varepsilon = 0.5m$ e $\Delta a = 0.001m/s^2$, or measurement errors related to acceleration and heights after a first filtering, at this point by replacing in the model above:

$$\begin{bmatrix} h(k) \\ v(k) \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} h(k-1) \\ v(k-1) \end{bmatrix} + \begin{bmatrix} \frac{1}{2} \cdot a(k) \cdot T^2 \\ a(k) \cdot T \end{bmatrix};$$

$$y(k) = h(k) + \Delta h(k);$$

In the mathematical model in form of state there is only the measurement error. In fact, in order to apply the Kalman filter we will need to know the error status so that we can calculate the covariance matrices, without the covariance matrices it is not possible to implement the phase of update of the filter. So we examine the error on the accelerations (system

input) that isolated from the average value grant the calculation of the vector $w_k$ :

$$\begin{bmatrix} h(k) \\ v(k) \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} h(k-1) \\ v(k-1) \end{bmatrix} + \begin{bmatrix} \frac{1}{2} \cdot (a(k) + \Delta a(k)) \cdot T^2 \\ (a(k) + \Delta a(k)) \cdot T \end{bmatrix};$$

$$y(k) = h(k) + \Delta h(k);$$

$$\begin{bmatrix} h(k) \\ v(k) \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} h(k-1) \\ v(k-1) \end{bmatrix} + \begin{bmatrix} \frac{1}{2} \cdot a(k) \cdot T^2 \\ a(k) \cdot T \end{bmatrix} + \begin{bmatrix} \frac{1}{2} \cdot \Delta a(k) \cdot T^2 \\ \Delta a(k) \cdot T \end{bmatrix};$$

$$y(k) = h(k) + \Delta h(k);$$

We derive now the expressions of the errors on the state that result white organized in a vector 2X1, mutually uncorrelated and with known covariance matrix:

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \cdot \Delta a(k) \cdot T^2 \\ \Delta a(k) \cdot T \end{bmatrix};$$

$$E[w(j)w(k)^T] = Q(k);$$

$$Q(k) = \begin{bmatrix} w_1^2(k) & 0 \\ 0 & w_2^2(k) \end{bmatrix};$$

The implementation of the filter requires knowledge of the P matrix at the initial instant (matrix defined $P_0 \cong Q$ ), but also the covariance matrix R related to the barometric height error:

$$E[\Delta h(j)\Delta h(k)^T] = R(k) = \Delta h(k)^2;$$

Once filtered data, the algorithm begins, initially the system is stationary so the standard deviation of accelerations is below a given threshold (determined empirically after some testing off-line).

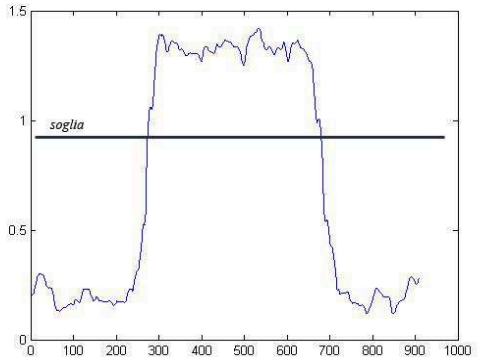



Figure 50. Standard deviation of accelerations along the axis Z.

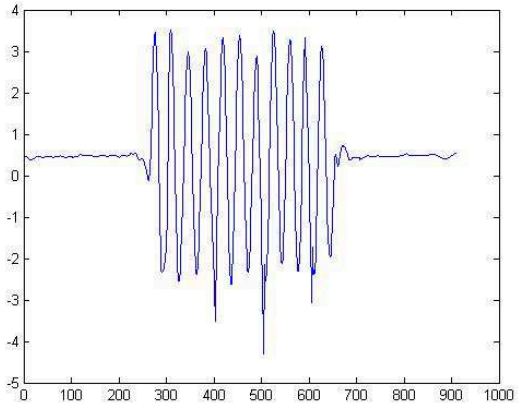



Figure 51. Accelerations along axis Z.

The system does not feel accelerations, so the Kalman filter is implemented; but we shall nevertheless proceed to the calculation of the height of reference by placing the inlet pressures to the calibration curve previously calculated. As soon as the system begins to move along the Z

axis, the standard deviation exceeds the threshold limit and the Kalman filter is implemented as described above. Of course, numerical arrangements were also introduced to deal with problems of integral drift in both phases of ascent and descent stages. For example, if you consider a person who is walking along a flight of stairs is clear that the speed along the Z axis is reset at each step, of course this does not occur in the algorithm because of the numerical drift. To cope with this problem we bind that the speed is zero at each step, and this applies both in the rising phase in which the acceleration is positive when restart, and negative just before we leave, and during the descent where the situation is reversed, ie, positive acceleration just before stopping and negative at the restart.

If the system should stop, of course, the standard deviation of accelerations falls below the threshold, therefore, no longer experiencing accelerations at the axis Z is momentarily interrupted the implementation of the Kalman filter, we bind that the speed is null and the reference height is the maximum predicted height (or minimum in case of fall). We can understand the proper functioning of the algorithm, simply analyzing the graphs for height estimated by the filter and height calculated using the calibration curve.

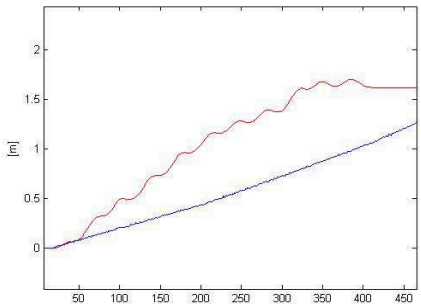



Figure 52. Height calculated through calibration curve.

By changing the variances of the matrix R, it has been possible to modulate the filter response. For values $\Delta h$ close to zero, the altitude response of the calibration curve prevails with errors up to 50%; increasing up to 0.3 the prediction of the Kalman filter prevails with error close to 3%; and finally for the variance values next to 1 error increases again, mainly because we assign weight to the accelerations along the Z axis. Verified the correct functioning of the algorithm it has been considered appropriate its implementation real-time, in other words an instantaneous reconstruction of the trajectory during the travel of a climb or descent. In the first step of development we have sought to establish the number of samples to be passed to the *smooth* functions, this is to get the best possible compromise between filtering and computation time. It is easy to understand, in fact, that if we introduce in the algorithm a buffer with large number of samples, while very well filtering the data, the calculation time increases considerably.

In the present case the algorithm, before starting the filter, expects that data from the pressure sensor are loaded into a buffer of 2000 samples. These are then filtered by a smooth function *lowess* type; in reality we also compared functions type *loess, sgolay*, *rloess* and *rlowess*, all allow good filtration but differ considerably the computation time. And in fact the *lowess* and the *sgolay* are the fastest. In the second step we tried to understand how to update the buffer in real-time.

For accelerations we can simply upgrade the buffer by adding a sample iteration by iteration using a technique mobile cell-tail; it consists in keeping fixed the buffer size by eliminating the first sample for each datum that is added, as shown in the following figure. It is not possible

the same reasoning for pressures, in fact, updating the buffer with a sample at a time, just using the same technique used for accelerations, the output data to the filter have a pattern which does not provide any information about altitude. Then it was considered appropriate to update the buffer with a single sample, but with a subarray consisting of 200 samples using a technique mobile cell-tail.

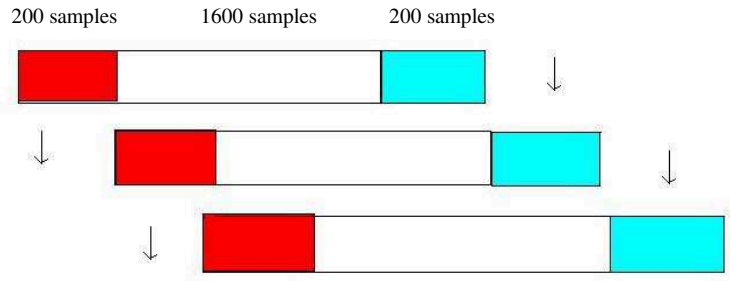



Figure 53. Update of the pressure buffer

In this case greatly improves the response of the filter but to the detriment of computation time; in fact, though as the sampling frequency is 50 Hz, little more than 4 seconds need before the altitude response is updated. Not only that, another problem due to this type of filtering is a numerical offset between two subsequent buffers, which must be calculated and corrected. At this point accelerations and pressures are filtered to the algorithm described above in the case. In order to assess when the altimetric Kalman filter is accurate, it has been required a comparison between the real path - in this case the reconstruction of 2 ramps up - and the height provided by the algorithm. The following figures show the estimated filter trajectory in red and in blue the real path in the trajectory OYZ.

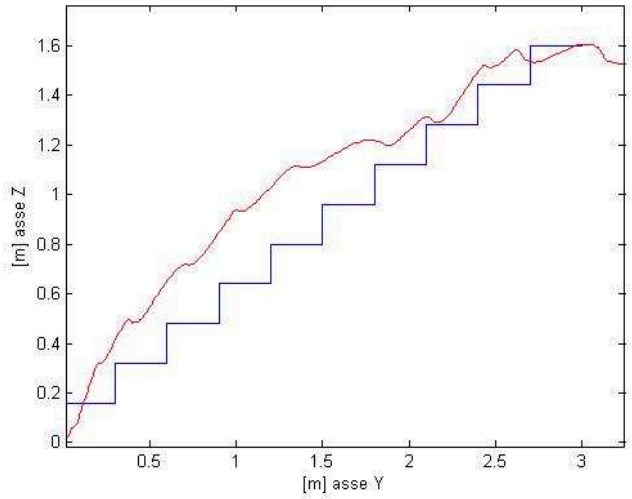



Figure 54. Comparison between real (blue) and estimated trajectory (red).

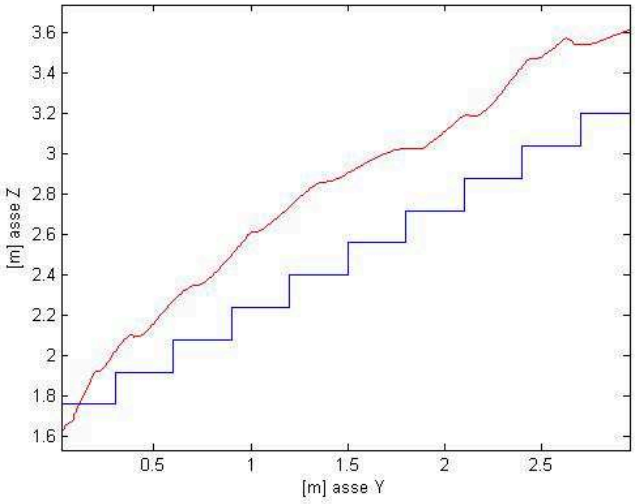



Figure 55. Comparison between real (blue) and estimated trajectory (red).

