Preface

This thesis is submitted as the Master Thesis Project for the Ingeniería Industrial Superior degree in Universidad Carlos III from Madrid (Spain). The work has been done performed during my Erasmus exchange program in the Norwegian Science and Technology University (NTNU) at the Department of Engineering Cybernetics in Collaboration with SINTEF. The supervision of this thesis has been performed by the NTNU professor Geir Mathisen and the SINTEF researcher Aksel Andreas Transeth.

I would like to thank my supervisor Aksel Andreas Transeth for his interest, patience, enthusiasm and excellent advice during this work. In the same way I would like to thank Geir Mathisen for giving me the opportunity to work within the Department of Engineering Cybernetics what has been an enriching experience for my education.

Moreover I would like to thank my parents, brother and sisters for the love and support that they gave me during this experience.

Jesús García Estébanez
Trondheim, September 30, 2009

NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk
Institutt for teknisk kybernetikk

MASTER THESIS

Kandidatens navn: **Jesús García Estébanez**

Fag: **Engineering cybernetics**

-

Oppgavens tittel (norsk): **GPS-IMU integrering for en slangerobot med aktive hjul**

Oppgavens tittel (engelsk): **GPS-IMU integration for a snake robot with active wheels**

Bakgrunn:

Robots shaped as snakes – snake robots – have a vast potential within areas such as seach and rescue, and inspection and maintenance.

Snake robots with active wheels are a specialized form of snake robots. The active wheels are advantageous in made-made environments such as office floors, factories and ventilation systems. The active wheels and the articulated body of the robots offer an efficient platform for moving both horizontally and vertically.

There are several challenges in controlling snake robots with active wheels. These are, for example, how to synchronize the wheels and joints, how to account for wheel slippage, and how to determine the global shape and position of an entire snake robot. The topic of this assignment is to investigate how to estimate the positions and orientations of the snake robot modules and wheels.

Tasks:

1. Perform a literature study with focus on GPS-IMU integration. Include, if possible, results within this topic for snake-like robots.
2. Find and/or develop a suitable model for doing state estimation and simulation (not necessarily the same model).
3. Combine simulated sensor data from IMU, GPS and wheel rotation sensors to estimate snake robot states.
4. Perform simulations and discuss the results.

Oppgaven gitt: 12.02.2009

Besvarelsen leveres: 01.10.2009

Besvarelsen levert:

Utført ved Institutt for teknisk kybernetikk

Veileder: Forsker MSc Aksel A. Transeth, SINTEF Anvendt Kybernetikk

Trondheim, den 12.02.2009

Geir Mathisen

Faglærer

Contents

Abstract 9

1. **Introduction** 12
  1.1. Motivation 12
  1.2. State of the art 12
  1.3. Contribution 13

2. **Thesis outline** 16

3. **Background theory** 19
  3.1. Kalman Filter theory 19
  3.2. Extended Kalman Filter theory (EKF) 22

4. **GPS/IMU fusion** 26
  4.1. History and state of the art 27
  4.2. GPS/IMU fusion implementation 28
  4.3. GPS/IMU fusion implementation. Ideal case 29
  4.4. GPS/IMU fusion implementation considering head angle and commercial sensors 41
  4.5. GPS/IMU fusion. Case with GPS signal interruption 50

5. Navigation system based in physical model 56
5.1. Kinematic model design 57
5.2. Link angle controller 70
5.3. Physic model and inertial sensor fusion 72
5.3.1.Error caused by wheel traction loss 73
5.3.2.Error caused by heterogeneous coefficient of friction 74
5.4. Physical model/inertial sensor fusion 75
5.4.1.Real case simulation 75
5.4.2.Fusion implementation 78

6. Conclusions 84

7. Future work 87

8. References 90

9. Appendix 92

Abstract

A snake robot will be defined herein as any multilink robot for whose shape and motion capabilities are reminiscent of a snake like PiKo [1]. PiKo is a five links snake robot with active wheels designed by SINTEF in collaboration with the Norwegian University of Science and Technology (NTNU).

Researchers have been greatly interested in the development of robots like PiKo because of its shape versatility and motion capacity in difficult terrains. These skills and properties are useful for rescue teams working in earthquakes, pipe inspection operations and other utilities where access and movement in the terrain are typically difficult.

When a working team decides to develop a snake robot, an important point to consider is the development of an efficient navigation system that reaches an accurate position of the robot. Technically speaking, it is prudent to design at the same time a state observer that gives us at least the real time position and velocity information of the robot body to be controlled.

The relevance of this information is derived from every control action applied to the robot will require some information about the situation of the robot over time. The controller will need feedback about the robot dynamics and the effect that the control actions have caused. Typically this information has three sources: the information that comes from external sensors, that from internal sensors that transmitting to the control place the measurements from the sensors in the robot body and estimated data from a physical model.

All of these feedback sources have some advantages and disadvantages. Implementing an external observer, with external sensors, will not cause space problem with sensors location in the robot body, but when the robot is working inside a pipe, underground or in another hard environment where the optical, magnetic or radio frequency contact is difficult or impossible, the information reception from an external observer is too difficult and expensive or simply impossible. Locating internal sensors in the body of the robot may solve has the problem with the measurements reception, but still pose some difficulties which must be considered by the designer. Many times space becomes a problem when locating some sensors inside the robot body due to size and weight constraints within the robot body. Basing the navigation system instead on a physical model that simulates the robot motion invites the possibility of error due to simplifications taken during the mathematical and physical development. It is impossible to develop a perfect physical model since all the variables, forces, and parameters that depend on the nature characteristics usually are random process and we can just raise a useful factor estimating the average of these effects in our concrete situation.

In this thesis a navigation system with a GPS (Global Positioning System) and IMU (Inertial Measurement Unit) fusion was achieved. This navigation system will be work as long the GPS signal is available. The application of the fusion technique further reduces one order the potential errors inherent in using only the GPS navigation system.

When the robot will encounter locations where the GPS signal is impossible, this thesis will present a set of tools that not being a universal solution, it will be a set of mathematical tools that depending on the case could give us an accurate navigation system.

During the time the GPS signal reception is impossible, this thesis presents the development and implementation of a physical model for a snake robot with active wheels which simulates the snake robot running behavior and studies the possibility to use the trajectory estimated by the model for reaching an accurate navigation system.

1. Introduction

1.1. Motivation

If it is desired to control one system (in our case the robot), it will be necessary to feedback some information about the robot dynamics. This information will be essential for controlling the robot and for guiding it to the desired position, velocity or acceleration. There are some cases where it is unknown what position, velocity, acceleration or a combination of these ones the robot has and, in this case, the robot can be difficult to control properly.

Usually these properties are observed by external sensors. Normally these measurements suffer different kinds of noise or perturbations which force the researcher to filter them or make different algorithms that estimate accurately dynamics values. These algorithms are called state observers.

1.2. State of the art

Nowadays, there are not efficient state observers applied for multilink robots. These observers have not been successfully applied because each link has its own frame of reference and the relative movement between the different links makes the system equations extremely nonlinear. This nonlinearity brings to create an accurate linear observe impossible even with linearization techniques

Navigation systems based on the fusion between GPS (Global Positioning System) and IMU (Inertial Measurement Unit) signals have been extensively used in the defense industries from the last thirty years and in other industries like aerospace and automobile more recently.

Before this multisensor fusion technology, the defense industry was guiding the missiles and aircrafts just by Inertial Navigation System (INS) [8]. That technology consisted of a double integration of the acceleration signal from inertial sensors and defined the initial point, that double integration would give us the object trajectory.

The problem of that technology was the accumulative error suffers by the double integration. If the sensor had an offset error signal, the obtained objet trajectory would be more erroneous as longer the offset signal affects the sensor.

Trying to fix that problem, in the seventies, the idea about GPS (Global Positioning System) and IMU (Inertial Measurement Unit) appeared. The GPS/IMU fusion will fix the offset problem, but this strategy will bring some other problems related with the GPS accuracy and GPS signal lost. The history and problems of this fusion are explained more thoroughly in Chapter 4.1.

The state of art of each concrete technique used during this thesis will be explained intensely in each chapter.

1.3. Contribution

In this Master Thesis Project, a set of tools is presented oriented to raise an accurate and effective navigation system in multilink snake robots. These tools can be used to simulate a planar motion of PiKo [1] adjusting previously the model parameters (link lengths, link weight, etc). Figure 1 does not show the actual PiKo aspect but rather is an illustration that how will be PiKo robot with the suitable shell.

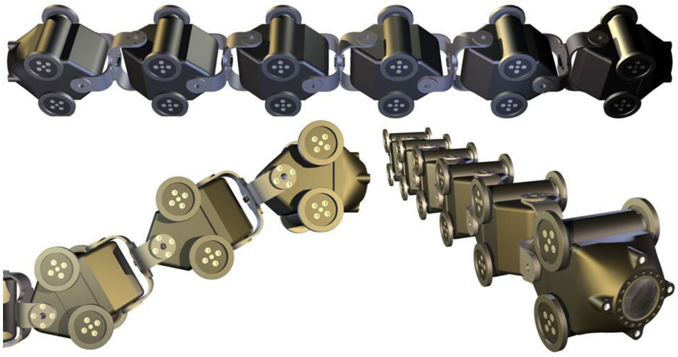


Figure 1 PiKo robot future appearance. [1]

During this thesis we will implement different sensor signal fusion techniques which are focused on obtaining an accurate navigation system. Problems of these techniques and their solutions will be discussed.

First of all, we will implement a multisensor fusion system based on Kalman Filter theory. In this thesis we will present under which conditions the filter works and demonstrate why this method is effective and analyze how the filter behavior is if some unexpected problems happen. For testing this strategy we will develop and simulate different multisensor filters and we will discuss which kind of problems each one has.

In the second part of this thesis we will extend and modify Fukaya, Iwasaki and Saito's physical model [11] by adding wheels for multilink snake robot with the aim to fusion the trajectory given by the physical model with the acceleration information given by the inertial sensors installed in the robot body. In this part, the GPS/IMU fusion strategy will not be useful anymore because we will suppose that the GPS signal is not able to be received.

To observe the system proprieties of the physical model that we have developed, we can take a look on the past because it has been used from the origin of automation. This technique is so risky due to the inherent approximations that we will take during the design of the physical model.

There are some natural coefficients which will be introduced in the model as averaged constants; however in the nature they are not constant and these simplifications will introduce some errors in the physical model. This thesis will study the effect of these model errors in the navigation system.

Summing up, we can divide the robot navigation into two parts. The first one where the fusion between GPS and inertial sensor (IMU) is able and the initial state is defined by the robot operator and a second part, where the robot enters inside an environment where the GPS signal is not available to be received. In the last operation mode, we will use a navigation system based on the physical model which we will develop in the second part of this thesis. The initial condition for this mode will be defined for the last state given by the GPS/IMU Kalman observer when this system was running.

2. Thesis outline

In the first part of this thesis, Chapter 4, we will suppose that the robot is on the surface being able to receive a GPS signal. That means, we have assumed that we have installed a GPS receptor in the snake robot head. In the same way, we have supposed that we have an inertial sensor (IMU) that gives us information about the robot acceleration. With the data we obtain from both sensors, we will apply a fusion technique based on Kalman Filter theory.

This fusion strategy is extensively used in different fields like navigation systems, slip control, aircraft navigation [2], [3] and many other uses. As we said, this fusion will be only available if the robot has a good GPS signal reception although we will also simulate how the filter works if during a short time period the filter looses the GPS signal and the only information that the robot will have will be the one that comes from inertial sensors one.

The second part of this thesis, Chapter 5, will be focus on the development and research of a kinematic physical model for a multilink snake robot with active wheels. At this point the designer has to choose a kinematic model where the friction model between the snake body and the terrain will be an important point to consider. We have supposed that the only contact points the robot has with the terrain are in the wheels. According to this fact, we have selected a friction model that has been used for vehicles tires where the friction forces have a direct relation with the slip between the tire and the terrain [4].

After having developed the physical model, we will analyze how the model works with the simplifications we have done. Comparing the mentioned model with the same model where, trying to simulate the nature, we have included random effects in some variables. For example, the coefficient of friction that in the first model we will simplify like an averaged constant is a variable value in reality.

In Chapter 5 we will also use the information that comes from the physical model and we will fusion it with the inertial sensors which the robot link will have installed. In this chapter, we will explain more about the problems we have found during the attempt of fusion the physical model with external sensors.

The next chapter, Chapter 6, will consist of a discussion about the obtained results. An exposition where we will explain the implementation viability, the problems we have found during the development and research of the navigation system and how we have solved it will follow. Basically, this point will try to analyze how effective the work has been done in the previous points and it will open us some research opportunities that will be interesting to analyze.

The end of this thesis, Chapter 7 will consist of a discussion about future works which this thesis opens. Difficulties and suggestion for further system improvements will be presented.