Taking different points of reference, 11 for each ramp, it has been possible to analyze the error. Whereas the maximum peak, as deduced from the following charts, is about 0.6 m on a maximum height of 3 meters of real total height. It follows a percentage of error ranging from 0

to 20%. So the use of height-pressure calibration curve and the Kalman filter leads to a significant reduction of the error, which using standard barometric reports previously seen can be as high as 70%

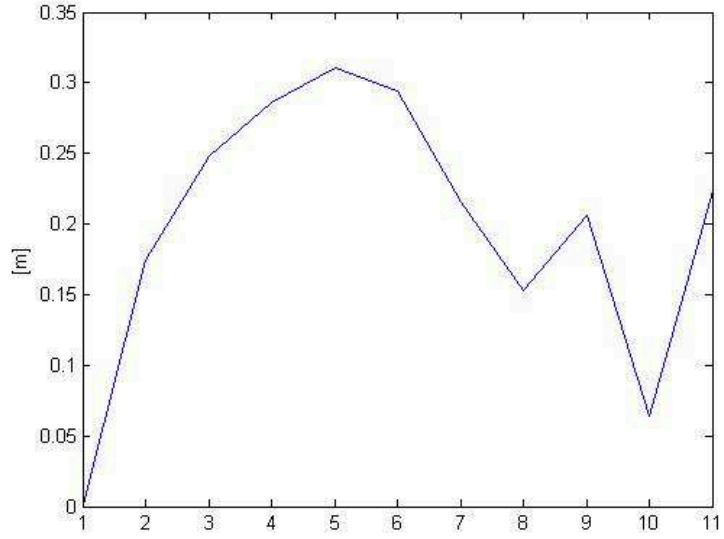



Figure 56. Error related to the first ramp.

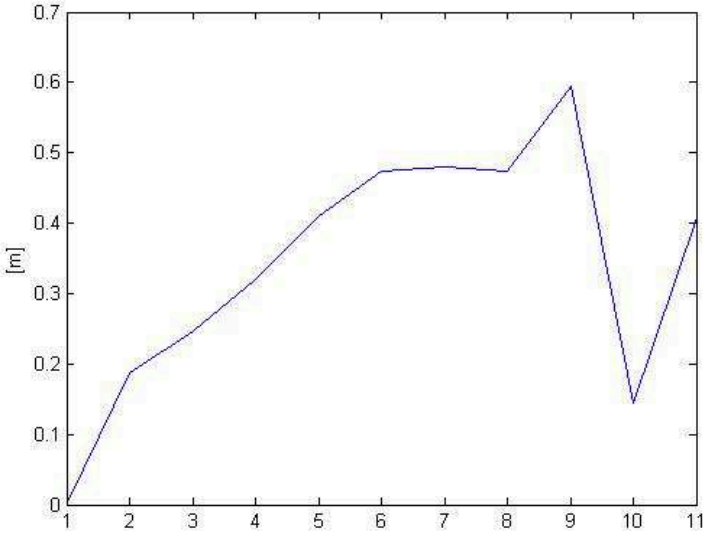



Figure 57. Error relate to the second ramp.

An example of application of the previous algorithm has been implemented during the NATO-ASI Conference which was held in Cesme in Turkey in August 2010, through the helicopter developed by the Coax ETH in Zurich, which uses a sonar to virtually determine the height. This can not make possible the flight independently in rooms where there are objects of different heights, given the impossibility of establishing a fixed distance from the ground. Interacting with the working group it has been suggested the use of the pressure sensor in the board control of the Coax, instead of the sonar to determine a fixed height of flight so that the helicopter maintained it independently. In the figure below we can see the block diagram of the implemented system, the log file was uploaded to the PC via the bluetooth communication. The good results obtained during tests of the algorithm allow us to use the system implemented in a way to employ the pressure sensor along with the accelerometer for reconstruction and navigation 3D. For a proper reconstruction of the heights, as well as the pressure sensor, the accelerometer can be used on this board, thinking of integrating twice the acceleration along the z axis of the system. At this point we can think of a matching between the sensory information and a mathematical model based on prediction and correction of the actual position of the system. For this reason we applied the Kalman filter. In fact, starting from a system in state form, which in this case is derived from the equations of motion, and considering statistical information about errors related to state variables and measures, we can rebuild elevation position with a ranging accuracy between 2% and 10%. In the NATO ASI Conference, it has been developed a height control based on this work jointly with other students (see figure 58).

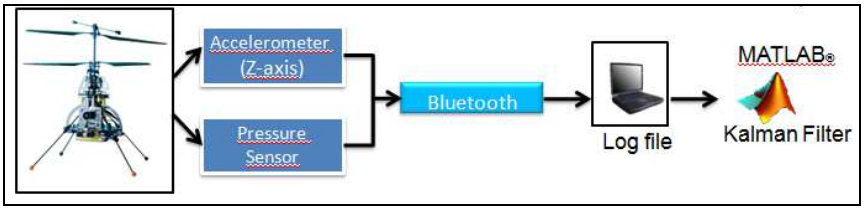



Figure 58. Block scheme of Height reconstruction and navigation using a pressure and accelerometer sensor in a coax

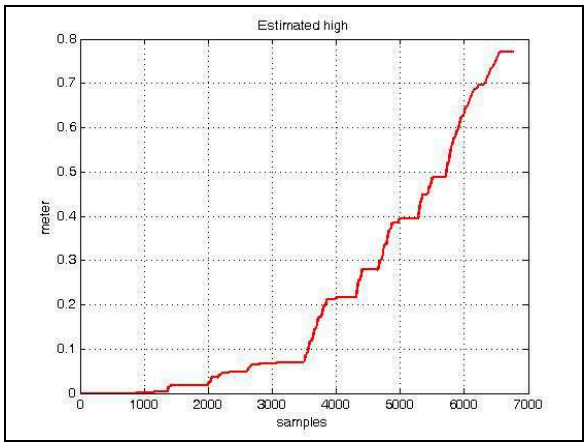



Figure 59. Height reconstruction test result

Whereas the peak, as deduced from the following charts, is about 0.6 m of maximum height of 3 meters (standard height of a room), total real height, it follows a mistake as a percentage ranging from 0 to 20%. So the use of high-pressure calibration curve and the Kalman filter leads to a substantial reduction of the error reports using standard barometric views which can be as high as 70%.

## 3.3 TRAJECTORIES RECONSTRUCTION

The planar trajectory reconstruction is based on a series of steps performed in a precise order. The first two phases, consist in data filtering and correction off-set: accelerations on plan OXY and angular velocities are in input to the MatLAB functions like FIR or SMOOTH types that clean up the signal from high frequency harmonic components. Then using statistical operators as the standard and average deviation it is possible to identify the off-set when the system is stopped and then correct it in the dynamic phase. To retrieve the angular position and displacement the Crank-Nicholson or trapezoidal integration is used. Finally, by adopting a mathematical model based on roto-translations, we calculate the absolute path of the system (car, robot, person) in motion. As with all digital filters, including the FIR filter (Finite Impulse Response) is a discrete-time filter, ie linear and stationary monodimensional systems, that taking an input sequence x[n] generates an output one y[n]. A discrete-time invariant linear system has a finite impulse response, when there is a M > 0, such that h(n) = 0 for n>M. The input-output relationship of the FIR filter is:

$$y(n) = \sum_{k=0}^{M-1} h(k)x(n-k)$$

Turning to the Z transform and applying the properties of the time translation, we obtain:

$$Y(z) = H(z)X(z)$$

Where

$$H(z) = \sum_{k=0}^{M-1} h(k) z^{-k} \Rightarrow H(z) = \frac{P(z)}{z^M}$$

is the impulse response of FIR filter, P(z) is a polynomial of degree M with real coefficients, and X(z), Y(z) are respectively the z-transform of x n] and y n x[n] and y[n]. The filter is uniquely determined by the zeros of P(z), in other words the solution of the equation P(z) = 0. Basic characteristics of these filters are stability and phase linearity: the first property derives from the fact that H(z) is a polynomial in 1/z, thus it has a pole inside the circumference with unitary radius, while the second is based on the following relation $\varphi(\omega) = \alpha\omega + \beta$ easily demonstrable. In MatLAB *fir1* function allows to synthesize the weights of linear phase FIR filters (symmetric weights) low-pass, high-pass, band-pass and band-stop types via the method of the windows. This technique consists in set at zero the response samples to the impulse of infinite duration hd(n) of the desired transfer function H(z) outside of a temporal window of size N, in the hope that the approximation is as good as larger is the size N (number of nonzero samples) of the window. Many are the time windows can be used, we take into consideration the rectangular one:

$$rett_N = \begin{cases} 1 & |n| \le \frac{N-1}{2} \\ 0 & altrimenti \end{cases}$$

therefore we may write

$$h_N(n) = rett_N(n) h_d(n);$$

in other words, $h_N(n)$ is obtained by multiplying the filter impulse we wish to approximate for the rectangular window $rett_N(n)$ of finite length. Let's study now how the filter with $h_N(n)$ reply approximates the filter

with $h_d(n)$ reply at N variation. For sake of ease we assume that we wish to approximate a low-pass filter characterized by the following frequency response:

$$rett_N = \begin{cases} 1 & |\omega| \le 1 \\ 0 & 1 < |\omega| < \pi \end{cases};$$

We denote by $H_N(j\omega)$ the reply in frequence of $h_N(n)$ filter. Two phenomena may be observed:

1. $H_N(j\omega)$ has a nonzero transition band, whose width converges to zero when N diverges;

2. $H_N(j\omega)$ presents fluctuations in both pass-band and band-gap, whose maximum width is about 0.2, regardless of the temporal dimension N of the window.

However, the above phenomena can be resolved, although not completely, choosing time windows $w_N(n)$ diverse from that rectangular. Characteristic of these windows is that they show discontinuities different from zero in the range $-\frac{(N-1)}{2} \le n \le \frac{(N-1)}{2}$.

The relationship between the desired filter and the FIR filter obtained with the window $w_N(n)$ is then:

$$h_N(n) = w_N(n) h_d(n);$$

Examples of commonly used windows as well as the rectangular window are listed below.

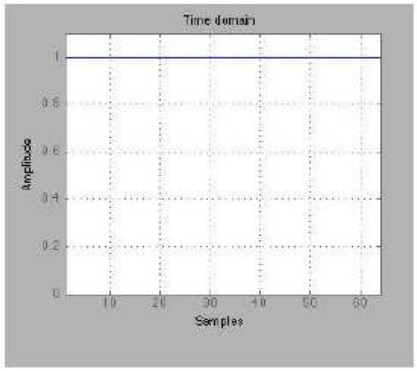



Figure 60. Rectangular window.

Triangular Window (or Bartlett):

$$w_N(n) = \begin{cases} 1 + \dfrac{2n}{N} & -\dfrac{(N-1)}{2} \le n \le 0 \\ 1 - \dfrac{2n}{N} & 0 \le n \le \dfrac{(N-1)}{2} \end{cases};$$

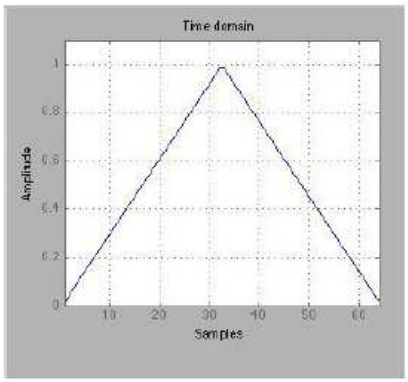



Figure 61. Triangular window.

Hamming Window:

$$w_N(n) = \frac{1}{2}\left[1 + \cos\frac{2\pi n}{N}\right] \qquad |n| \le \frac{(N-1)}{2};$$

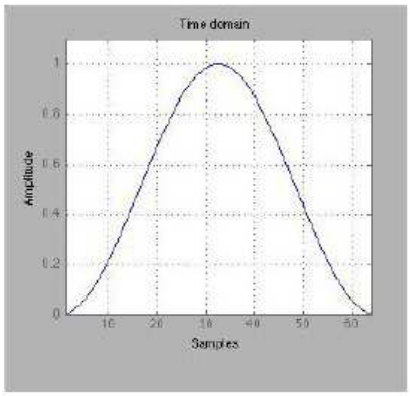



Figure 62. Hamming window.

In the present case, the designed FIR filter is based on the Hamming temporal window with N=15 weights, the MatLAB function requires the sampling frequency of the sensor (50 Hz) and the transition frequency equal to 0.1Hz. At this point we can filter the accelerations organized in a buffer updated at each iteration, as described at the previous paragraph.

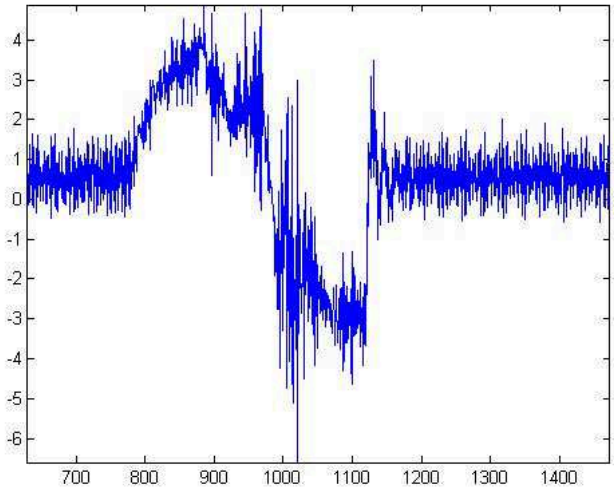



Figure 63. Unfiltered accelerations.

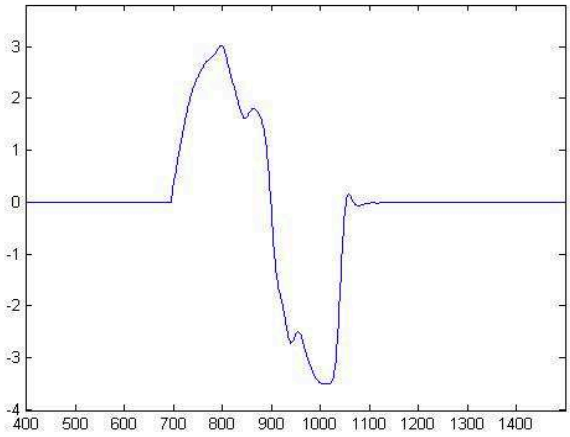



Figure 64. Filtered Accelerations.

The filtering of the angular speeds was instead carried out by the *smoothing* function. In reference to sample distribution, which in this case are the angular speeds, quite often it happens that they represent the phenomenon observed with a variable percentage of error during the process of measurement. It is possible to use *smoothing* algorithms to generate a model that can interpolate the data set by minimizing the error of the average iterated by samples of a temporal window

The MatLAB implements several as moving average, linear fitting, the quadratic fitting or Savitzky-Golay technique; all modulating in the degree of interpolation. In our case we used a *smooth* squared function with degree of interpolation equal to 0.3 on a a buffer of 200 samples.

The following are two figures related respectively to the interpolated and not interpolated angular speeds.

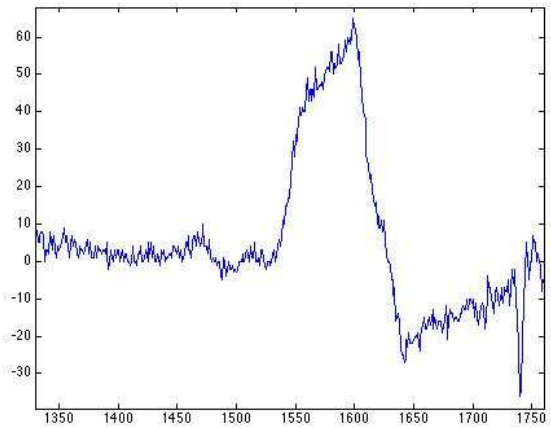



Figure 65. Angular speeds not subjected to smoothing.

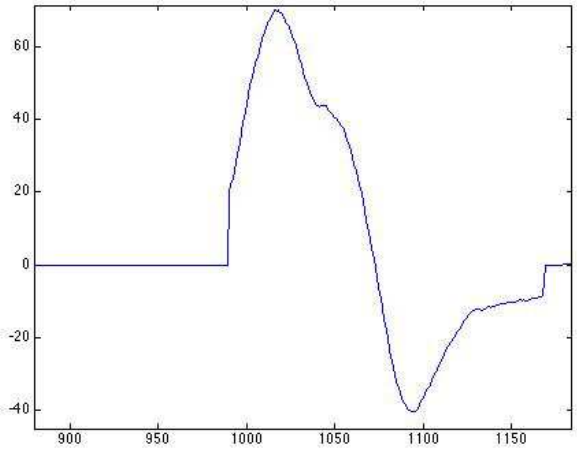



Figure 66. Angular speeds subjected to smoothing.

Before analyzing the mathematical model for the construction of a trajectory, the statistical parameters will be introduced: average and standard deviation. These ones, calculated on a number of samples and compared with appropriately chosen thresholds, allow to correct the offset due to thermal drift and to distinguish the phase of motion from the phase of quiet. First, however, we must resume some basic concepts of

descriptive statistics. In statistics data that are used are measures from which we want to extract certain information. These data can be classified into quantitative data and qualitative or numeric or nonnumeric. The numerical data can be divided into continuous and discrete. The arithmetic average is the first statistical parameter that is taken into account, if the data set taken into consideration consists of N elements:

$$\{x_1, x_2, ....., x_N\}$$

we define as the arithmetic average the following expression

$$\bar{x} = \frac{1}{N}\sum x_i;$$

Similarly it is possible to define the average relative to a general function f(x):

$$\bar{f} = \frac{1}{N}\sum f(x_i);$$

The measure of the dispersion of data is guaranteed by another statistical parameter: the variance, depending on the arithmetic average previously calculated:

$$V(x) = \frac{1}{N}\sum_i (x_i - \bar{x})^2;$$

for any function f the previous becomes:

$$V(f) = \frac{1}{N}\sum_i (f(x_i) - \bar{f})^2;$$

The V(x) previously calculated can be manipulated to arrive at a simpler form:

$$V(x) = \frac{1}{N}\sum_i (x_i - \bar{x})^2 = \frac{1}{N}\sum_i (x_i^2 - 2x_i\bar{x} + \bar{x}^2);$$

From the formula concerning the variance it is possible to estimate the standard deviation calculated as follows:

$$\sigma = \sqrt{V(x)}$$
$$\sigma = \sqrt{\bar{x^2} - \bar{x}^2}$$
$$\sigma = \sqrt{\frac{1}{N}\sum_i (x_i - \bar{x})^2}$$

When the system is stopped actually the acceleration along the direction of travel is different from zero.

This as mentioned above could be due both to thermal drift and from the imperfect alignment of the axes related to the inertial platform with respect to the absolute reference system. To calculate the offset when the system is stopped just apply the MatLAB function *std* to a buffer of size N, if the output value is below a certain threshold, the system is stationary, so from the average acceleration it is possible to calculate the off-set. If instead, the standard deviation exceeds the numerical threshold, the system is in motion so we subtract to the accelerations the offset identified in quiet phase using the following formula:

$$acc\_motion(k) = acc\_motion(k) - (off\_set) \qquad off\_set > 0$$
$$acc\_motion(k) = acc\_motion(k) + (off\_set) \qquad off\_set < 0$$

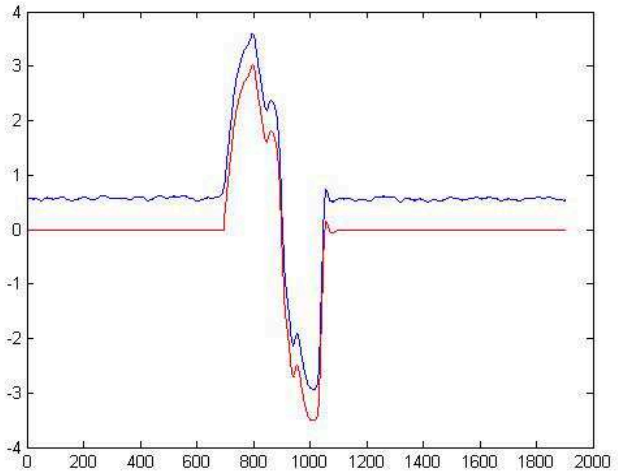



Figure 67. Correction off-set.

The integration method for acceleration and angular velocities (based in part on the method of the parabolic segment of Archimedes) is to be considered in any interval $[x_i, x_{i+1}]$ of width $h$ the trapezoid rectangle which has as its vertices the points of the function in the extremes. Each of these trapezoids have the same height, ie $h$ The area of the trapezoid k-th is given by:

$$T_k = \frac{y_{i+1} + y_i}{2} \cdot h;$$

(the area can be considered with a negative sign if the function is not positive).

The sum of the $N$ trapezoids $T_K$ allows an approximation of the defined integral, ie

$$\int_a^b f(x)dx \approx \frac{b-a}{2N}(y_0 + 2y_1 + .... + 2y_{N-1} + y_N);$$

This formula is called the "trapezoidal quadrature" formula or "Crank-Nicholson." It can be shown that if $f$ is twice differentiable with continuous second derivative, the error of the trapezoidal quadrature formula is given by:

$$\frac{1}{12}h^3\left\|f^{"}\right\|;$$

Here is the sequence of integrations that allows switching from acceleration to displacement of the system in motion. For completeness it was first reported the time-continuous integration form and then the time-discrete:

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

In the discrete-time domain the previous becomes:

$$v_0 + \int_{t0}^{tN} a(t)dt = v_0 + \sum_{k=0}^{N} T \cdot \frac{a(k) + a(k+1)}{2};$$

By repeating the same reasoning it is possible to obtain the relative position, in fact:

$$p_0 + \int_{t_0}^{t} v(t)dt = p(t);$$

While in the discrete-time the resolutive formula becomes:

$$p_0 + \int_{t0}^{tN} p(t)dt = p_0 + \sum_{k=0}^{N} T \cdot \frac{v(k) + v(k+1)}{2};$$

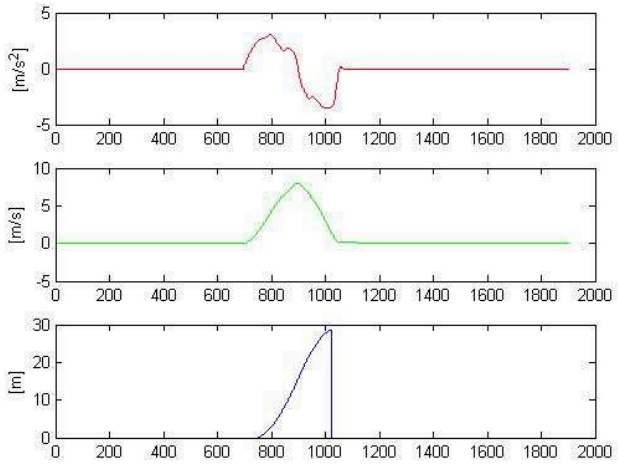



Figure 68. Acceleration, speed, position.