3. Background theory

In this chapter, our goal will be to introduce the reader the mathematical tools we have used for a multisensor fusion. If we want to fusion the information that comes from different sensors, we will have to make a mathematical model that includes the sensor we are using. Basically, the sensor in the model will be a noisy input. In our case we assume that it is a white noise the one that will be affecting the measurements. We assume that the nature of this noise follows a Gaussian distribution where the mean is located in zero and has a variance that is directly dependent on the precision of the sensor. Typically the amplitude of this white noise is defined in the sensor data sheet.

3.1. Kalman Filter theory

The Kalman filter is a state estimator developed by a Hungarian mathematician called Rudolf E. Kalman (1930 -) in 1960 [5]. Kalman formulated a recursive filter which for a linear system and being able to observe some noisy outputs from the system, the filter gives us a state estimation that minimizes the noise effect in the observer estimation. The conditions for the proper filter working are that the system has to be linear, as we said before, and the noisy output has to be affected by a white noise. This mathematical tool has been extensively used due to in many systems we can measure the output with the adequate sensor that we can represent like an output affected by a white noise. We can find many documents with the demonstration that this filter minimizes the noise effect in the estimation [6].

During this chapter we will detail the algorithm presented by Kalman in 1960 [7]

Given a process with a state vector $x \in \Re^n$, the first step is to define our linear system in the space state representation as

$$\boxed{\begin{aligned} x_k &= Ax_{k-1} + Bu_{k-1} + w_{k-1} \\ z_k &= Hx_k + v_k \end{aligned}} \tag{1}$$

where w_k and v_k are the state noise effect (proportional with the sample time) and the measurement noise effect (proportional with the sensor accuracy) respectively.

Before introducing the Kalman Filter we have to define some variables which will play important roles in the algorithm.

Define the *a priori* error as the error between the real state and the state estimation before the measurement z_{k-1} as

$$e_{(k|k-1)} = x_k - \hat{x}_{(k|k-1)} .$$

In the same way we can define the *a posteriori* error as the difference between the real state and the state estimation after the measurement z_k. In other words

$$e_{(k|k)} = x_k - \hat{x}_{(k|k)} ,$$

where the *a posteriori* state estimation is

$$\hat{x}_{(k|k)} = \hat{x}_{(k|k-1)} + K\left(z_k - H\hat{x}_{(k|k-1)}\right)$$

where according to the Kalman filtering theory, the Kalman gain is defined as

$$K_k = P_{(k|k-1)}H^T\left(HP_{(k|k-1)}H^T + R\right)^{-1} .$$

Given the a posteriori and a priori errors the corresponding estimation covariance matrices will be

$$\begin{aligned} P_{(k|k-1)} &= E\left[e_{(k|k-1)}\, e_{(k|k-1)}{}^T\right] \\ P_{(k|k)} &= E\left[e_{(k|k)}\, e_{(k|k)}{}^T\right] \end{aligned} .$$

According to the assumption that both noises have a white noise nature, we can define the matrices Q and R as the covariance matrices of each noise respectively. Specifically,

$$\begin{aligned} p(w) &\rightarrow N(0,Q) \\ p(v) &\rightarrow N(0,R) \end{aligned} . \tag{2}$$

So the recursive Kalman Filter runs in two stages as Figure 2 shows. A first stage where the algorithm calculates the next stage and a prediction of the covariance error matrix and a second stage where the algorithm improves the estimation considering the error covariance matrix we have calculated previously. Illustrating the Kalman Filter running we have the Figure 2.

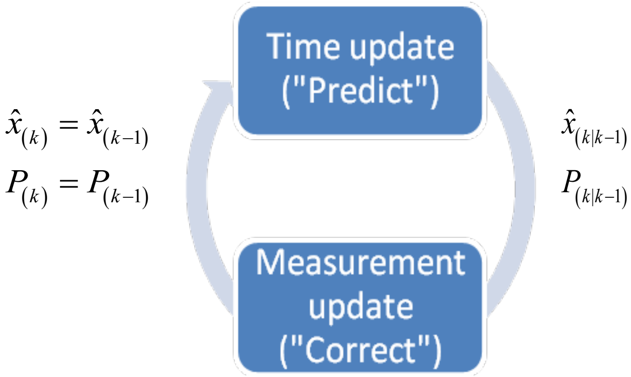


Figure 2 **Kalman loop sketch.**

.

The calculus done in each stage are:

TIME UPDATE STAGE

Project the state ahead: $\Rightarrow \hat{x}_{(k|k-1)} = A\hat{x}_{k-1} + Bu_{k-1}$

Project the error covariance ahead: $\Rightarrow P_{(k|k-1)} = AP_{k-1}A^T + Q$

MEASUREMENT UPDATE

Compute the Kalman gain:

$$\Rightarrow K_k = P_{(k|k-1)}H^T\left(HP_{(k|k-1)}H^T + R\right)^{-1}$$

Update the estimated with the new measurement:

$$\Rightarrow \boxed{\hat{x}_{(k|k)} = \hat{x}_{(k|k-1)} + K_k\left(z_k - H\hat{x}_{(k|k-1)}\right)} \quad (3)$$

Update error covariance:

$$\Rightarrow \boxed{P_{(k|k)} = \left(I - K_k H\right) P_{(k|k-1)}} \quad (4)$$

Applying this algorithm recursively, we will arrive to the better possible state estimation given the conditions previously explained.

We have seen how useful and which accurate estimation the Kalman filter will give us if the process fulfills the requirements previously explained. The problem is that this filter only works with a linear system and white noise. For skip the linear limitation the Extended Kalman Filter (EKF) was developed.

3.2. Extended Kalman Filter theory (EKF)

Let us assume that a system has a state vector $x \in \Re^n$. Now, the nonlinear process can be defined by

$$\boxed{\begin{aligned} x_k &= f(x_{k-1}, u_{k-1}, w_{k-1}) \\ z_k &= h(x_k, v_k) \end{aligned}}, \quad (5)$$

where f and h the nonlinear difference equations.

In reality, of course the values w_k and v_k are unknown in each time step. However we could arrive to a first approximated measurement as

$$\begin{aligned} \tilde{x}_k &= f(\hat{x}_{k-1}, u_{k-1}, 0) \\ \tilde{z}_k &= h(\tilde{x}_k, 0) \end{aligned}, \quad (6)$$

where $\hat{x}_k$ is some *a posteriori* estimated state (from a previous step).

Before applying the filter, we have to linearize the system (5). Our goal will be to arrive to the next representation

$$\boxed{\begin{aligned} x_k &\approx \tilde{x}_k + A\left(x_{k-1} - \hat{x}_{k-1}\right) + W w_{k-1} \\ z_k &\approx \tilde{z}_k + H\left(x_k - \tilde{x}_k\right) + V v_k \end{aligned}}, \quad (7)$$

where each variable means:

- x_k and z_k are the actual state and the output vector (measurement) respectively

- $\tilde{x}_k$ and $\tilde{z}_k$ are the approximations of the vectors done in (6).
- w_k and v_k are random noise that affect the state vector and output vector (measures) respectively.
- A is the linearization of f against the space state vector x. In other words it is the Jacobian matrix with respect the state vector:

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}\left(\hat{x}_{k-1}, u_{k-1}, 0\right). \tag{8}$$

- W is the linearization of f against the noise vector W. In the same way we will calculate the Jacobian matrix with respect the noise vector. Specifically,

$$W_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}}\left(\hat{x}_{k-1}, u_{k-1}, 0\right). \tag{9}$$

- H is the linearization of the output with respect the state vector calculating the Jacobian matrix. So,

$$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}\left(\tilde{x}_k, 0\right). \tag{10}$$

- V is the linearization of the output with respect the noise vector v calculating the Jacobian matrix as

$$V_{[i,j]} = \frac{\partial h_{[i]}}{\partial v_{[j]}}\left(\tilde{x}_k, 0\right).$$

Defined the new matrix, the recursive algorithm will have the same two stages represented in Figure 2.

In the EKF the mentioned two steps will be as

TIME UPDATE STAGE

Project the state ahead. $\Rightarrow \hat{x}_{(k|k-1)} = f\left(\hat{x}_{k-1}, u_{k-1}, 0\right)$

Project the error covariance ahead. $\Rightarrow P_{(k|k-1)} = A_k P_{k-1} A_k^{\ T} + W_k Q_k W_k^T$

MEASUREMENT UPDATE

Compute the Kalman gain.

$$\Rightarrow K_k = P_{(k|k-1)} H_k^{\ T}\left(H_k P_{(k|k-1)} H_k^{\ T} + V_k R_k V_k^T\right)^{-1}$$

Update the estimated with the new measurement.

$$\Rightarrow \boxed{\hat{x}_{(k|k)} = \hat{x}_{(k|k-1)} + K_k\left(z_k - h\left(\hat{x}_{(k|k-1)}, 0\right)\right)} \quad (11)$$

Update error covariance $$\Rightarrow \boxed{P_{(k|k)} = \left(I - K_k H_k\right) P_{(k|k-1)}} \quad (12)$$

It is easy to observe that applying EKF the computational cost will increase a lot against the ordinary Kalman filter due to we have to calculate A and H matrices in each step.

That computational cost should be a problem to consider because if the robot dynamic is high, it will make us to apply the estimator with a small time step which makes it impossible to do the work in real time with the technology we have today. Anyway in this thesis we will not be so concerned about the computation cost due to the simulations will not be done in real time.

4. GPS/IMU fusion

During this chapter, we will introduce the reader the GPS (Global Positioning System) and IMU (Inertial Measurement Unit) signals fusion in different cases oriented to the development of state observer which will be used for obtaining an accurate navigation system.

First we will explain the origins of this technology. Subsequently, we will study three application cases beginning from an ideal case and finishing with a case where we have implemented real sensors properties.

In the first two cases we will not consider the robot head angle[1]. In other words, this means that the inertial sensor theoretically will give us the measurements in a global or absolute frame of reference.

In this first case, our state vector components, that we want to estimate or observe, will consist of the position, velocity, and acceleration per each axis. The sensors will be sampling only the position and the acceleration and both signals will enter in the system affected by a random noise. Also, we will simulate how the system works if the sensors suffer an offset error in the signal.

In the other two simulated cases, we will introduce one factor more. The inertial sensor will not give us the measurement in an absolute or global frame of reference so we should apply a rotation matrix according to the head angle for obtaining the measurements in the global or absolute frame of reference as the GPS does. We will not have any sensor that measures the head angle but instead the IMU (Inertial

---

[1] We will call head angle, the angle between the absolute frame of reference and the frame of reference aligned with the first link.

Measurement Unit) will have a gyroscope that provides us the head angle velocity. Our new state vector will contain the head angle and the head angle velocity.

As we did in first case, we will simulate the effect of an offset signal in the sensor.

The navigation system will be successful if we can reduce the error that the navigation system just based on the GPS receiver gives us.

4.1. History and state of the art

The fusion between GPS/IMU sensors has been extensively used in the aircraft and defense industries in the last two decades.

The origin of this technology comes from the Cold War where the Soviet Union and the United States of America armies met a challenge in getting the most accurate and sophisticated guidance system for their intercontinental ballistic missiles.

After the Second World War and before the Cold War, the ballistic guide system was based in the inertial navigation theory. The inertial navigation system is an integrative system that for its accumulative nature increases the error as long as the system is running. It was in the end of fifties when the Soviet Union launched the first satellite (Sputnik I 1957) where the idea about the guidance of ballistic missiles with satellites signals appears.

This military engineering research and development concerning the satellite guidance reached its peak in 1972 when the U.S. Army and U.S. Air Force started to create the GPS (Global Positioning System) satellite net.

At the beginning this technology was restricted for military uses. During those days the best accuracy possible was bigger than 5 meters. Anyway, although still the accuracy and precision were not so good, the defence industry focused its effort and researches on trying to develop an efficient navigation and navigation system based on the fusion between GPS and Inertial Navigation (INS) technology [8].

Step by step the GPS system became opened for civil uses. First the USA army introduced an error in the signal for civil uses making some uses of the GPS technology difficult or impossible when our civil application, using the GPS signal, needs a high precision. This signal degraded was called Selective Availability and introduced an error close to 100 meters. Since the year 2000, the U.S. army has stopped degrading the signal. Anyway the precision available in these days for civil uses continues to be far from the military one because some frequencies (L2 frequencies) are still closed and encrypted for civil uses.

Nowadays many civil uses are trying the fusion between GPS and IMU for navigation or navigation systems. The better precision that we can raise with the GPS evolution DGPS (Differential Global Positioning System) and working in some areas called Augmentation areas will be about 10 cm although it is almost impossible to reach this precision dynamically.

This improvement in the precision and accuracy in the GPS system has opened many research lines in the fusion between GPS and Inertial sensor for civil uses [9] as the one we will analyse in this thesis.

4.2. GPS/IMU fusion implementation

We are going to analyze how the GPS/IMU system works in different situations and discuss the relevance for each strategy with respect to usage on snake robots.

The first case we will analyze will be an ideal case where we suppose that the GPS signal could be sampled in the same frequency as the inertial sensor, X and Y acceleration components, are sampled respect the global or absolute frame of reference. This means that we are considering that the IMU sensor frame of reference is aligned with the global frame of reference during the whole simulation.

The second case of study will be more realistic. In this case, the GPS sample time and the precision will be taken from commercial GPS receivers. Besides, the IMU sensor will not be aligned during the simulation with the global system and its angle will change giving us the measurements respect the sensor frame of reference instead of the global one. The angle will be measured by a gyroscope which will introduce a new error factor in the system.

In the third case, we will analyze how the navigation system works in an ideal case if the designed state observer which is based on the Extended Kalman Filter suddenly looses the GPS signal and has to work only with the IMU sensor. This situation will open new study points for future researches and works.

For all the cases we will apply the Kalman Filter fusion technique which mathematical development was explained in the Chapter 3.1.

Let us illustrate how a state observer works when we define it in the space state form.

Observe that our noisy process follows the sketch illustrated in Figure 3.

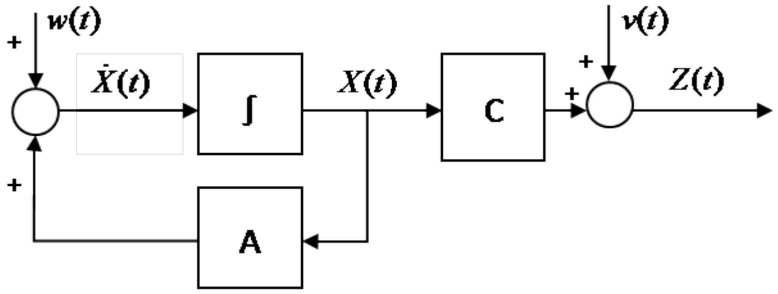


Figure 3 **Linear system block diagram.**

where we are not able to measure the state $X(t)$ (state vector) in each time step. The only information we can observe is the output $Z(t)$ through the sensor we have located in the observed system. So summarizing, the observer will be a process that estimates completely or partially the state vector observing the output vector that for us is the measurements sampled. The estimated state vector is $\hat{X}(t)$ in the diagram. So the observer attached to the system as the Figure 4 shown.

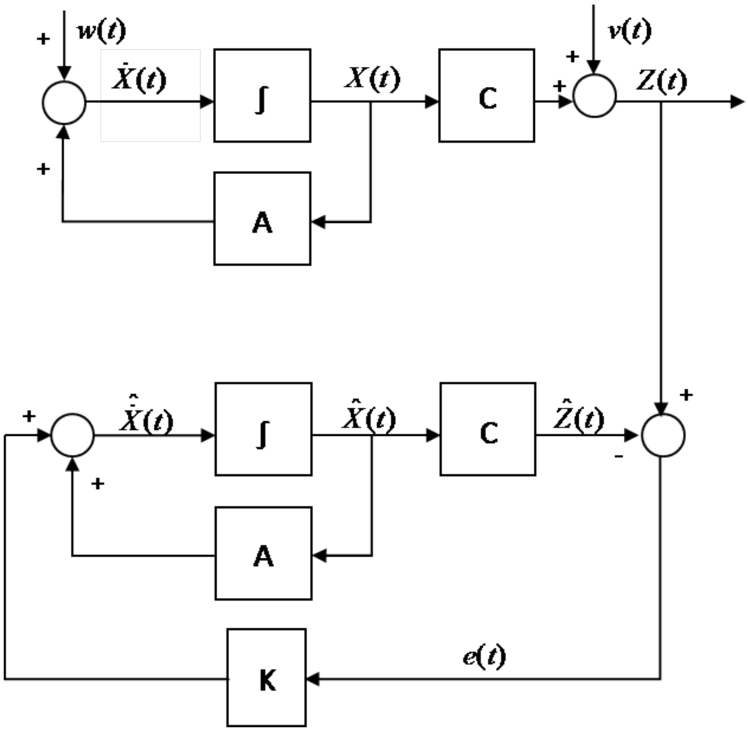


Figure 4 **Linear system with state observer block diagram.**

4.3. GPS/IMU fusion implementation. Ideal case

The first issue to consider will be to define the characteristic of the sensors which will give us the information about the state of the object. As we said previously, in this point we will not make the simulation with real sensor characteristics although the IMU sensor properties will be based on the properties of the MTI-G AHRS manufactured by XSENS. (Appendix)

	GPS sensor **(per axis)**	**Inertial sensor** **(per axis)**
Bias error	1 m	0.02 m/ s^2
Full scale	-	±50 m/s^2
Sample frequency	120 Hz	120 Hz

Table 1 **Sensor characteristics. Ideal case.**

Having defined the sampling characteristics, it is time to define the system that represents our process. Remember that on this chapter we have considered that the GPS (Global Positioning System) and the IMU (Inertial Measurement Unit) sensors will give us the X and Y measurements components in the absolute frame of reference.

Define a continuous system without noise in the space state representation as

$$\begin{aligned}\dot{x}(t) &= Ax(t) \\ z(t) &= Cy(t)\end{aligned},$$

where the state vector is defined as

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \\ x_5(t) \\ x_6(t) \end{bmatrix} \Rightarrow \begin{cases} \mathrm{x}_1 = \text{x position of the robot head.} \\ \mathrm{x}_2 = \text{y position of the robot head.} \\ \mathrm{x}_3 = \text{x velocity of the robot head.} \\ \mathrm{x}_4 = \text{y velocity of the robot head.} \\ \mathrm{x}_5 = \text{x acceleration of the robot head.} \\ \mathrm{x}_6 = \text{y acceleration of the robot head.} \end{cases}$$

so we can simply observe that

$$\begin{aligned}&\dot{x}_1 = x_3 \text{ and } \dot{x}_2 = x_4 \\ &\dot{x}_3 = x_5 \text{ and } \dot{x}_4 = x_6\end{aligned}.$$

Remember that we are measuring the output Y so the matrix C will define which variables or variable combinations we are measuring. According to this and remembering that we will measure the position and acceleration in each component, the matrix C will be

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

which means that the vector we will sample will be

$$z(t) = \begin{bmatrix} z_1(t) \\ z_2(t) \\ z_3(t) \\ z_4(t) \end{bmatrix} \Rightarrow \begin{cases} \text{x position mesurement from the GPS.} \\ \text{y position mesurement from the GPS.} \\ \text{acceleration mesurement along x-axis from the IMU sensor.} \\ \text{acceleration measurement along y-axis from the IMU sensor.} \end{cases}.$$

So summarizing our continuous system, we have that

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \\ \dot{x}_4(t) \\ \dot{x}_5(t) \\ \dot{x}_6(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \\ x_5(t) \\ x_6(t) \end{bmatrix}$$

$$\begin{bmatrix} z_1(t) \\ z_2(t) \\ z_3(t) \\ z_4(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \\ x_5(t) \\ x_6(t) \end{bmatrix}.$$

Before starting to calculate the filters parameters we have to check the system observability. This checking will confirm us that we will be able to estimate the state vector in each time step, sampling the system output with the sensors we have installed supposing the initial state is known.

The way to check the system observability will be to demonstrate that the observability matrix defined by

$$M_o = \begin{bmatrix} C \\ CA \\ CA^2 \\ CA^3 \\ CA^4 \\ CA^5 \end{bmatrix}$$

has the same rank like the number of variables to observe.

The easiest way to confirm the observability is to use the Matlab function *obsv( )* which, introducing the system defined in the space state form as input in the Matlab function, Matlab will give us the observability matrix M_o. If later we apply the Matlab function *rank( )* to the observability matrix obtained previously, Matlab will give us the rank of the matrix and we will confirm that is completely observable if the matrix rank is the same as the number of state variables.

In Chapter 3.1 we defined the Kalman Filter algorithm for a discrete system. Due to the system defined in the continue form we have to discretize the system before applying the discrete Kalman filter algorithm. We will not explain the used process to transform the continuous system to the discrete form because there are many mathematical tools that we can use to do it like the Matlab function *c2d()*.

After applying the discretization tool, we have obtained the next discrete space state representation for a sample frequency of 120 Hz. Remember that in this case we suppose that the GPS is able to sample using the same velocity like IMU sensor has.

$$c2d\left(A\right)=\begin{bmatrix}1 & 0 & 0.00833 & 0 & 3.472\cdot10^{-5} & 0\\ 0 & 1 & 0 & 0.00833 & 0 & 3.472\cdot10^{-5}\\ 0 & 0 & 1 & 0 & 0.00833 & 0\\ 0 & 0 & 0 & 1 & 0 & 0.00833\\ 0 & 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}. \qquad (13)$$

$$C=\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}$$

With the system written in the discrete form we will be ready to apply the Kalman estimator.

.In the first case of analysis we have simulated a trajectory like it is shown in the Figure 5.

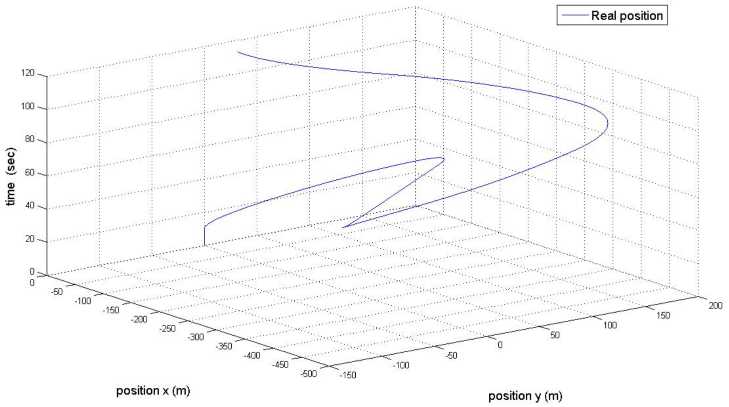


Figure 5 **Real trajectrory.**

This trajectory, according to the sensor specifications we have written in the beginning of this chapter and supposing that the IMU sensor has not any offset, will generate the measurements per axis which we will present in Figure 6 compared with the real signal we are measuring. In Figure 6 we can observe the position and acceleration measurement we have obtained in the GPS and IMU sensors respectively in each axis.

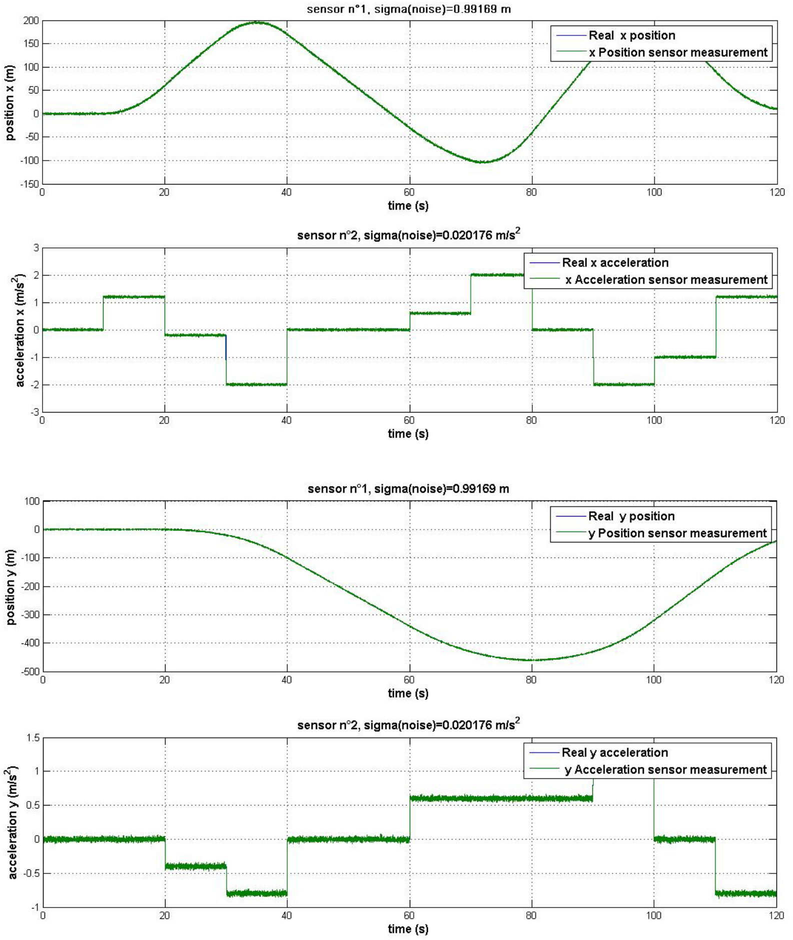


Figure 6 **Measurements obtained in both axes.**

The first question that we should answer is if it is logic to apply Kalman Filter fusion technique for this problem. If we want to corroborate this question, we have to demonstrate that applying Kalman Filter technique, we will improve the position navigation instead to have obtained the trajectory integrating two times the IMU sensor signal. The last option in comparison to the Kalman Filter is easier in the mathematical development and computational cost.

The initial point in the analysis will be to study how well the direct double integration of IMU signals works. The algorithm will be as simple as integrate two times the acceleration and given the initial position to obtain the robot trajectory.

In Figure 7 we can observe the results obtained by the direct integration of the acceleration signal from the inertial sensor.