The same reasoning applies to the angular speeds, through their integration it is possible to derive the relative angle, then coordinate transformations will determine the absolute angle.

$$\theta_0 + \int_{t_0}^{t} \omega(t)dt = \theta(t);$$

$$\theta_0 + \int_{t0}^{tN} \omega(t)dt = \theta_0 + \sum_{k=0}^{N} T \cdot \frac{\omega(k) + \omega(k+1)}{2};$$

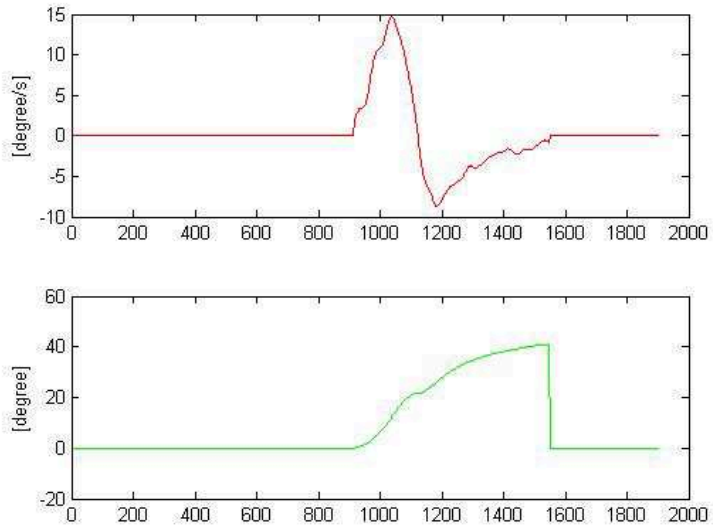



Figure 69. Angular speed, angle.

To rotate a reference system (so leaving it orthonormal), it is sufficient to fix the angle axis x' must have with axis x, that rotation is automatically determined. Let's suppose to wish to rotate the reference system in anticlockwise sense for a $\alpha$ angle.

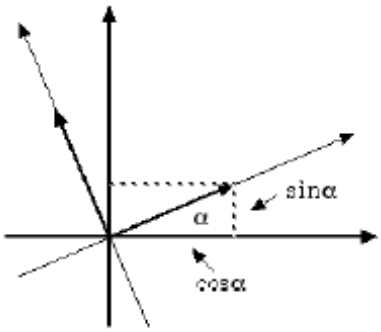



Figure 70. Axis rotation by an angle $\alpha$.

So the first unit vector of the new base must be:

$$u_1 = \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix};$$

While the second, having to be orthogonal to it and always as a rule 1, will be:

$$u_2 = \begin{pmatrix} -\sin\alpha \\ \cos\alpha \end{pmatrix};$$

So at this point the change of reference is given by:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix};$$

and hence the change of reversed reference will be:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & -\cos\alpha \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix};$$

What has been seen for the reference systems in the plan can be repeated in space. And it is not difficult to imagine that, though at the expense of much heavier notations, by setting a new base on the space, we can build the matrix of change of coordinates that will link the “old” (x,y,z) coordinates with the "new" (x',y',z'). Even this one will be achieved by taking the coordinates of the vectors for columns that we choose as a new base and it will be a 3X3 matrix. There are other ways to change the reference system, beyond rotation. If a point moves parallel to the x axis and from the point $P_0$ = (1000, 1), to describe its movement it would be more convenient to have a reference system where the origin of the new axis coincides with the point $P_0$, so that the new coordinates $P_0$ become (0,0).

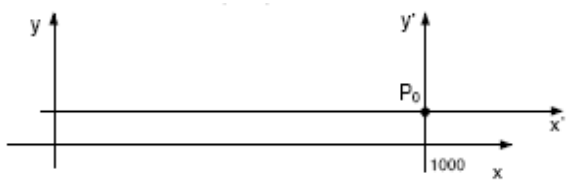



Figure 71. Translation.

Given, then, a point $P_0 = (x_0, y_0)$ we would translate axes (x and y) so that the new reference ox'y ' had its origin in $P_0$; the connection between the coordinates x and y, and the coordinates x' and y' will be as follows:

$$\begin{cases} x' = x - x_0 \\ y' = y - y_0 \end{cases};$$

We can elementarily deduce it from the following figure:

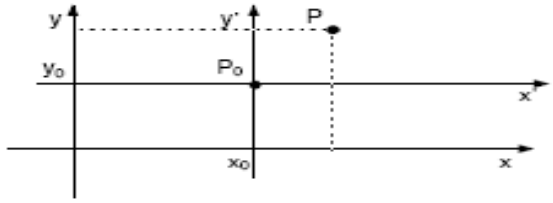



Figure 72. Example of translation.

Once retrieved displacement and relative angle for integration, we can update the position using the following transformation, which is a roto-translation respect to the absolute reference system.

$$\begin{cases} X = X_0 + x\cos\theta + ysen\theta \\ Y = Y_0 - x\cos\theta + ysen\theta \end{cases};$$

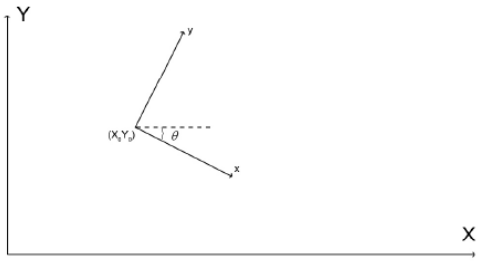



Figure 73. Reference system.

X and Y are the coordinates in the reference system concerning the sampling instant (i-1); x and y are the coordinates of the same system at the instant (i); and $\theta$ the angle between the two reference systems. The position respect to the absolute reference system is updated every change of direction; in practice the algorithm calculates the related angle that once ended rotation (when the gyroscopes have output null) is added to the absolute angle, as shown clearly in the following image:

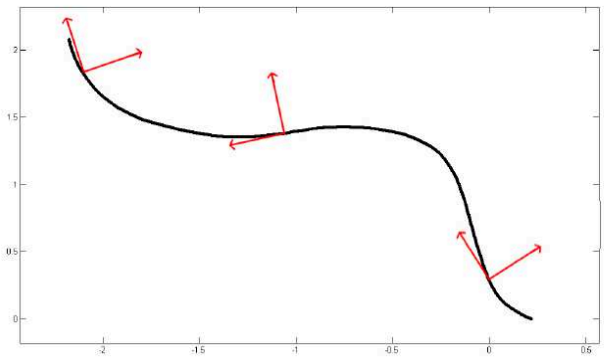


Figure 74. Reconstruction of the trajectory in the absolute reference system.

The following figures show the reconstruction of trajectories in the MatLAB environment, developed by the algorithm in post-processing:

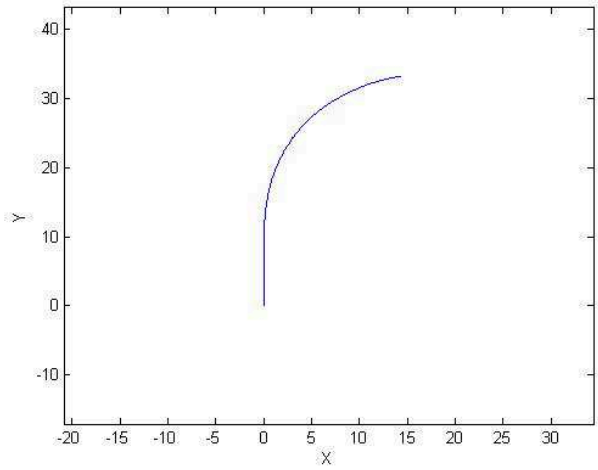



Figure 75. Reconstructed trajectory.

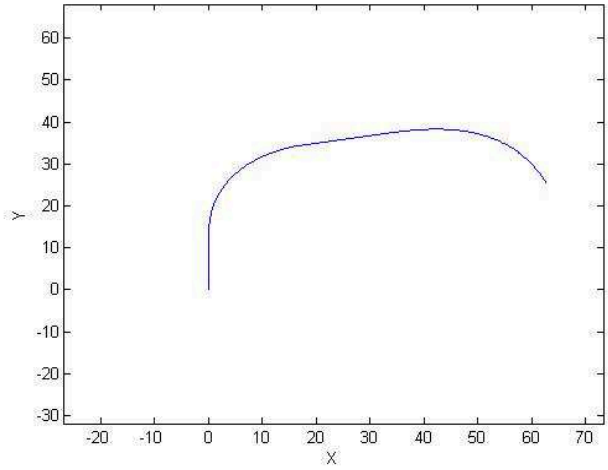



Figure 76. Reconstructed trajectory

We discussed the reconstruction of planar and altimetric trajectories, using the Kalman filter in the first case, and numerical integration with a coordinate transformation in the second. But these are off-line analyses, for a 3D and in real-time reconstruction it has been then revised the data organization in buffer and statistical parameters have been introduced as

average, mode and standard deviation to let distinguish three-dimensional motion from the planar motion.

The distinction between planar motion and altimetric motion has been developed in two different phases: in the first the height trend estimated by the Kalman filter was considered as a discriminating factor; in the second, on the other hand, harmonic analyses and statistics acceleration along the Z axis were carried out. The vertical displacement is also calculated with 6% of errors, but the assess of the filter can not be used as a discriminating parameter between two-dimensional and three-dimensional motions, due to the response of the MEMS device used as a pressure sensor. In fact, this presents a suspended membrane with thin legs, deforming themselves according to the gas molecules that go depositing; consequently, the higher is the density, the greater will be the decline of legs.

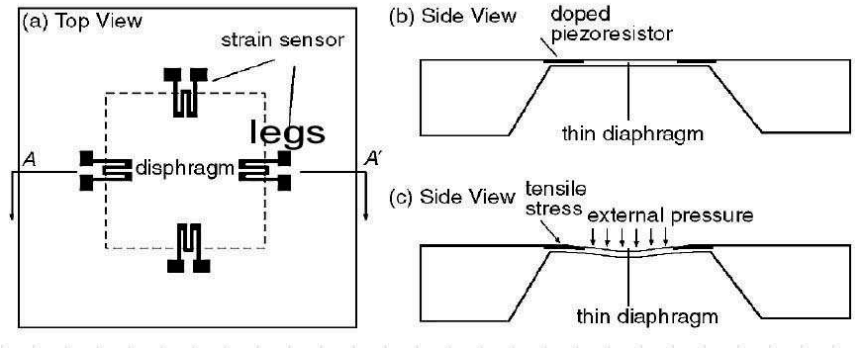



Figure 77. MEMS Pressure sensor.

That applies obviously for the reverse operation too, in fact, if the air gets thinner, the membrane and the legs go back to rest position.

The "sensor" system is a mechanical system, so if it is subjected to a stress, which in this case is the deposition or not of air molecules on the membrane, it responds with a certain rise time, overdisplacement, and

settling time. Where rise time means the time for the system to vary from 10% to 90% of the scheme value. While settling time and overdisplacement are respectively the interval after which the response settles down, and the oscillations around the condition value.

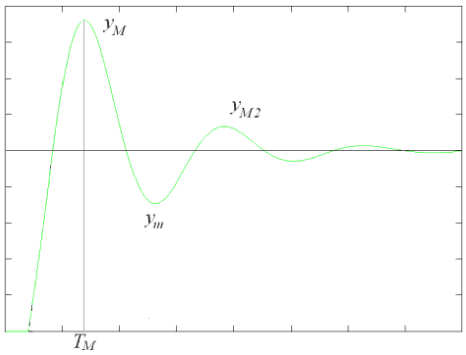



Figure 78. System overdisplacement and settling time

Because of these last two not filterable parameters, as the case of gyroscopes and accelerometers, the height estimated by the Kalman filter can not be used as discriminating factor between planar motion and three-dimensional motion. In fact, if the system in motion, once reached a certain quota, moves up on planar motion, the pressure sensor modulates the height estimated due to overdisplacement.