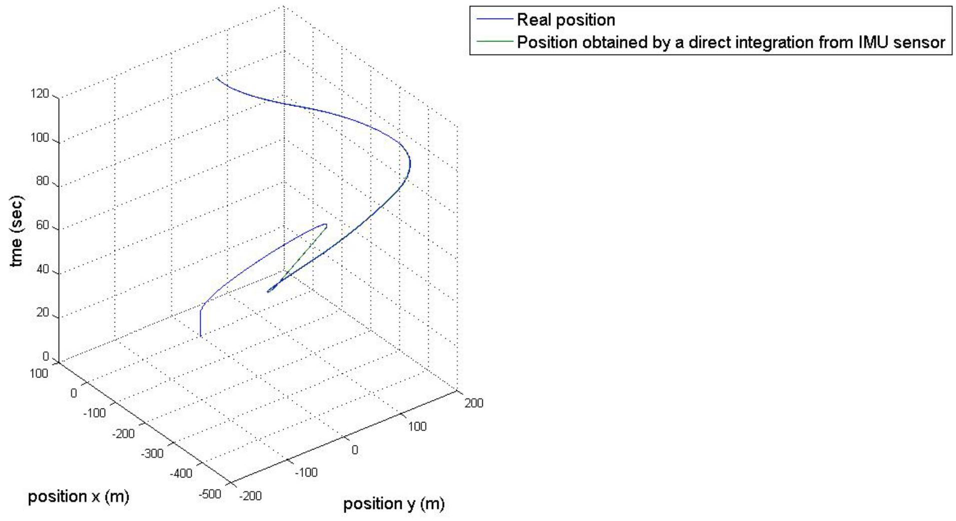
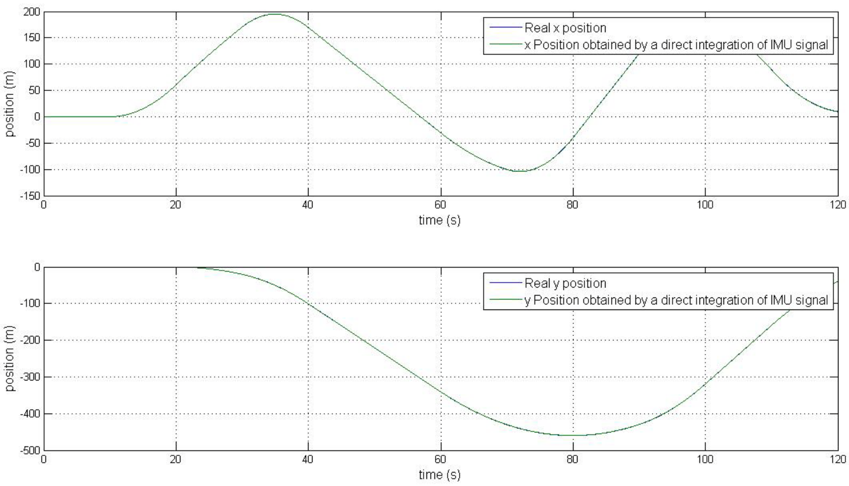


Figure 7 **Robot real trajectory vs. GPS measurement.**

The results obtained for the X and Y axes are



.

Figure 8 **Trajectory per axis from a direct integration of the IMU sensor.**

Having a look at Figure 7 and Figure 8, the two times integration of the IMU signals gives us a satisfactory navigation system and the calculated trajectory follows the real trajectory rightly. That phenomenon is due to the elimination of the effect of a pure white noise by the integration.

We could assert that in this first case of analysis it has no sense to implement a Kalman Filter due to the accurate estimation obtained by the direct integration. That conclusion makes us wondering why this method is extensively used in the fusion of Inertial Navigation with GPS.

To answer this question we should analyze how an offset noise that enters in the IMU signals will affect the trajectory navigation obtained by a directly integration of the mentioned signals.

Introducing an offset noise values as

$$a_{x\,offset} = 0.01 \, {}^{m}\!/\!{}_{s^2}$$

$$a_{y\,offset} = 0.01 \, {}^{m}\!/\!{}_{s^2}$$

we should observe how the direct integration works in this case.

According to the trajectory shown in Figure 5 and considering the offset defined previously, the measurements along the X axis are shown in Figure 9.

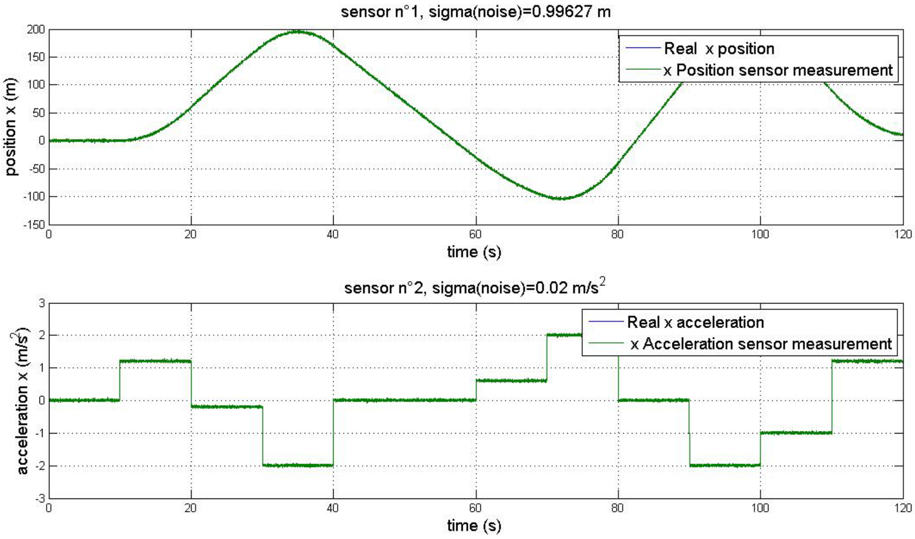


Figure 9 **X axis measurements (with offset).**

As in the previous figure, the measurements along the Y axis measurement are illustrated in Figure 10.

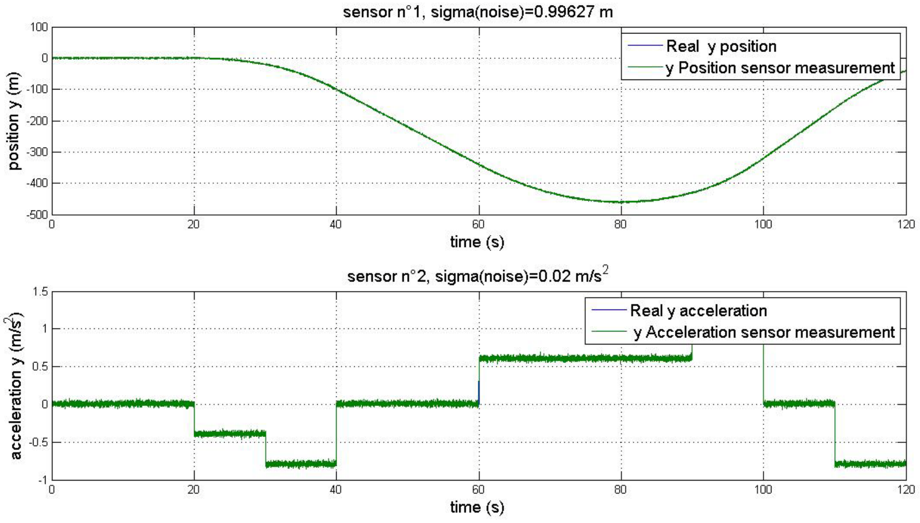


Figure 10 **Y axis measurements (with offset).**

Observe that the offset error signal value is small and in Figure 9 and Figure 10 we cannot appreciate the introduced error.

Illustrating how the error affects the trajectory navigation, Figure 11 shows how the obtained trajectory after integrate two times the acceleration signals shown in Figure 9 and Figure 10 is.

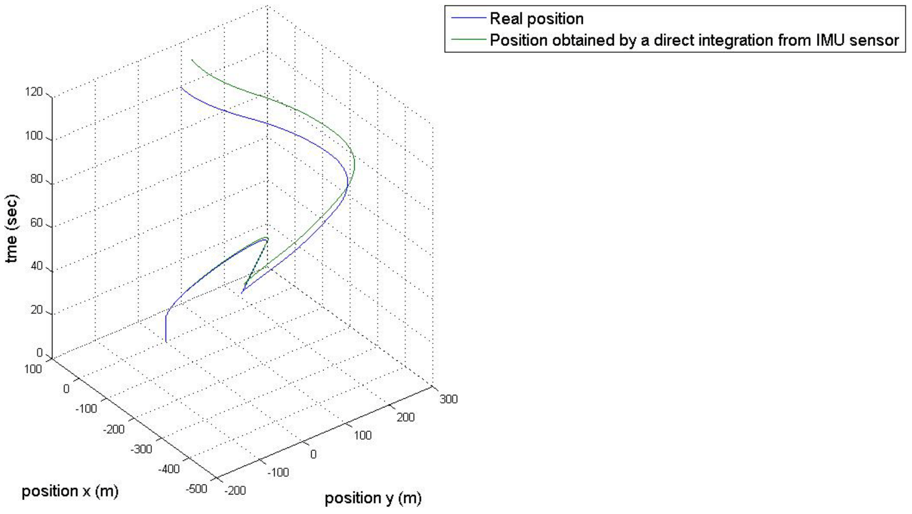


Figure 11 **Real trajectory vs. trajectory obtained by direct integration from IMU signal.**

As we can observe in Figure 11, with the direct integration we do not reach acceptable trajectory navigation. Observing Figure 12, we can better understand how the offset introduces an accumulative error, caused by the integration, in each axis.

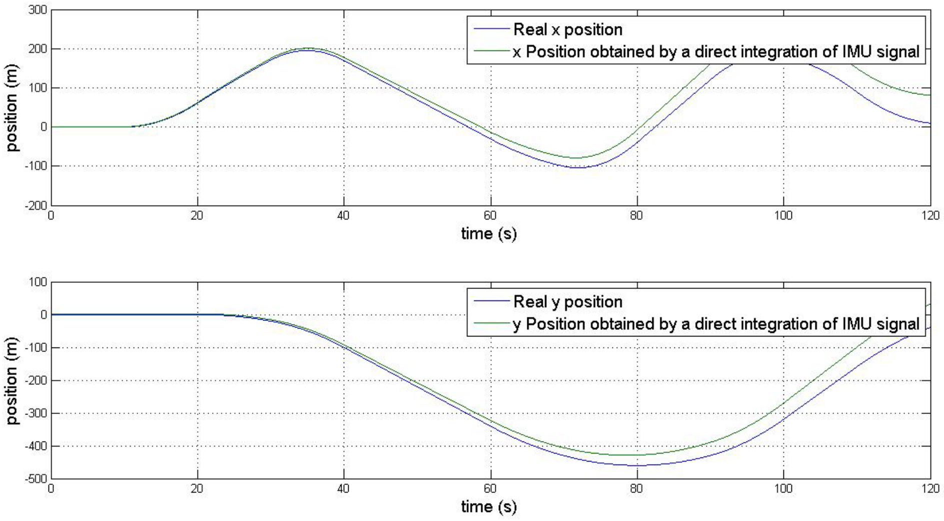


Figure 12 **Real position vs. position obtained by direct integration of Imu signal.**

The unsatisfactory trajectory we have obtained by the direct integration of the inertial signal will make us reject this option.

The navigation system which is just focused on the GPS signal will obtain a better precision in the position of each axis in comparison to the direct integration of the inertial sensors. Just using the GPS navigation system we can obtain a measurement error close to 1 m. We can observe the trajectory obtained by the GPS system in Figure 13.

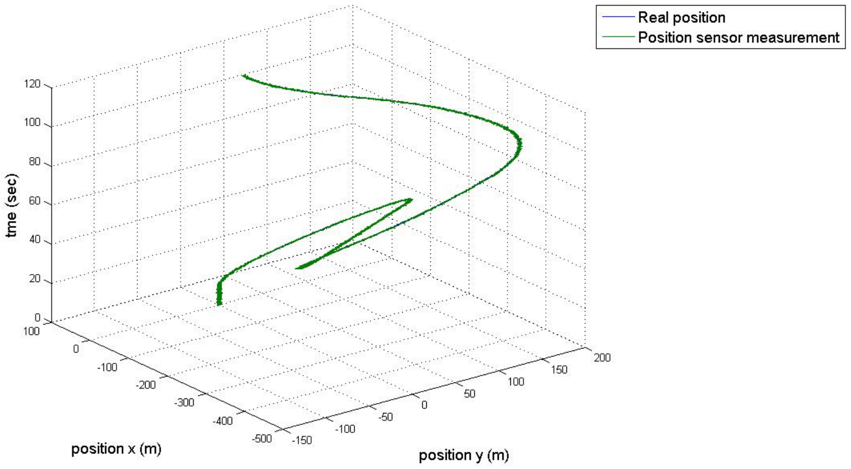


Figure 13 **Real position vs. GPS measurement.**

The results obtained in the two previous attempts, suggested a try to implement the Kalman Filter technique and to check if we can reach a better navigation system when the sensors are affected by an offset error.