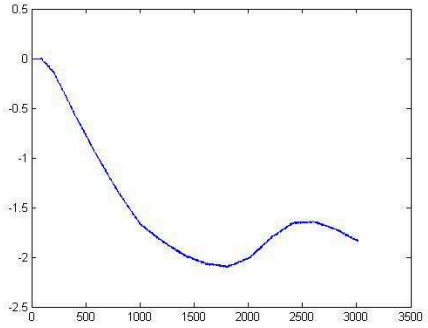



Figure 79. Pressure sensor overdisplacement.

However, from off-line analyses of the accelerations along the z axis, we can see different amplitudes according to the motion described by the system: planar or orthogonal to the OXY plan. The accelerations - as shown in the following figures – are, therefore, the only discriminating factor between the 2D motion and 3D motion. It would be sufficient to implement appropriate mathematical tools and numerical thresholds to determine if the system (car, robot, person) drives along a climb or descent route.

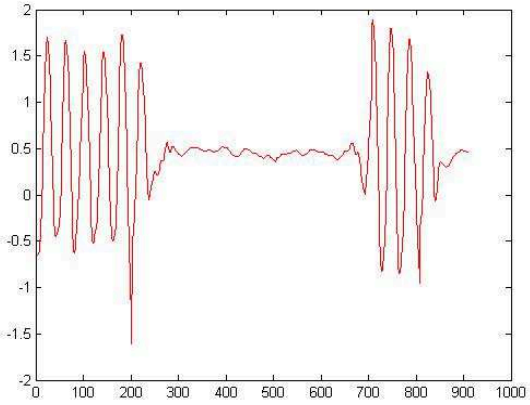



Figure 80. Accelerations along Z axis in the planar case.

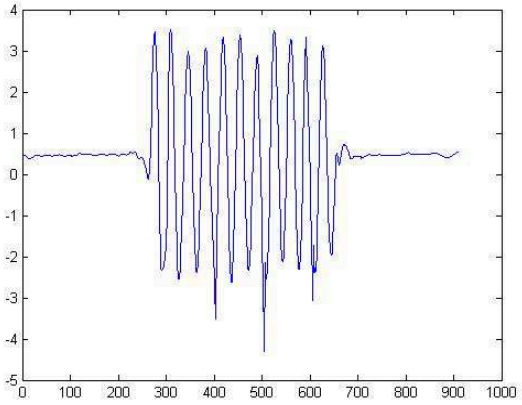



Figure 81. Accelerations along Z axis in navigating a ramp.

In a first stage we had decided to perform the spectral analysis of accelerations using the form of the FFT, as a discriminating 2D/3D element, as well the standard deviation, which is used to identify the stages of movement or quiet along the Z axis.

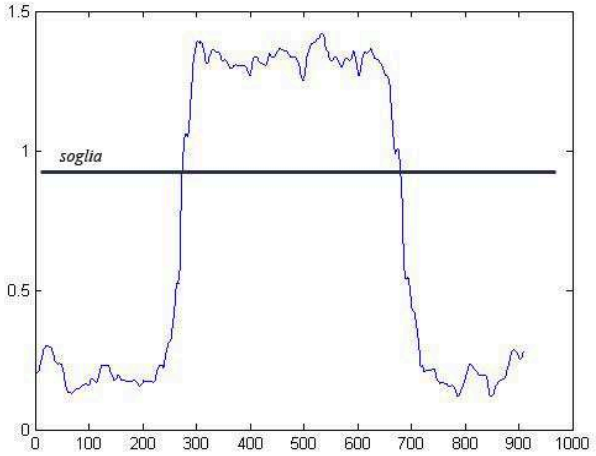



Figure 81. Standard deviation of accelerations along Z axis

The algorithm for the calculation of the FFT is a function which implements the form of the following formula:

$$F_d(x_n) = \sum_{k=0}^{N-1} x_k e^{-j\frac{2\pi}{N}kq} \qquad per \qquad q = 1....N-1$$

From the following graph which relates the form of the FFT, it is not possible to identify a discriminating threshold, this because the value of the form even during a ramp-distance falls below the threshold. Which is why we have decided to switch to statistical analyses as average or mode, the latter understood as the parameter indicating the sample that is repeated more frequently in a data set.

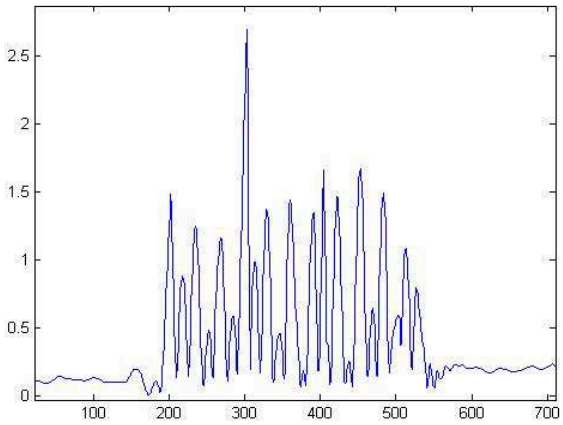



Figure 82. Module of the FFT related to accelerations.

The average is best suited to distinguish the plan motion from climbs; in fact, by the following figure it is deduced that accelerations along the positive z-axis, expressed in blue have an average value greater than those in red on the plan motion; a Boolean variable keeps track of this behavior, allowing the comparison of the average with the numerical threshold once and reducing the computation time of the algorithm.

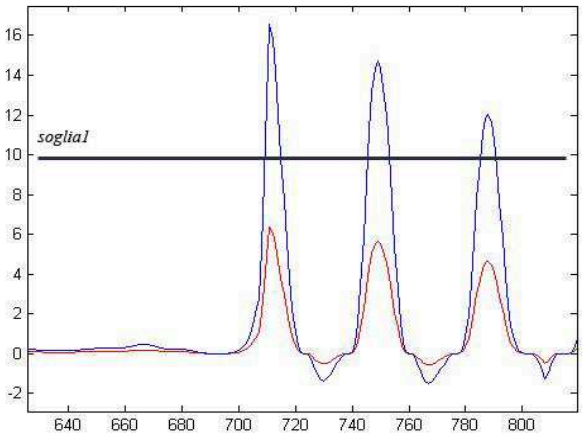



Figure 83. Average of accelerations

The mode is more suitable to discriminate the planar motion from descent phases, following the same reasoning as made for climbs. In fact, following analyses made in post-processing on accelerations during the descent phase of a ramp, it has been possible to deduce that mode is a tougher discriminating method than average; because at the end of this phase of descent we find negative peaks due to soil binding reaction.
The mode by identifying samples recurring more frequently, better identifies the prevalence of negative accelerations.

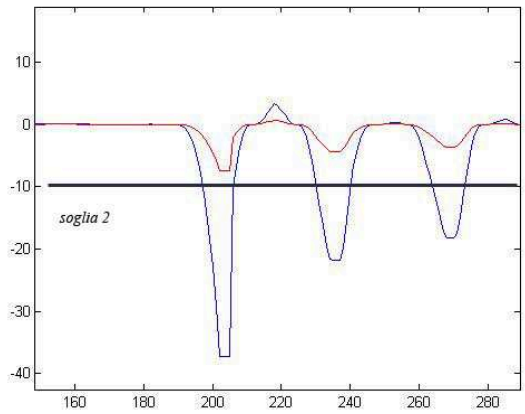



Figure 84.Mode of accelerations along Z axis.

By considering the discriminating statistical parameters, the mathematical models for the OXY and altitude reconstruction, it should be possible to get 3D graphics both off-line and in real-time.

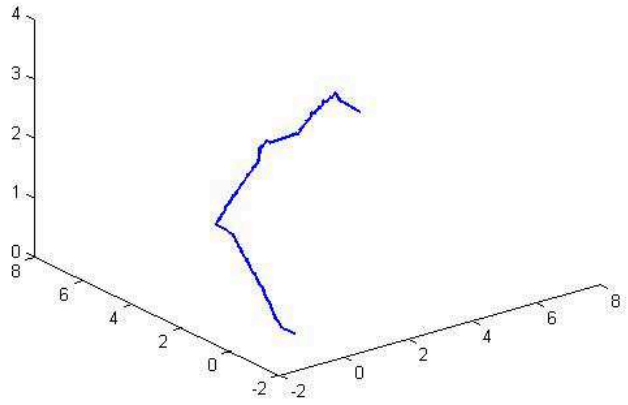



Figure 85.The figure shows: three-dimensional reconstruction of ramps uphill arranged orthogonally to each other.

## 3.4 RECONSTRUCTION OF PLANAR TRAJECTORIES PER VEHICLE: TEST

The algorithm for the reconstruction of trajectories related to a vehicle requires, as first steps, to develop the processing of accelerations and angular speeds through FIR filters and numerical integration - widely discussed previously. For this application it has been necessary the introduction of two Kalman filters: the first one to compensate the drift (thermal and integral) concerning movements; the second is applied to the angular variations. Along a closed path it is possible to initialize the matrices of variances in order to reduce the error between the actual path points and the reconstructed path.

The system consists of an inertial platform ST-iNEMO appropriately placed inside the car and from the MatLAB development environment, which implements the algorithm in real-time and through appropriate GUI allows the user to monitoring the reconstructed path.

Figure 86. Fixing the inertial platform.

To evaluate the error between actual trajectory and reconstructed trajectory by the algorithm, it has been required putting along the path some cones used as metric reference (markers). In this case we drove an oval path constituted of two 50 meters long straight roads, with references put at the beginning and at the end, and two curves with average radius of 15 meters and references put at beginning, middle and end curve. The purpose of the experience is stopping the car and then restarting again at each marker, to be able to evaluate at each step the reconstruction error. The following images show the planimetric map and details of the path made in an appropriate open space.

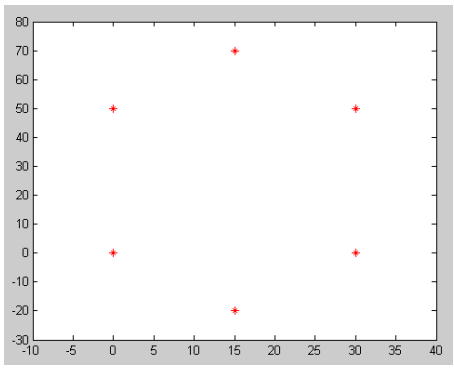



Figure 87. Planimetric map.

Figure 88. Particular of the long straight road.