Our noisy system will continue being defined like we did in equation (1) where the discrete matrix A and C are defined in equation (13).

Firstly we have to define how the noises enter in the system or in other words we have to define R and Q matrices. In equation (2) we defined the R matrix as the matrix which represents the covariance error in the variables. The variables are independent so the R matrix will just have components in the diagonal. These covariance values will be dependent on the sensors bias error.

$$R = \begin{bmatrix} v_1^2 & 0 & 0 & 0 \\ 0 & v_2^2 & 0 & 0 \\ 0 & 0 & v_3^2 & 0 \\ 0 & 0 & 0 & v_4^2 \end{bmatrix}$$

v_1 = Position error in x axis from GPS sensor.
v_2 = Position error in y axis from GPS sensor.
v_3=Acceleration error x axis from the IMU sensor.
v_4=Acceleration error y axis from the IMU sensor.

The Q matrix, in its discrete form depends on the sample time that we have chosen because Q represents the error which we made during the discrete integration. In the Q matrix, if the α factor is high, the filter will converge faster but so oscillating instead if the value is small the filter will converge slow but will arrive to an accurate estimation [10].

$$Q^* = \begin{bmatrix} T_{sample}^5/20 & 0 & T_{sample}^4/8 & 0 & T_{sample}^3/6 & 0 \\ 0 & T_{sample}^5/20 & 0 & T_{sample}^4/8 & 0 & T_{sample}^3/6 \\ T_{sample}^4/8 & 0 & T_{sample}^3/3 & 0 & T_{sample}^2/2 & 0 \\ 0 & T_{sample}^4/8 & 0 & T_{sample}^3/3 & 0 & T_{sample}^2/2 \\ T_{sample}^3/6 & 0 & T_{sample}^2/2 & 0 & T_{sample} & 0 \\ 0 & T_{sample}^3/6 & 0 & T_{sample}^2/2 & 0 & T_{sample} \end{bmatrix} \tag{14}$$

$$\Rightarrow Q = \alpha\, Q^* / norm(Q^*)$$

Defined the system, it is time to apply the recursive Kalman filter as is shown in Figure 2.

The first thing we have to define is the initial point of the system in the algorithm and how the matrices are defined in the first step in the loop. The matrices in the first time steps are

$$\hat{x}_{k-1} = x(0)\,, \quad P_{k-1} = Q\,.$$

After defining the initial point we are ready to apply the recursive Kalman filter explained in this thesis in Chapter 3.1.

Remember that our measures were the ones plotted in Figure 9 and Figure 10 and the real trajectory is shown in Figure 5.

After applying the Kalman Filter, during the state estimation in the X axis is illustrated in Figure14.

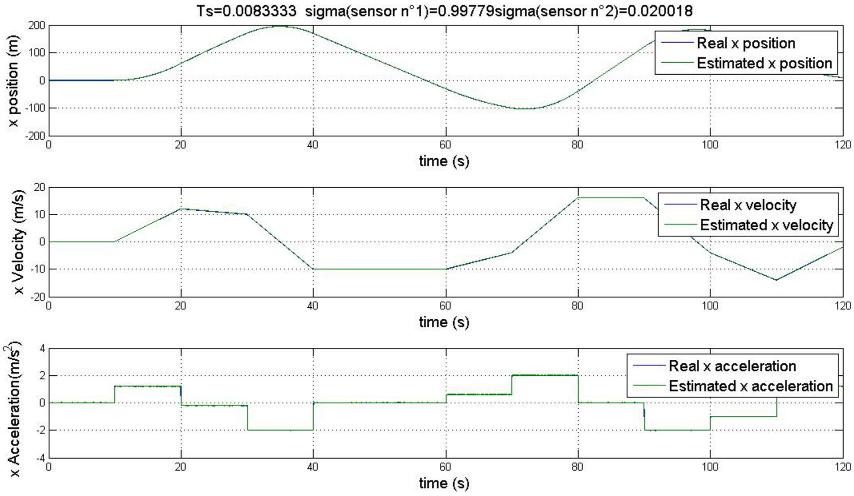


Figure 14 **Real state vs. state estimation X axis.**

In the same way, during the state estimation in the Y axis is depicted in Figure 15.

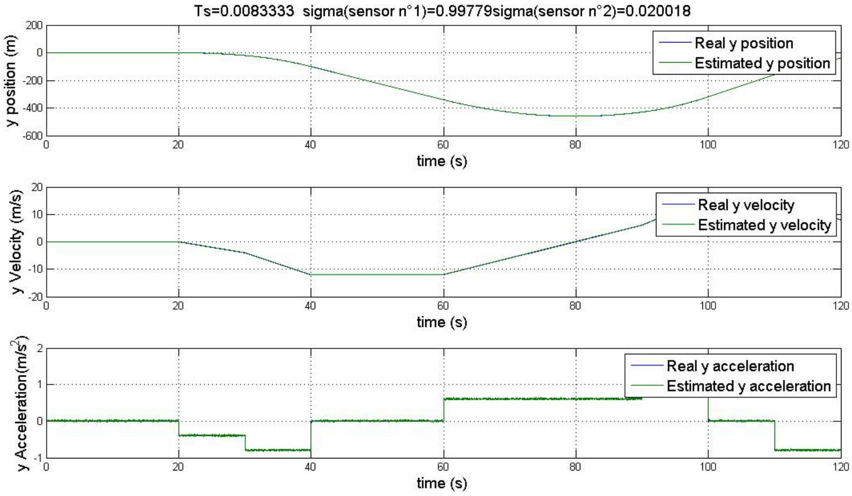


Figure 15 **Real state vs. state estimation Y axis.**

In Figure 16, we can observe how well the estimated trajectory follows the real trajectory shown in Figure 5 making the differentiation between both lines impossible.

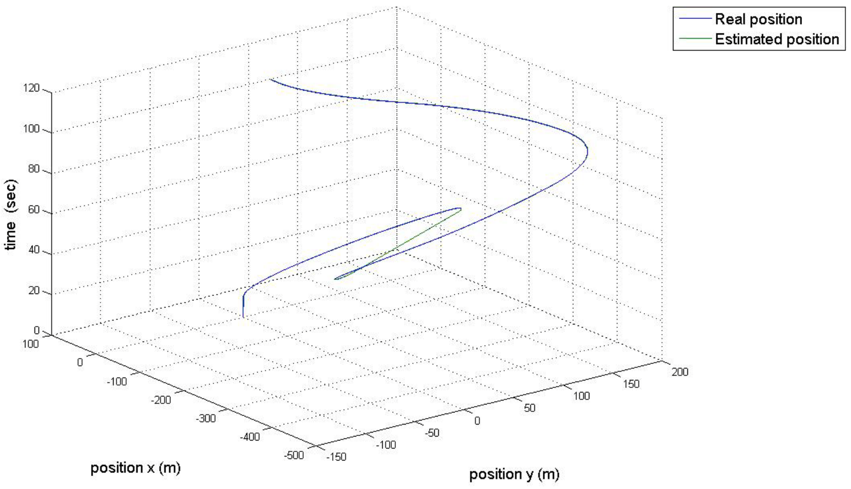


Figure 16 **Real trajectory vs. estimated trajectory.**

Observing Figure 15 and Figure 16, we could think that the trajectory navigation system applying Kalman fusion filter gives us a better result than the other systems which were analyzed before (GPS navigation and double integration) because it is impossible to appreciate the difference between the two lines. If we want to verify this assertion we should make an error analysis and conclude that the error which is made is lower than the GPS error.

A good factor for measuring the error will be the Root Mean Square Deviation (RMSD).

The RMSD error is defined as

$$\boxed{RMSD(x)=\sqrt{\frac{\sum_{i=1}^{n}\left(\hat{x}_i-x_i\right)^2}{n}}}, \tag{15}$$

where n is the number of samples we have taken during the simulation.

Applying equation (15) in our simulation we have obtained the next results:

RMSD(error($x_{position}$)) = 0.5860593 m

RMSD(error($y_{position}$)) = 0.4930698 m

RMSD(error(V_x))= 0.09659628 m/s

RMSD(error(V_y)) = 0.09613739 m/s.

With the obtained results we can confirm that the Kalman Filter fusion technique works quite well in this case. We have obtained a navigation system much better which reduces the error to the half value in comparison to the navigation system based on in the GPS system. Thanks to the obtained results for the state observer, any control action that we will apply to the object will be more efficient due to the state feedback will be more truthful.

The simulations and the Kalman implementation have been obtained executing the Matlab script *main_multi_dc_kf.m* included in the CD attached to this thesis.

4.4. GPS/IMU fusion implementation considering head angle and commercial sensors

In this chapter, we will try to make the filter work in a more realistic case. First of all, we will work with commercial GPS receivers and we have to consider the angle between the IMU frame of reference and the absolute or global frame of reference. This angle will be measured by a gyroscope that is installed in the IMU (Inertial Measurement Unit). This gyroscope will introduce a new error in the observation similar to the other sensors did.

Another point is that in Chapter 4.3, we supposed that we could have the sensor information decoupled in each axis from the absolute frame of reference. That means that the IMU sensor axes and the absolute or global frame of reference have the same direction. Instead, in this new case, we are going to generate a new trajectory and we are going to consider that the head angle (where the sensor is located) will change so the IMU frame of reference and the global or absolute frame of reference will not be aligned .

In the same way like we did in the first case, we have to define the sensor properties. The sensor properties have been taken from the sensor MTI-G AHRS manufactured by XSENS. (Appendix)

In the following table we present the sensor properties for the next case of analysis.

	GPS sensor (per axis)	**Inertial sensor (per axis)**	**Gyroscope**
Bias error	2.5 m	0.02 m/s^2	1 deg/s
Full scale	-	±50 m/s^2	±300 deg/s
Sample frequency	4 Hz	120 Hz	120 Hz

Table 2 Sensor characteristics. Head angle.

We have to observe the relationship between the sensor measurements frame of reference and the global measurements frame of reference. Illustrating this relaioonship we have Figure 16.

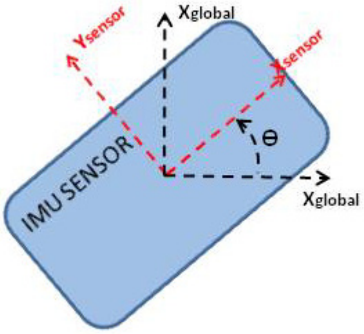


Figure 17 **IMU sensor sketch attached to the snake robot head.**

Considering that we have a new variable θ that represents the head angle and with the gyroscope we can measure the variation of this angle $\dot{\theta}$, we have to redefine the system. The new system will be rewritten as

$$x(t)=\begin{bmatrix} x_{pos}(t) \\ y_{pos}(t) \\ v_x(t) \\ v_y(t) \\ a_x(t) \\ a_y(t) \\ \theta(t) \\ \dot{\theta}(t) \end{bmatrix} \Rightarrow \dot{x}(t)=\begin{bmatrix} 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0 \end{bmatrix} x(t)+w(t) \qquad (16)$$

in the continuous form.

Another time thee output equation will represent the measurements we are sampling. Given the measurements vector

$$z(t)=\begin{bmatrix} z_1(t) \\ z_2(t) \\ z_3(t) \\ z_4(t) \\ z_5(t) \end{bmatrix} \Rightarrow \begin{cases} \text{x position measurement from the GPS.} \\ \text{y position measurement from the GPS.} \\ \text{acceleration measurement along the x-axis from the IMU sensor.} \\ \text{acceleration measurement along the y-axis from the IMU sensor.} \\ \text{angular velocity measurement from the IMU sensor.} \end{cases}$$

expression for the output equation will be the next one.

$$z(t) = \begin{bmatrix} x_{pos} \\ y_{pos} \\ \cos(\theta) a_x + \sin(\theta) a_y \\ -\sin(\theta) a_x + \cos(\theta) a_y \\ \dot{\theta} \end{bmatrix} + v(t) \tag{17}$$

It is easily observable the effect of a rotation matrix which changes between the absolute frame of reference to the relative system from the IMU (Figure 17). This rotation matrix makes the system non linear. That nonlinearity will make us to regret the idea to apply the ordinary Kalman Filter as we did in the previous case. As we explained in the Chapter 3.2, for trying to skip this limitation the Extended Kalman Filter (EKF), which is also explained in the mentioned chapter, was developed.

So according to the equations (5) and (7) we have to calculate the linearization matrices.

The A matrix will not change because there is not any nonlinearity in the equation (16). The only change respect to the equation (16) will be that we should express the equation in the discrete form. We will discretize it with the Matlab function *c2d( )* using the bigger sample time of the sensors, in our case the GPS sample time 4 Hz.

$$A = \begin{bmatrix} 1 & 0 & 0.25 & 0 & 0.03125 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0.25 & 0 & 0.03125 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.25 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{18}$$

Remember that the H matrix is the $h(x)$ linearization by the Jacobian matrix (10).