The reconstruction of the trajectory obtained by double integration of acceleration is affected to error due to thermal and numerical drift As in the case of large displacements, as it happens just for the vehicle, the error is of several meters; to reduce it, it has been considered appropriate to implement a Kalman filter so as to give more weight to the *optimus observer* implemented in *predict* than to the only sensory response. Consider the following mathematical model:

$$\begin{bmatrix} y(k+1) \\ v\_y(k+1) \\ a\_y(k+1) \end{bmatrix} = \begin{bmatrix} 1 & T & 0 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} y(k) \\ v\_y(k) \\ a\_y(k) \end{bmatrix} + \begin{bmatrix} \varepsilon_y(k) \\ \varepsilon_{v\_y}(k) \\ \varepsilon_{a\_y}(k) \end{bmatrix};$$

$$\begin{bmatrix} \bar{y}(k+1) \\ \overline{v\_y}(k+1) \\ \overline{a\_y}(k+1) \end{bmatrix} = \begin{bmatrix} y(k+1) \\ v\_y(k+1) \\ a\_y(k+1) \end{bmatrix} + \begin{bmatrix} \psi_y(k+1) \\ \psi_{v\_y}(k+1) \\ \psi_{a\_y}(k+1) \end{bmatrix};$$

which in matrix compact form becomes:

$$Y(k+1) = A \cdot Y(k) + \mathrm{E};$$
$$\bar{Y}(k+1) = C \cdot Y(k) + \Psi;$$

In $y(k+1), v\_y(k+1), a\_y(k+1)$ they are the variables of the mathematical model, $\bar{y}(k+1), \overline{v\_y}(k+1), \overline{a\_y}(k+1)$ correspond respectively to the measured acceleration and to the speed and displacements obtained by acquired accelerations integration. Model and measure error - indicated through variables $\varepsilon$ and $\psi$ - allow to draw variances for Q and R matrices initialization.

$$Q = \begin{bmatrix} \varepsilon_y^2 & 0 & 0 \\ 0 & \varepsilon_{v\_y}^2 & 0 \\ 0 & 0 & \varepsilon_{a\_y}^2 \end{bmatrix};$$

$$R = \begin{bmatrix} \psi_y^2 & 0 & 0 \\ 0 & \psi_{v\_y}^2 & 0 \\ 0 & 0 & \psi_{a\_y}^2 \end{bmatrix};$$

The P matrix, solution of Riccati equation, is initialized as identity matrix 3X3:

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix};$$

The values of the main diagonal of the matrix Q are initialized close to zero, this to give more weight to the filter observer. In fact, increasing them to the threshold 1, the merely sensory information prevails. Matrix R values can be got from a simple statistical analysis of data acquired from the accelerometer under resting conditions.

$$R\_iniziale = \begin{bmatrix} 10^{-3} & 0 & 0' \\ 0 & 2.59 \cdot 10^{-4} & 0 \\ 0 & 0 & 6.85 \cdot 10^{-9} \end{bmatrix};$$

The calibration of the matrix R has been made in post-processing, processing the data acquired after a straight path of known length. The reconstruction is as more true as the higher is position and speed variance; the best result is obtained for $R(1,1) = 10^2$ and $R(2,2) = 10$.

$$R\_finale = \begin{bmatrix} 10^{2} & 0 & 0' \\ 0 & 10 & 0 \\ 0 & 0 & 6.85 \cdot 10^{-4} \end{bmatrix};$$

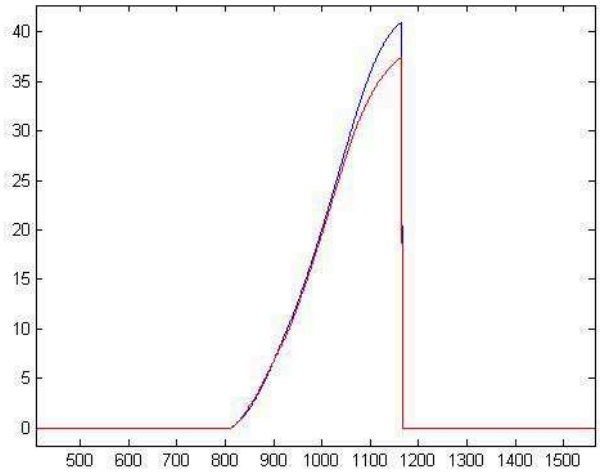



Figure 89. Comparison between the filtered (red) and unfiltered shifting.

The following figure shows the pattern of movements over time; in red the shift estimated by the Kalman filter is indicated, and in blue shift obtained only by integration of the acquired accelerations. It is clear that the estimate of the filter is more faithful to the straight line covered.

The considerations made for the related displacements are taken up for the corners, too. The thermal drift also concerns gyroscopic information, as numerical shift regards the angles obtained by angular speeds integration. Let's consider the following mathematical model obtained from the equations of kinematics:

$$\begin{bmatrix} \theta(k+1) \\ \omega(k+1) \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \theta(k) \\ \omega(k) \end{bmatrix} + \begin{bmatrix} \varepsilon_\theta(k) \\ \varepsilon_\omega(k) \end{bmatrix};$$

$$\begin{bmatrix} \overline{\theta}(k+1) \\ \overline{\omega}(k+1) \end{bmatrix} = \begin{bmatrix} \theta(k+1) \\ \omega(k+1) \end{bmatrix} + \begin{bmatrix} \psi_\theta(k+1) \\ \psi_\omega(k+1) \end{bmatrix};$$

in matrix compact form begins:

$$\Theta(k+1) = A \cdot \Theta(k) + \mathrm{E}$$
$$\overline{\Theta}(k+1) = C \cdot \Theta(k) + \Psi$$

$\theta(k+1), \omega(k+1)$ are respectively the angular position and angular speed in output to the filter observer; $\overline{\theta}(k+1), \overline{\omega}(k+1)$ are measures affected by noise: the angular velocity is acquired from the gyroscope, the angle - intended as a measure - is obtained by integration of $\overline{\omega}$.

Considering model errors and measurement errors, it is possible to initialize matrices Q and R, respectively; the matrix P is instead initialized as an identity matrix2X2:

$$Q = \begin{bmatrix} \varepsilon_\theta^2 & 0 \\ 0 & \varepsilon_\omega^2 \end{bmatrix};$$

$$R = \begin{bmatrix} \psi_\theta^2 & 0 \\ 0 & \psi_\omega^2 \end{bmatrix};$$

$$P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix};$$

The variances in the model present in matrix Q are initially placed at values close to zero; while the variances of matrix R are calculated through a statistical analysis of data supplied by the gyroscope under resting conditions. By leaving unchanged the variance of the angular speeds and modulating instead the one related to the corners only, it is possible modifying the response of the Kalman filter, this until orientation at the beginning and end of a closed *reconstructed trajectory* (oval in this case) matches.

In particular, increasing the variance, the output of the filter is more faithful to the sensory information and the drift prevails, vice versa, the angle is amplified and the trajectory is inclined to close itself.

$$R\_iniziale = \begin{bmatrix} 6.42 \cdot 10^{-2} & 0 \\ 0 & 8.68 \cdot 10^{-2} \end{bmatrix};$$

$$R\_finale = \begin{bmatrix} 6.42 \cdot 10^{-4} & 0 \\ 0 & 8.68 \cdot 10^{-8} \end{bmatrix};$$

The following figure shows a comparison between the angular displacement obtained by simple integration of sensory information (in blue) and the Kalman filter output (in red).

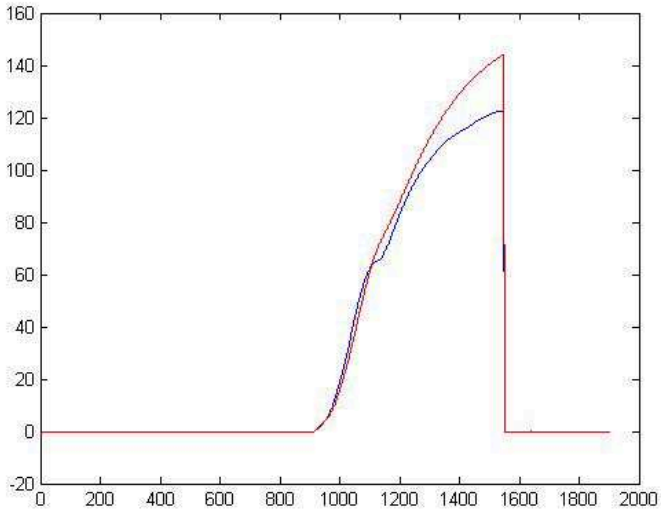



Figure 90. Comparison between angular filtered displacements (red) and unfiltered one (blue).

The reconstruction of the trajectory with respect to the absolute reference system is also in this case through rotary translations; ie updating the absolute angle, at each change of direction, as clearly shown in the following figure. The reference systems in red express the point where there is the absolute position update:

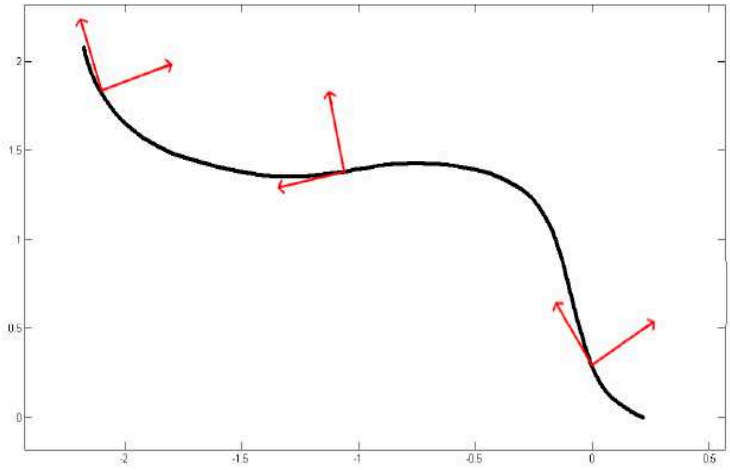



Figure 91. Reconstruction of the trajectory through semi relative template.

The use of the Kalman filter for the realization of sensor fusion is a widely used technique and it produces excellent results in terms of accuracy. It is often used for the integration of GPS measurements with those made by the inertial navigation system, although in this case there is only the platform and it is assumed that measures are: accelerations, speed and position all obtained by integration.

The same structure of the filter and its reliability, regardless to the type of sensors on which to perform the sensor fusion, is highly dependent on the model representing the system and on the error covariance matrices associated with it. To evaluate the effectiveness of the Kalman filter to compensate the effects of drift, the trajectories reconstructed by the

algorithm during the journey of the oval path have been reported. It is clear that the error between the filtered trajectory and the markers of the metric map is consistently below 5 meters, about 2% of 200 meters walked, as opposed to the second case in which the drift effect leads to an exponential increase of the error that reaches up to 12%.

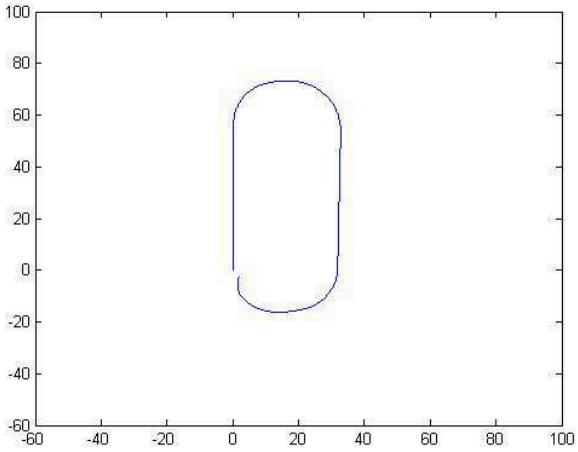



Figure 92. Reconstruction of the trajectory through filter.

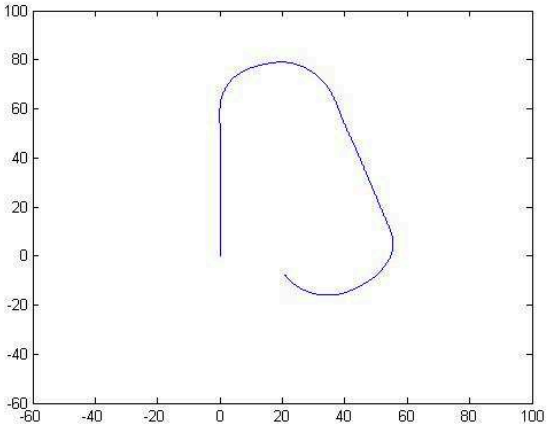



Figure 93. Reconstruction of the trajectory without Kalman filter.

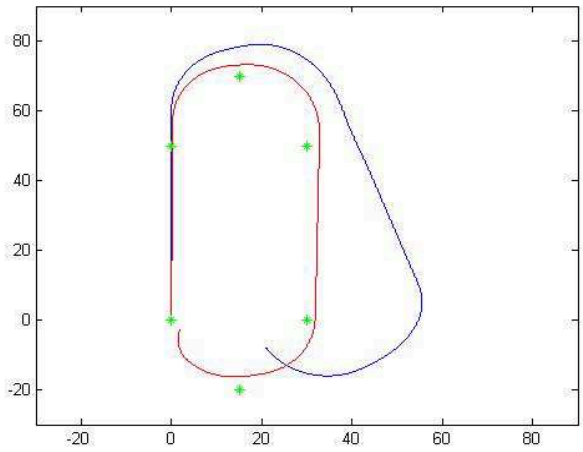



Figure 94. Comparison between filtered and not filtered trajectories on metric map.

## 3.5 BUMP DETECTION

Bump detection algorithm does not work with interpolated data, but it needs a directly analysis of acceleration values. When robot knocks against an obstacle, the acceleration data are similar to figure95 (where green and blue lines are X and Y accelerometer data values respectively).

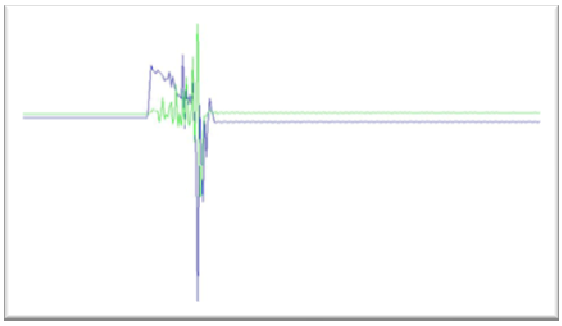


Figure 95. Acceleration values when a knock occurs on front left of the robot

If robot trajectory is known, it is simpler to fix bumping position. So the best performances are when both algorithms work together. The bump detection algorithm works with two thresholds: one to detect bumps, and another to establish bumping point. In fact, Newton's Third Law states that for every action there is an equal and opposite reaction. This means that any force exerted onto an object has a counterpart force that is exerted in the opposite direction back onto the first object. So obstacle exercises an equal and opposite snapshot force on the robot. X and Y axis acceleration data are used to determine the bumping zone; easily eight contact zones could be retrieved with an accelerometer. Practically eight couples of thresholds have to be set in accordance with the robot mass. Several acquisition campaigns have been performed with two different robots; in both the cases the eight thresholds have been easily found. The collision algorithm does not need a data interpolation. It is very fast because it carries out a snap analysis of acceleration. When there is a collision, there is a characteristic gait, as shown in 95 (the chart of acceleration in a collision moment).

Simply using an opportune threshold value serves to know if there is collision and its force. The collision algorithm is a separate task, because if you have the same reconstruction and collision having a delay, it is very negative for this application. This algorithm can be used for an interaction with the human environment. At the same time we can distinguish between a collision or a voluntary contact and interaction. Generally, the detection of a collision between the robot and an obstacle is performed by using a series of sensors around the robot itself. These sensors aren’t cheap but they have the drawback to wear and tear with continuous use. Here, the collision is detected by using an accelerometer

which does not wear out; the developed algorithm is capable of detecting up to eight positions of collision around the robot on a board level. The X and Y axis acceleration data are used to determine the collision zone; easily eight contact zones could be retrieved with one accelerometer. Practically eight couples of thresholds have to be set in accordance with the robot mass.

Several acquisition campaigns have been performed with two different robots; in both the cases the eight thresholds have been easily found. Actually the Z-axis of the accelerometer is not used, or it is used to discriminate surface only.

In order to demonstrate the algorithm performance, a GUI MatLAB interface has been developed. In this interface (see figure 96), the user can select the Virtual COM port to which the iNEMO board is connected. The Graphical Interface has two buttons: one to Connect/Disconnect iNEMO, and one to Start/Stop the mathematical algorithms. Moreover, a grid of eight bump point is depicted on the left of the interface, meanwhile the plotting area on the right tracks the movements of the robot. Any time the robot knocks against an obstacle, one of the eight squares of GUI becomes red, so it is possible to known the bump point. Any square works as an indicator, so after some seconds from the bump detection it becomes again green.

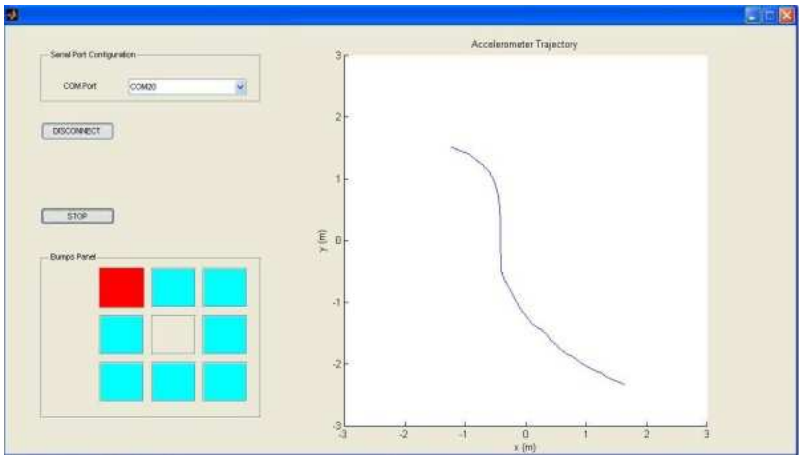



Figure 96. Picture of MatLAB GUI

# CONCLUSION

We have shown that thanks to the use of inertial sensors or MEMS (Micro Electro Mechanical System), we may develop different types of solutions. The objective was to develop methods and algorithms of sensor fusion by IMUs applied to service robotics, and in this dealing it has been shown what was done and the results obtained.

In the first chapter we have shown that the processing of data and the study of the state of art are fundamental for research. The solutions to the discussed problems allow various possibilities of discussion: the choice of data filtering, for example, takes place between quality, accuracy and speed of processing, where the preference of one at the expense of others is made on the basis of application and target.

Some of the solutions render less expensive different applications and can be matched to existing systems; this is what occurred both in the treatment of the bump, that in the reconstruction of planar trajectories for vehicles or even in the statistical analysis of the movements of the industrial robot and in all other situations described and shown.

The use of temperature and pressure sensors, alongside inertial classical sensors, such as accelerometers, gyroscopes and magnetometers, has allowed developing a three-dimensional localization in a simple way. The algorithm resolution allows recognizing nearly the rungs of the ladder, with an efficient localization that can be used also in all those cases where high precision is not needed but a qualitative localization.

This has been made possible thanks to sensors at lower cost and high level of integration.

The collaboration with STMicroelectronics has allowed to develop different approaches to the use of inertial platform of great impact for an industrial/commercial research which could have soon an outlet and a job.

The results shown in the second chapter expand the IMUs use also in the industrial environments, with particular attention to safety and process control, without the need for invasive techniques, but rather techniques that can be easily implemented regardless of the instrument tools used.

The final part, where we again talked about the use of inertial platforms with GPS, is the one that is deepened and on which we are still working and will work.

Of course it must be acknowledged that the inertial platform and MEMS sensors were certainly high-level and, above all, are so low-cost to make easy the prospect of an implementation of the algorithms developed in many different devices.

# ACKNOWLEDGMENTS

Fundamentally, it has been the help of students and tutors during these years, especially many thanks to the students of the degree course in engineering, graduates and undergraduates met who have given a proactive and collaborative support to the work presented both in the test phase that in the development one, and especially: S.C. Pernaci, M.M. De Benedetti, F. Sudano, S. Cirrone, G.D. Nicolosi. Thanks to Prof. Giacomo Spampinato of the Swedish university MÄLARDALENS School of Innovation Design and Technology, for his collaboration, support and caring advices.

Thanks to the tutor, Prof. Giovanni Muscato for the Catania University, afferent to the DIEEI (Dipartimento di Ingegneria Elettrica, Elettronica e Informatica) and Eng. Ph.D. Adriano Basile of STMicroelectronics jointly to the ART group (Automation, Robotic and Trasportation) of the IMS-System Lab of the Catania Site, who have hosted me in their offices and laboratories and provided the used material.

Finally thanks to the coordinator Prof. Luigi Fortuna and colleagues in the PhD program for their availability and friendship shown and to all the opportunities for fruitful and operative confrontation.

# BIBLIOGRAPHY


[1] Dorian Challoner "*Low Cost Robot Manipulator*", US PATENT N° 2003/0178964.

[2] M. Quigley, R. Brewer, S. P. Soudararaj, V. Pradeep, Q. Le, and A. Y. Ng "*Low-cost Accelerometers for Robotic Manipulator Perception*".

[3] S. Haddadin, A. Albu-Schäffer, A. De Luca, G. Hirzinger "*Collision detection and reaction: A contribution to safe physical human-robot interaction*".

[4] N. Abbate, A. Basile, C. Brigante, A. Faulisi "*Developmente of a MEMS based wearable motion capture system*".

[5] S. Miyazaki "*Long-term unrestrained measurement of stride length and walking velocity utilizing a piezoelectric gyroscope*".

[6] B. Najafi, K. Aminian, F. Loew, Y. Blanc, and P. Robert "*Measurement of stand-sit and sit-stand transitions using a miniature gyroscope and its application in fall risk evaluation in the elderly*".

[7] H. Dejnabadi, B. Jolles, and K. Aminian "A *new approach to accurate measurement of uniaxial joint angles based on a combination of accelerometers and gyroscopes*".

[8] D. Karantonis, M. Narayanan, M. Mathie, N. Lovell, and B. Celler "*Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring*" .

[9] S. Morris and J. Paradiso "*Shoe-integrated sensor system for wireless gait analysis and real-time feedback*".

[10] Sabatini, C. Martelloni, S. Scapellato, and F. Cavallo "*Assessment of walking features from foot inertial sensing*".

[11] D. Roetenberg, P. J. Slycke, P.H. Veltink "*Ambulatory Position and Orientation Tracking Fusing Magnetic and Inertial Sensing*".

[12] E. Charry, D. T.H. Lai, R. K. Begg, M. Palaniswami "*A study on band-pass filtering for calculating foot displacements from accelerometer and gyroscope sensors*".

[13] Y. K. Thong, M.S. Woolfson, J.A Crowe, B.R. Hayes-Gill and R.E Challis "*Dependence of inertial measurements of distance on accelerometer noise*".

[14] Y. K. Thong, M. Woolfson, J. Crowe, B. Hayes-Gill, and D. Jones "*Numerical double integration of acceleration measurement in noise*".

[15] E. Charry, D. T.H. Lai, R. K. Begg, M. Palaniswami "*Frequency analysis of inertial sensor data for measuring toe clearanc*e".