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cos(\hat{\theta}_{k-1}) & \sin(\hat{\theta}_{k-1}) & -\sin(\hat{\theta}_{k-1})\hat{a}_{xk-1} + \cos(\hat{\theta}_{k-1})\hat{a}_{yk-1} & 0 \\ 0 & 0 & 0 & 0 & -\sin(\hat{\theta}_{k-1}) & \cos(\hat{\theta}_{k-1}) & -\cos(\hat{\theta}_{k-1})\hat{a}_{xk-1} - \sin(\hat{\theta}_{k-1})\hat{a}_{yk-1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{19}$$

This matrix will be calculated before applying the Kalman equations in the $\hat{x}_{k-1}$ step.

The noise in this system enters like a linear effect. For that reason we will not calculate W and V because it is the identity matrix. That means that $V_k R_k V_k^T = R$ and $W_k Q_k W_k^T = Q$ as in the ordinary Kalman Filter.

The Extended Kalman Filter (EKF) will be more accurate as smaller the sample time is. The problem we have here is that the sample times we have in both sensors are different. The GPS sensor is sampling with a frequency of 4 Hz meantime the inertial sensor and the gyroscope are sampling with a frequency of 120 Hz.

We could try to extrapolate data between two GPS samples. However with this strategy we would not increase the filter accuracy and perhaps we could increase the effect of the noise.

Explained the problem with the different sample times, if we want to apply the EKF, we will have to resample the IMU signal, before apply the filter (angle velocity and inertial measurements), with the GPS frequency.

For resampling the signal, we could take one sample each ${}^{T_{sample\,IMU}}\!/\!{}_{T_{sample\,GPS}}$. This algorithm works like a low pass filter and it is dangerous because we could filter some frequencies that are important for the navigation system. According to the Nyquist theorem, we cannot observe any process that has a dynamic which is faster than half of the sampling frequency. In other words, in our system a dynamic faster than 2 Hz will not be able to be observed by the multisensensor Kalman filter unless we change another GPS system with a higher sample frequency. These sensors are much more expensive or these GPS receivers are restricted for military uses. For using these frequencies it is necessary to obtain special permission.

Coming back to the resampling problem, the better strategy we could apply is using *resample()* Matlab tool. This tool will resample the signal taking care about aliasing effect because Matlab will make a frequency analysis and the resampled vector will be better than the one obtained taking one each ${}^{T_{sample\,IMU}}\!/\!{}_{T_{sample\,GPS}}$ samples.

Another time, if we consider that the sensors are only affected by a white noise and they have not any offset in the signal, the navigation system that we could get integrating two times the inertial sensor and obtained the angle just integrating the gyroscope signal, will be enough accurate or at least we could reach a better navigation system that just observes the GPS signal.

Observing Figure 18, we can see the trajectory obtained by the direct integration of the inertial signal compared with the real trajectory that continues being the one shown in Figure 5.

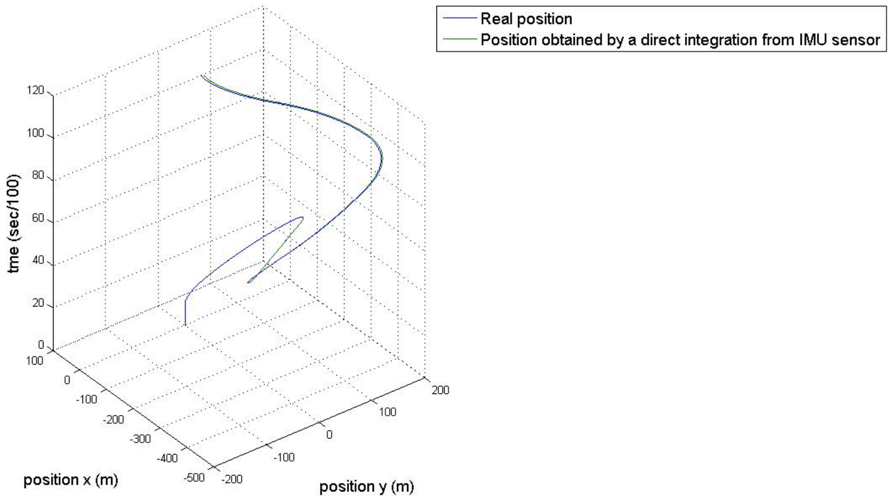


Figure 18 **Real trajectory vs. trajectory obtained by direct integration from IMU signal.**

And the trajectory per axis obtained after two integrations of the acceleration is

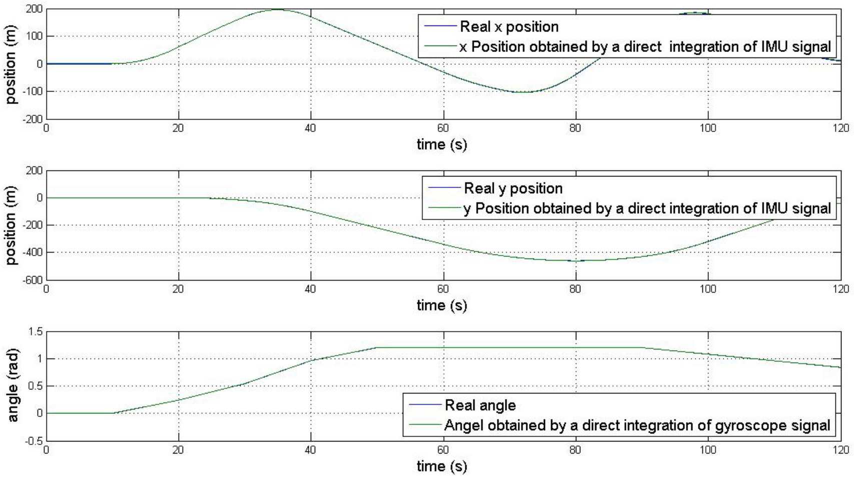


Figure 19 **Real position vs. position obtained by direct integration of sensor signals.**

Although graphically we can observe that the estimation follows pretty well the real trajectory we need to calculate the RMSD error to confirm the last assertion.

The values obtained, applying the (15) equation are the following ones:

$$\mathrm{RMSD}(\mathrm{error}(x_{position})) = 1.0472540 \text{ m}$$

$$\mathrm{RMSD}(\mathrm{error}(y_{position})) = 0.4535007 \text{ m}$$

We can observe that the error which we made is not lower than the GPS error so in this case, when we have supposed no offset error in the signal, is not justified to apply Kalman Filter. The use of Kalman Filter will make our navigation system using powerful processors and the computability cost will be elevated.

As we did in the previous case of analysis, we will analyze what happens with the double integration when the sensors are affected by an offset error. Another time, like in the other cases, the real trajectory will be the one shown in Figure 5.

The offset error introduced in the sensor signals will be the next ones:

$$a_{x\,offset} = 0.01\, m/s^2 \quad a_{y\,offset} = 0.01\, m/s^2 \quad \dot{\theta}_{offset} = 1.10^{-3}\, rad/s$$

With this offset, the measurements per axis obtain the one shown in Figure 20. Observe that the offset error is small and the figure seems like a white noise added to the real value.

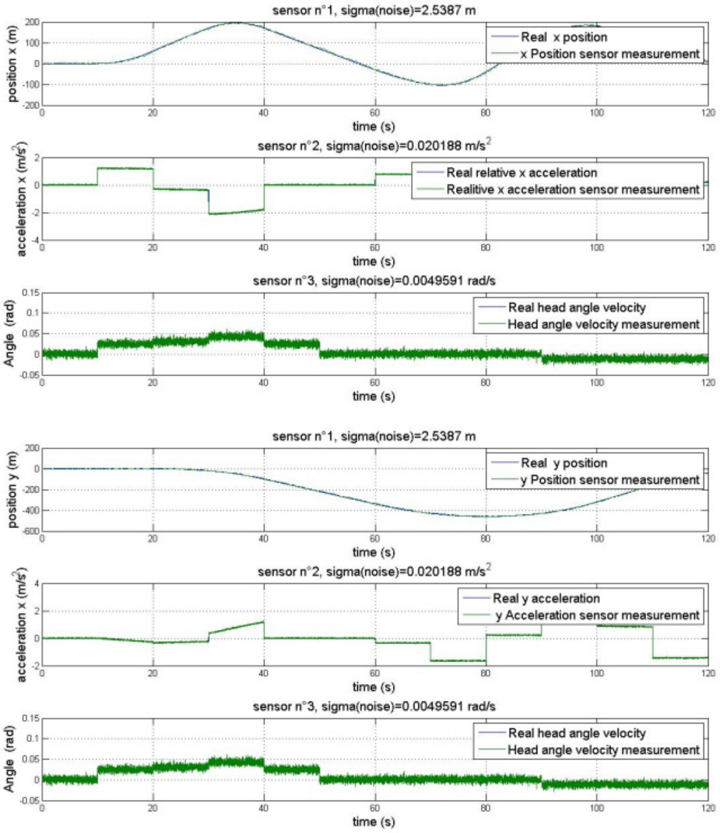


Figure 20 Measures obtained in both axis.

Applying the rotation equation in each step we will obtain the acceleration in the absolute frame of reference. If these absolute signals are integrated two times we will obtain the position.

After to have integrated the acceleration signals two times, the trajectory obtained is depicted in Figure 21.

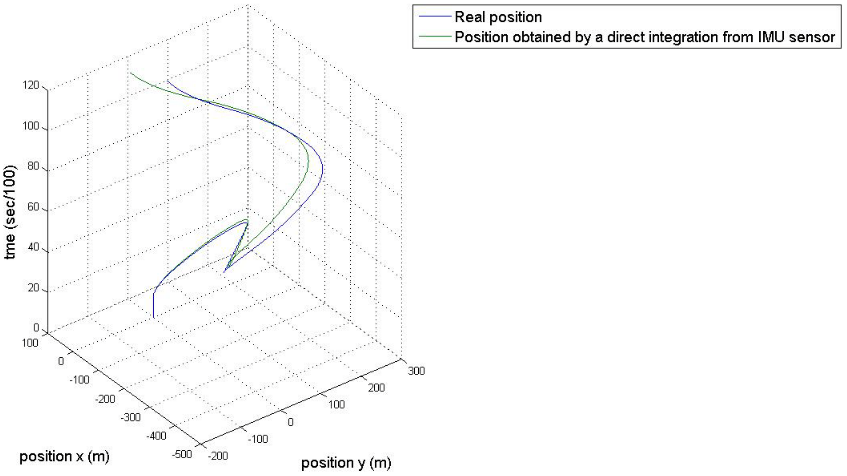


Figure 21 **Real trajectory vs. trajectory obtained by direct integration from IMU signal.**

Applying the double integration the results obtained per axis are shown in Figure 22.

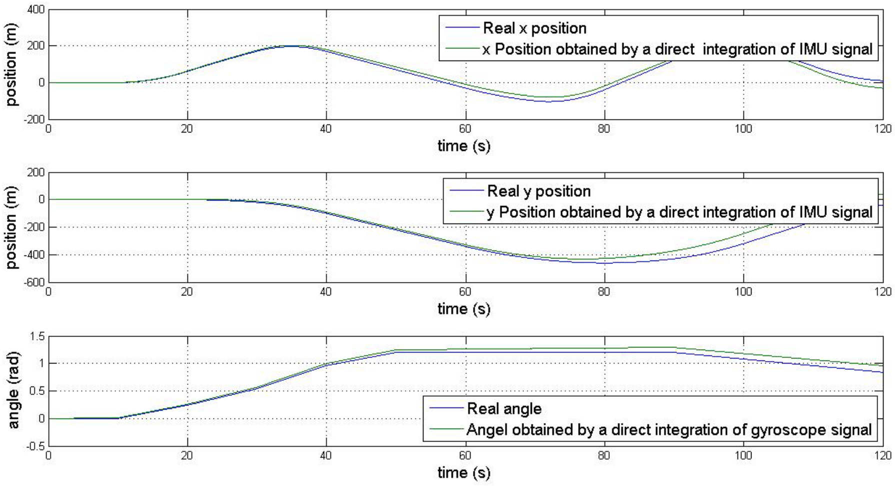


Figure 22 **Real position vs. trajectory obtained by direct position from IMU signal.**

In Figure 22 is easily to observe that we are having a huge error integrating two times directly the acceleration and this error is accumulative and will increase over the time. Another time for confirming this assertion, we will analyze the RMSD error.

RMSD(error($x_{position}$)) = 15.701296 m

RMSD(error($y_{position}$)) = 37.558238 m

If we take attention to the error that the navigation system suffers by the direct integration, we will realize that the error is much bigger that the error that the GPS gives us measuring the position. We could conclude that the integration will not be good method for an accurate navigation system because the error will be increasing over the time if the sensors are affected by an offset error.

It is time to analyze how the EKF works and if it is able to correct the offset error and if we are able to arrive to a better navigation system than the error made is lower than the error that the GPS navigation system has. In this case, if we have been able to reach a better estimation, we will conclude that the EKF will be an acceptable method for navigation the trajectory.

The real trajectory in the simulation will be the one shown in Figure 5 as we assumed in the previous cases and the measurements taken by the sensors will be another time the ones shown in Figure 20.

The offset error will be the same than the one we used in the previous point for analyze the effect of a direct double integration of the IMU signals

After applying the method explained in Chapter 3.2, defined the system like it is written in the equations (16) and (17), and calculated for each step the H matrix as the equation (19) says, the estimated trajectory obtained is shown in Figure 23.

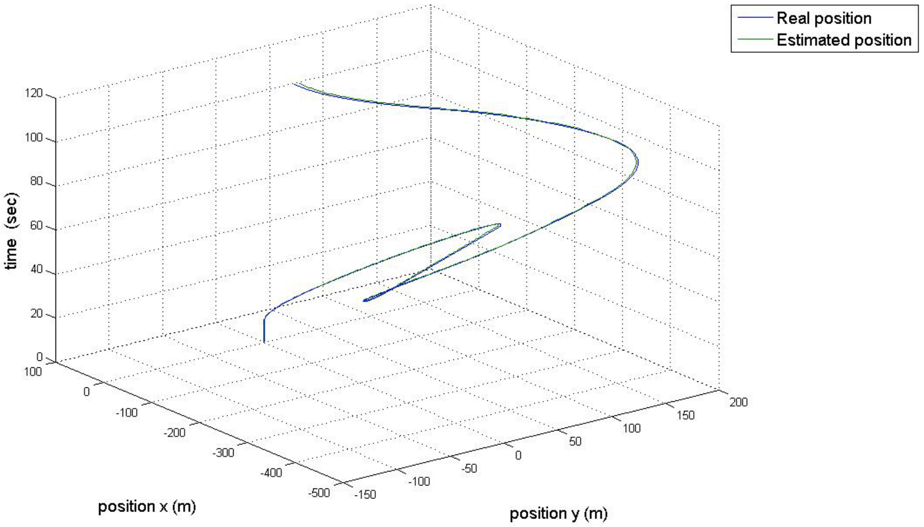


Figure 23 **Real position vs. estimated position.**

During state estimation for the X axis is illustrated in Figure 24.

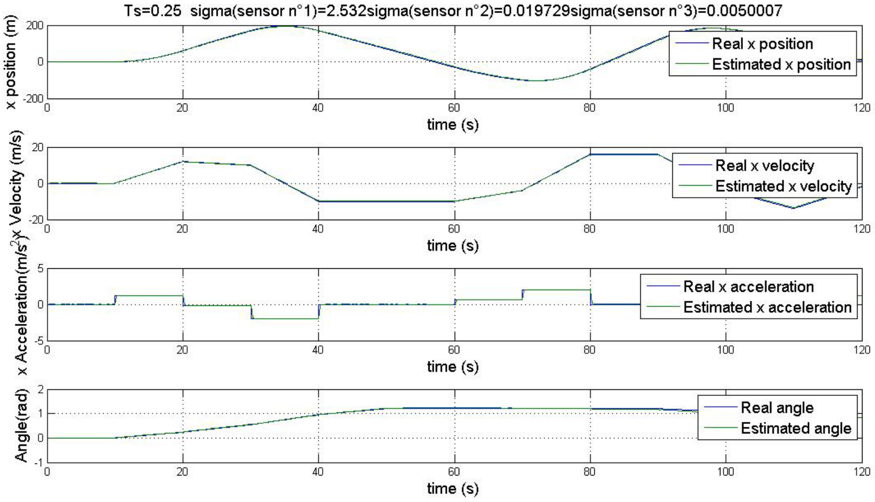


Figure 24 **State estimation for X axis. Real case with offset error.**

In the same way, the estimation for the Y axis is shown in Figure 25.

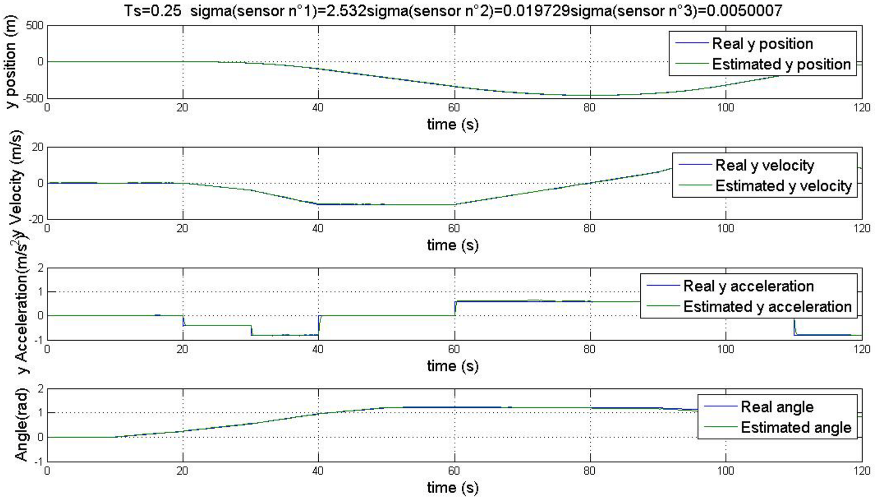


Figure 25 **State estimation for Y axis. Real case with offset error.**

Just observing Figure 24 and Figure 25, it is easy to conclude that the estimation will be quite accurate. Another time, if we want to know if the estimation has been successful and the obtained results have been better than the GPS navigation system we will analyze the RMSD error between the estimation and the real trajectory.

The error per axis calculated according with equation (15).will be

$$\text{RMSD}(\text{error}(x_{position})) = 1.4306315 \text{ m}$$

$$\text{RMSD}(\text{error}(y_{position})) = 0.9558090 \text{ m}$$

The error suffered after applying EKF is lower than the precision that the one that the GPS navigation system has. We can conclude that if we are working with commercial sensors and in a real case (suffering offset error in the signals) the Extended Kalman Filter method will give us an important improvement and the implementation of powerful microprocessor focused on the navigation system could be justified.

Every chart and results shown in this chapter can be checked running the Matlab script *main_multi_dc_kfextended.m.*

4.5. GPS/IMU fusion. Case with GPS signal interruption

In this chapter we will try to analyze how the Kalman Filter works in a case where we had a problem in the measurement system. It is so usual that GPS systems loose the contact with satellites giving a null signal. Simulate this situation will be a good way to analyze how sensitive the filter is when the measurements which enter in the filter are unusual values.

Usually when we have to design an observer we have to choose between having a quickly observer which arrives to a fast estimation and to suffer a high oscillation or to arrive to a slow estimation and the estimation will be so stable. Many times the dynamic of the system forces us to use a fast observer because if we use a slow observer and the dynamic is really fast the observer will not be able to follow the system.

The observer will be more sensitive as faster it is and if an unusual value is sampled and the effect in the filter will be higher. So finally the filter design will be a choice between the velocity that involves imprecision and sensitivity and precision and dynamics limitations. As we have explained in the equation(14), the convergence velocity will be defined for a proportional value which multiplies the Q matrix.

In the previous case, for a quite realistic situation where the received signals were affected by a white noise and an offset noise the Kalman Filter we have gave us satisfactory results decreasing the uncertainty that the GPS navigation has. In this case we will simulate the same trajectory navigation but in this case, between the 2 second and the 5 second, the GPS will lose the satellite signal and instead the GPS will introduce just a noise signal in the filter.

The simulation will be another time with the following offset noise signals.

$$a_{x\,offset} = 0.01\, {}^{m}\!/_{s^2}$$

$$a_{y\,offset} = 0.01\, {}^{m}\!/_{s^2}$$

$$\dot{\theta}_{offset} = 1.10^{-3}\, {}^{rad}\!/_{s}$$

The real trajectory in the analyzed simulation will continue being the one shown in Figure 5.

This time the measurement taken by the GPS receiver will be.

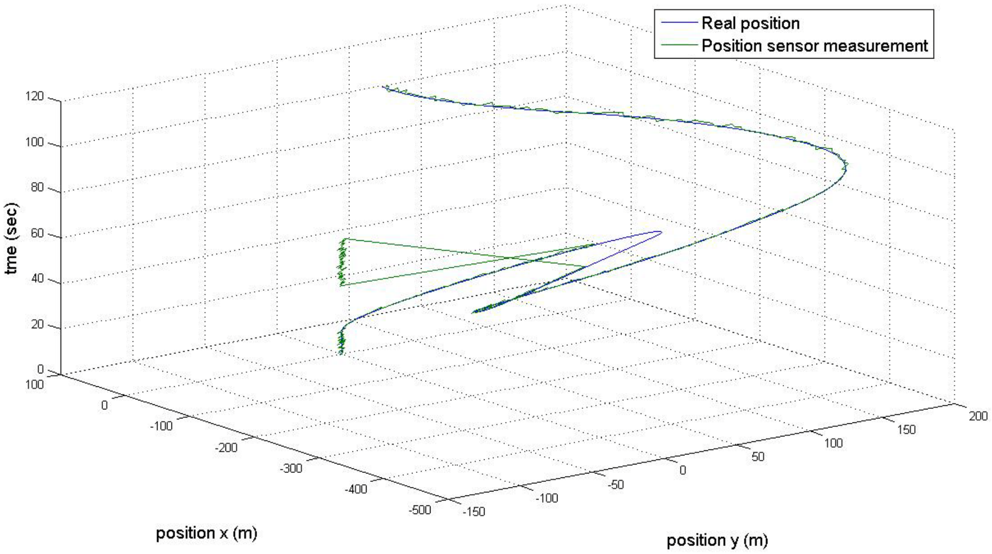


Figure 26 **Real trajectory vs. GPS measurements.**

Observe that between the 30 and 50 seconds, the GPS signal is 0 and introduce a fake measurement in the filter.

Per axis, the measurements will be the next ones. In the X axis the measurement obtained are shown in Figure 27.

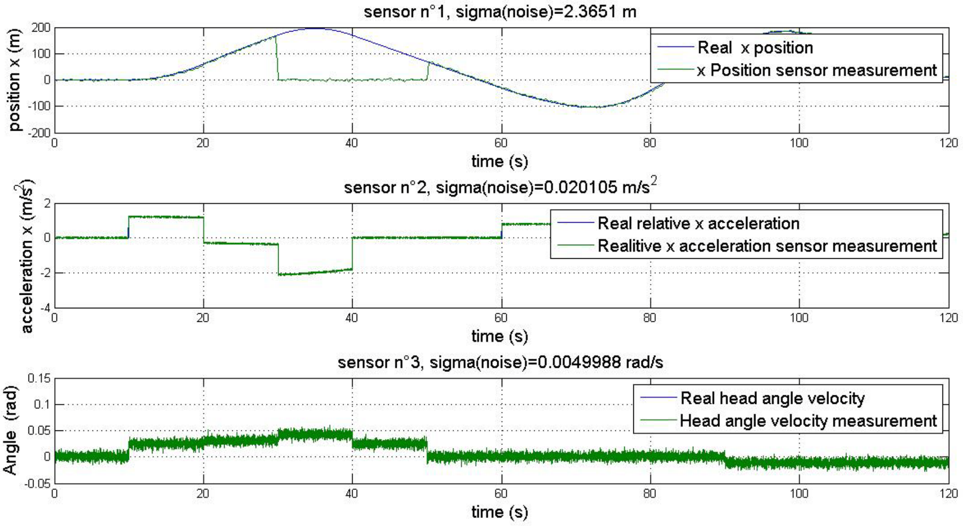


Figure 27 **Sensors measurements X axis. GPS signal lost.**

In the Y axis the measurements obtained are depicted in Figure 28.

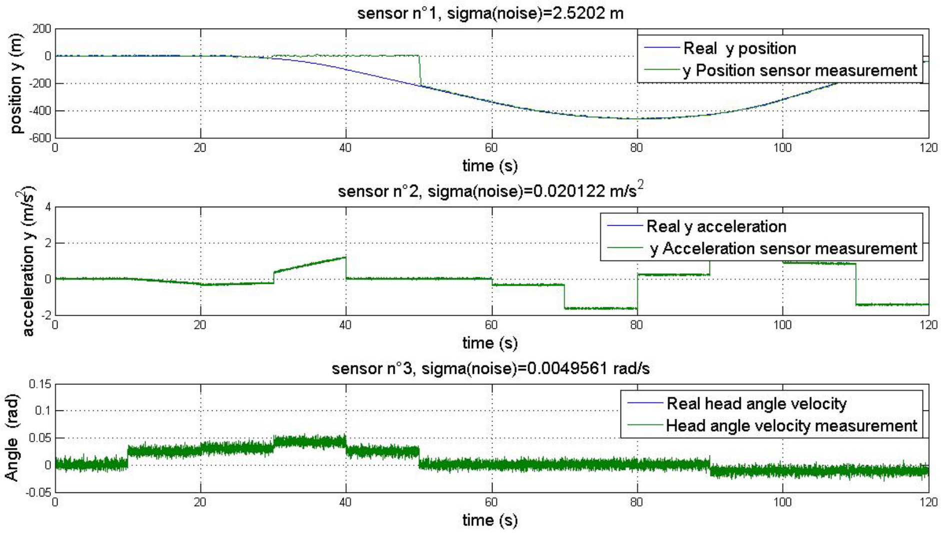


Figure 28 **Sensors measurements Y axis. GPS signal lost.**

In Figure 27 and Figure 28 are easy to observe the period of time where the GPS signal is 0 and how the inertial sensors and the gyroscope from the IMU continue measuring correct values without any problem.