[16] X. Yun, E.R. Bachmann "*Design, Implementation, and Experimental Results of a Quaternion-Based Kalman Filter for Human Body Motion Tracking*".

[17] X. Yun, E. R. Bachmann, R. B. McGhee "*A Simplified Quaternion-Based Algorithm for Orientation Estimation From Earth Gravity and Magnetic Field Measurements*".
[18] B. Huyghe, J. Doutreloigne "*3D Orientation Tracking Based on Unscented Kalman Filtering of Accelerometer and Magnetometer Data*"..
[19] F. Camps, S. Harass, A. Monin "N*umerical Calibration for 3-axis Accelerometers and Magnetometers*"..
[20] Makoto Tanigawa, Henk Luinge, Linda Schipper, and Per Slycke "*Drift-Free Dynamic Height Sensor using MEMS IMU Aided by MEMS Pressure Sensor*", Xsens Technologies.
[21] Gaetano Iuculano, Domenico Mirri "*Misure elettroniche*".
[22] Corrado Di Natale "*Tecniche di calibrazione*" Sensors and MicroSystems team.
[23] P.Mazzoldi, M.Nigro, C.Voci "*Elementi di Fisica-Meccanica e Termodinamica*".
[24] Jose Guivant, Eduardo Nebot and Stephan Baiker "*Autonomous Navigation and Map building Using Laser Range Sensors in Outdoor Applications*".
[25] Titterton, D.H., and J. L. Weston, "*Strapdown inertial navigation technology*"..
[26] Shin, Eun-Hwan. "*Accuracy Improvement of low-cost INS/GPS for land applications*" PhD Thesis, University of Calgary.
[27] El-Sheimy, Naser "T*he Development of VISAT - A Mobile Survey System for GIS Applications*" PhD Thesis, University of Calgary.
[28] Jekeli Christopher "*Inertial Navigation Systems with Geodetic Applications"*.
[29] Cazzaniga Noemi E. "*Sviluppo e implementazione di algoritmi per la navigazione inerziale assistita*", Doctorate Thesis Politecnico di Milano.
[30] Rogers, Robert M. "*Applied mathematics in integrated navigation systems*".
[31] El-Sheimy, Naser, Sameh Nassar, and Ahoelmagd Noureldin "*Wavelet De-Noising for IMU Alignment*".
[32] El-Sheimy, Naser "*Inertial Techniques and INS/DGPS Integration*".
[33] Schimelevich, L., and R. Naor. "*New Approach to Coarse Alignment*".
[34] Farrell, Jay A. "*Aided Navigation, GPS with High Rate Sensors*".
[35] Mohinder S. Grewal, Lawrence R. Weill, Angus P. Andrews "*Global Position System Inertial Navigation And Integration*".
[36] J.F. Vasconcelos, P. Oliveiray, Carlos Silvestre "*Inertial Navigation System Aided by GPS and Selective Frequency Contents of Vector Measurements*".
[37] Mattia De Agostino "*Stima Degli Errori Nei Sensori Inerziali A Basso Costo*".

[38] D. H. Titterton J. L. Weston "*Strapdown Inertial Navigation Technology*".
[39] David Halliday,Robert Resnick, Jearl Walker "*Fondamenti Di Fisica*".
[40] Dieter Fox, Wolfram Burgardy, Frank Dellaert, Sebastian Thrun "*Monte Carlo Localization: Efficient Position Estimation for Mobile Robots*".
[41] Jose Guivant and Eduardo Nebot "*Solving Computational and Memory Requirements of Feature Based Simultaneous Localization and Map Building Algorithms*".
[42] Autar K. Kaw, Egwu E. Kalu, Duc Nguyen "*Numerical Methods With Applications*".
[43] D.S.G. Pollock "*Smoothing With Cubic Splines*".
[44] Christian H. Reinsch "*Smoothing by Spline Functions*".
[45] Carl De Boor "*A Practical Guide to Splines*".
[46] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery "*Numerical Recipes in C*".
[47] Raul Rojas "*The Kalman Filter*".
[48] D. Simon "*Kalman Filtering*".
[49] Greg Welch, Gary Bishop "*An Introduction to the Kalman Filter*".
[50] Autar K. Kaw, Egwu E. Kalu, Duc Nguyen "*Numerical Methods With Applications*".
[51] Alison K. Brown "T*est Results Of A Gps/Inertial Navigation System Using A Low Cost Mems Imu*", Ph.D Thesis.
[52] S.-J. Kim and R. A. Iltis, *"Performance Comparison of Particle and Extended Kalman Filter Algorithms for GPS C/A Code Tracking and Interference Rejection"*.
[53] Fredrik Gustafsson, Fredrik Gunnarsson, Niclas Bergman, Urban Forssell, Jonas Jansson, Rickard Karlsson, Per-Johan Nordlund "*Particle Filters for Positioning, Navigation and Tracking*" , Final version for IEEE Transactions on Signal Processing.
[54] Jonathan Ladd, Xinhua Qin "*A GPS Solution to Precision Approach and Landing (Even in the Presence of Anti-Spoofing)"*.
[55] Salah Sukkarieh, Eduardo M. Nebot and Hugh F. Durrant-Whyte, "*Achieving Integrity in an INS/GPS Navigation Loop for Autonomous Land Vehicle Applications*"
[56] Cheng Xi-jun, Cao Ke-jin, Xu Jiang-ning, Li Bao "*Analysis on Forgery Patterns for GPS Civil Spoofing Signals*"
[57] Oscar Pozzobon,, Luca Canzian, Matteo Danieletto, Andrea Dalla Chiara "*Anti-spoofing and open GNSS signal authentication with signal authentication sequences*"
[58] Isaac Skog and Peter H¨andel "*A Low-Cost Gps Aided Inertial Navigation System For Vehicle Applications*"
[59] Antonio Cavaleri, Beatrice Motella, Marco Pini and Maurizio Fantino "*Detection of Spoofed GPS Signals at Code and Carrier Tracking Level*"

[60] Robert Watson, Mark Petovello, Gérard Lachapelle, Richard Klukas, "*Impact of Oscillator Errors on IMU-Aided GPS Tracking Loop Performance*".
[61] T. S. Bruggemann, D. G. Greer, R. A. Walker "*GPS Fault Detection with IMU and Aircraft Dynamics*".
[62] Jon S. Warner, Roger G. Johnston "*GPS Spoofing Countermeasures*".
[63] Yafei Ren, Xizhen Ke "*Particle Filter Data Fusion Enhancements for MEMS-IMU/GPS*".
[64] Navid Nourani-Vatani Jonathan Roberts, Mandyam, V. Srinivasan "I*MU Aided 3D Visual Odometry for Car-Like Vehicles*".
[65] Hanching Grant Wang "*Low Authority Gps Aiding Of Navigation System For Anti-Spoofing*", US PATENT N°2009/0254278.
[66] Giuseppe Lunetta "*Elementi di Statistica descrittiva e Inferenza statistica*".
[67] Mattia De Agostino, "*Appunti di navigazione inerziale"*.
[68] M. Quigley, R. Brewer, S. P. Soundararaj, V. Pradeep, Q. Le ed A. Y. Ng "*Low-cost Accelerometers for Robotic Manipulator Perception"*.
[69] Y. K. Thong, M. S. Woolfson , J. A. Crowe, B. R. Hayes-Gill e D. A. Jones "*Numerical double integration of acceleration measurements in noise"*.
[70] T. Harada, H. Uchino, T. Mori and T. Sato, "*Portable Orientation Estimation Device Based on Accelerometers, Magnetometers and Gyroscope Sensors for Sensor Network*".
[71] Matthias Mühlic "*Particle filters, an overview*".

# PUBBLICATIONS

- ✓ C. Bruno, D. Longo, D. Melita, G. Muscato, S. Sessa, G. Spampinato - "Heterogeneous robot cooperation for interventions in risky environments" – IARP International Workshop, Robotics and Mechanical assistance in Humanitarian De-mining and Similar risky interventions – HUDEM 2008 – The American University in Cairo (AUC), March 28-30 2008 - Cairo - EGYPT
- ✓ F. Bonaccorso, C. Bruno, D. Longo, G. Muscato – "Structure and model identification of a vortex-based suction cup" – CLAWAR 2008 11th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines, September 08-10 2008, Coimbra – PORTUGAL
- ✓ F. Bonaccorso, C. Bruno, L. Cantelli, D. Longo, G. Muscato, G. Spampinato, "Control of Welding for Rapid Manufacturing: Two Cases Study". SIDRA08 - September 2008 - Vicenza - ITALY
- ✓ F. Bonaccorso, C. Bruno, L. Cantelli, D. Longo, G. Muscato – "A Modular Software Interface for the Control System of an Arc Welding Robot" – HSI09 Human Machine Interaction in Industry Automation and Home Automation – May 2009 Catania – ITALY
- ✓ F. Bonaccorso, C. Bruno, L. Cantelli, D. Longo, G. Muscato – "Control of a Shaped Metal Deposition Process" –

PHYSCON 2009 (4th International Scientific Conference on Physics and Control) – September 2009 - Catania – ITALY

- ✓ F. Bonaccorso, C. Bruno, L. Cantelli, D. Longo, G. Muscato, S. Rapisarda - "A Closed Loop Welding Controller for a Rapid Manufacturing Process" – TRAM 09 – Trends in m Aerospace Manufacturing – September 2009 – Sheffield – UK
- ✓ F.Bonaccorso, C.Bruno, L.Cantelli, D.Longo, G.Muscato, S. Rapisarda, G.Spampinato - "Control of Welding for Shaped Metal Deposition" SIDRA09 - Semptember 2009 - Siracuse - ITALY
- ✓ P. Aranzulla, F. Bonaccorso, C. Bruno, L. Cantelli, G. Lanteri, D. Longo, D. Melita, G. Muscato and A. Pennisi "An innovative autonomous outdoor vehicle based on Microsoft Robotic Studio" - CLAWAR 2010 - September 2010 - Nagoja - JAPAN
- ✓ F. Bonaccorso, C. Bruno, L. Cantelli, D. Longo, G. Muscato, S. Rapisarda "A feedback control system for a rapid production process based on robotic welding deposition" - ISR2010 - June 2010 - Munich - GERMANY

# COURSE/CONFERENCE/PARTNERSHIP

- ✓ IRIS09 International School in Robotics and Intelligent Systems. Focus theme: Assistive robots and systems for healthcare and rehabilitation. IASI, Romania 2009, 22-26 July part of AT-EQUAL 2009 – Advanced Tecnologies for Enhanced Quality of Life IASI, Romania 2009, 22-26 July
- ✓ Doctorate school SIDRA 2010 "Robotica" - Bertinoro, 12-17 July 2010
- ✓ Doctorate school EMUNI 2010 "Science and Technology: Knowledge and Cultural Aggregation" - Catania, 28 August - 10 September 2010
- ✓ Microchip 16 bit curriculum, Catania, 25-28 May 2009
- ✓ Schneider Electric - PLC Course, Catania, 28-31 July 2008
- ✓ NATO ASI 2010: Advanced All-Terrain Autonomous Systems Çeşme, Turkey: Altin Yunus Resort - August 15 - 24, 2010

Collaboration with ST-Microelectronics – Catania Site – ART group (Automatic Robotics and Transportation) of IMS-System Labs – L7 Building – 3rd floor