We will study how the filter works with different α values in the Q matrix (equation (14)).

In the first case, defined $\alpha = {T_{sample}}/{10}$ in Q, after apply the EKF the trajectory estimated by the system is shown in Figure 29.

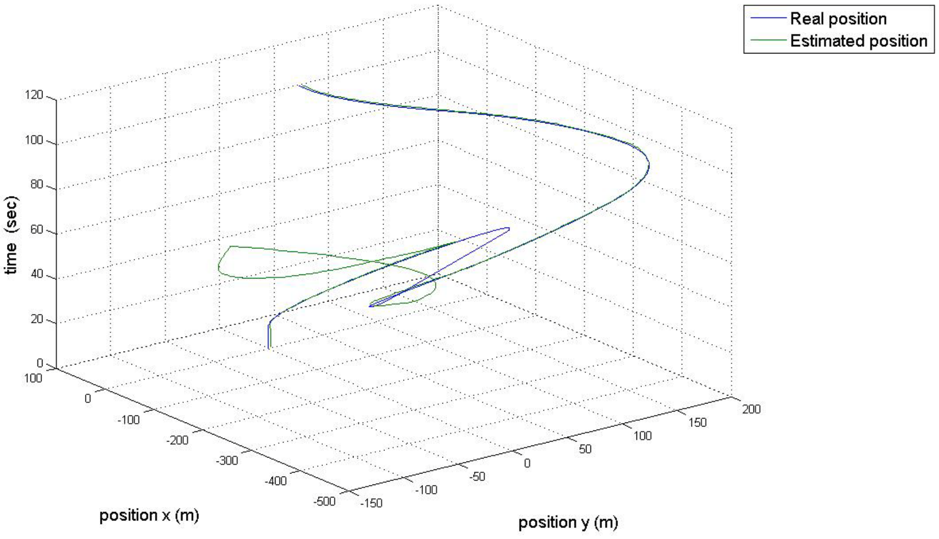


Figure 29 **Real trajectory vs. estimated trajectory.**

In the previous and the following charts we can see how the filter is so sensitive to an unusual or fake value but instead one time the GPS receiver gives the filter a real measurement the system recovers quickly an acceptable estimation.

After applying the EKF, the state estimations in the X axis is illustrated in Figures30.

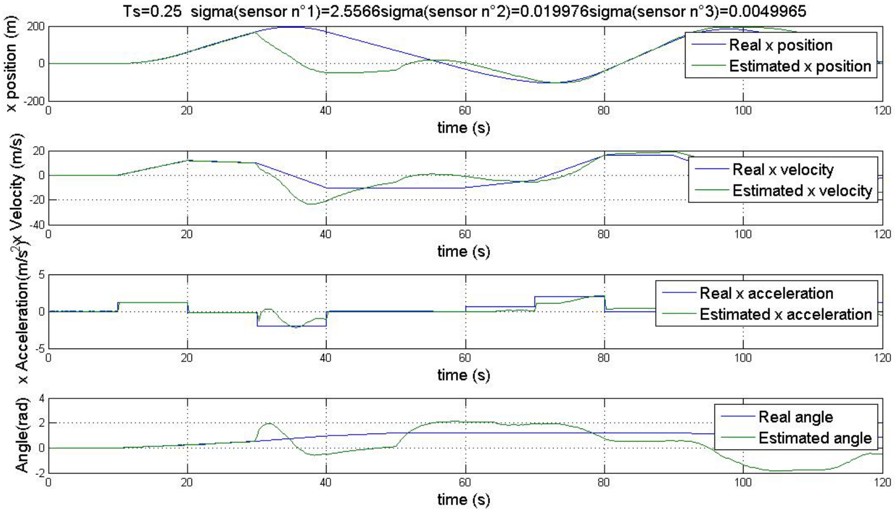


Figure 30 **Real X axis values vs. estimated X axis values.**

In the same way the state estimation in the Y axis is depicted in Figure 31.

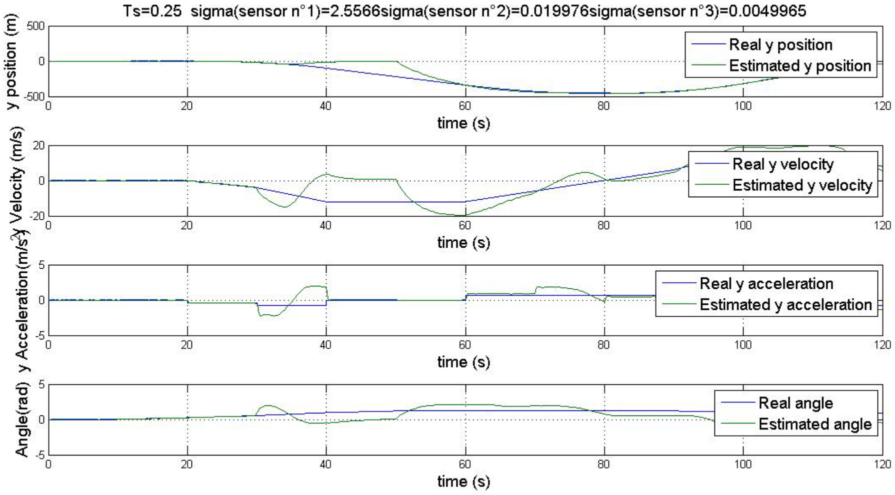


Figure 31 **Real Y axis values vs. estimated Y axis values.**

Observe how the system in the filter is around the 65 seconds giving us an acceptable estimation but during the GPS signal disconnection the error made by the filter is so high.

So we have seen that the fusion between GPS and IMU signals that we have designed will be so sensitive to unusual values.

Instead if we decrease α, we will decrease the velocity of the filter and the sensitivity. That will make that the filter will not be less affected for the unusual value but instead, when the filter starts to receive good values again the filter will take too much time for correcting the trajectory and to reach an acceptable estimation.

In Figure 32 for an $\alpha = {}^{T_{sample}}\!/\!_{100}$, we can observe how the filter still has not arrived to a good estimation in the end of the simulation after to having suffered the loss of the GPS signal.

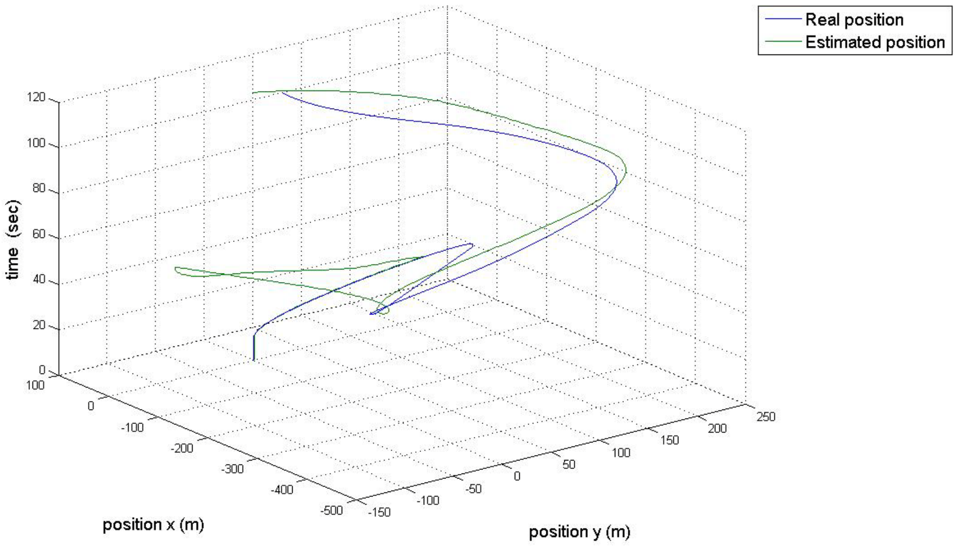


Figure 32 **Real trajectory vs. estimated trajectory.**

An interesting future development opportunity will be to modify the code and to try to implement inside of the Kalman Filter an algorithm able to detect when a strange value is sampled and to ignore it.

For example, the detection of an unusual value could be done calculating the increment between one value and the next one and if we detect an increment so high we will assume that this value is a fake one.

The method explained before will not be the only one we could implement and after we have detected the unusual value, we could take many strategies for trying to minimize the effect that this value will cause in the observer.

5. Navigation system based in physical model

As we explained in the introduction in this point we will design a physical kinematical model that simulates the robot motions when we apply the control action.

The action controls that we could apply in the robot are two. We will be able to control the velocity of the coaxial pair of wheels which the robot has in each link. Besides, we will be able to control the join angle between two consecutive links applying a torque in the motor that we have joining two consecutive links.

The kinematic physical model will give us the trajectory velocity and acceleration of each link in each time step. The first thing that we have to define is the force system that will affect each link that will generate acceleration on it.

Considering that the only external forces that will affect the system will come from the friction that the tires will have with the terrain. We will develop a friction physical model. In our case, this fiction model explains the forces like a force proportional with the tire slipping [4].

Before to deduce the kinematic model equations, we should illustrate the link dimensions and properties because some of these dimensions will appear in the physical model development.

In the Figure 33 we present a simplified sketch about the link aspect.

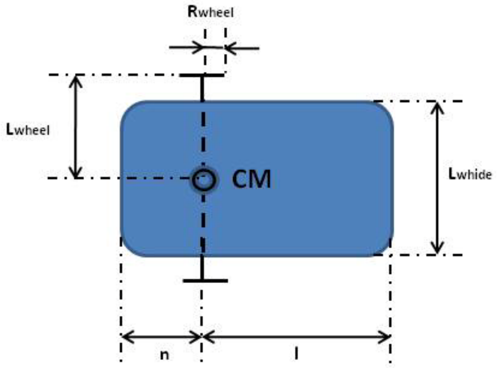
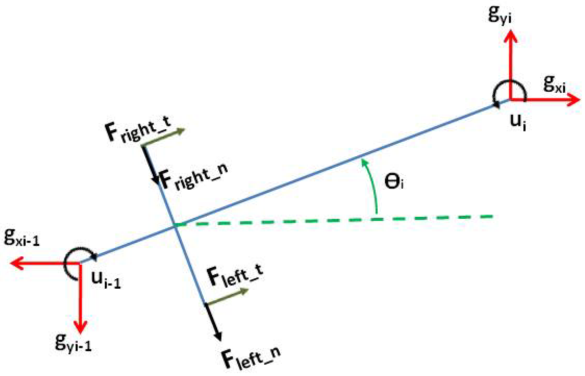


Figure 33 **Link dimensión sketch.**

5.1. *Kinematic model design*

The system development is based in Fukaya, Iwasaki and Saito's multilink kinematic model [11]. Fukaya, Iwasaki and Saito's model has been deduced for a multilink robot where the link center of mass is located in the middle of each link and the friction forces come from the Coulomb frictions forces. In our model we have adapted and extended the Fukaya, Iwasaki and Saito's model for a multilink robot which has not the center of mass located in the middle of the link and the forces come from the friction that appears in the active wheels.

We will begin trying to deduce the equation of motion for a multilink robot with active wheels. The First Newton Law will give us the robot equilibrium equations. The forces in each link are shown in the Figure 34.



Equilibrium equations.

$$\sum f_x = m_{link}\ddot{x}_{CM}$$

$$\sum f_y = m_{link}\ddot{y}_{CM}$$

Figure 34 **Link system of forces.**

In the sketch the g forces represent the cohesion forces or in other words the forces that transmit one link to the neighbor ones. The forces F represent the friction forces in the wheels and the u torques are the torques that we apply for controlling the angle between link and link.

We are going to deduce the equilibrium equations for one example in the case that our robot has only three links and then we will extrapolate the equations for any number of links. This procedure will give us the chance to deduce the physics system equations in an easy manner without loss of generality.

Applying the equilibrium equation we obtain

$$\begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix}\begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \ddot{x}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} g_{x1} \\ g_{x2} \end{bmatrix} + \begin{bmatrix} f_{x1} \\ f_{x2} \\ f_{x3} \end{bmatrix}$$

$$\begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix}\begin{bmatrix} \ddot{y}_1 \\ \ddot{y}_2 \\ \ddot{y}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} g_{y1} \\ g_{y2} \end{bmatrix} + \begin{bmatrix} f_{y1} \\ f_{y2} \\ f_{y3} \end{bmatrix}$$

and defining the following matrices as

$$M := \begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix} \quad X := \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad Y := \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \quad g_x := \begin{bmatrix} g_{x1} \\ g_{x2} \end{bmatrix} \quad g_y := \begin{bmatrix} g_{y1} \\ g_{y2} \end{bmatrix}$$

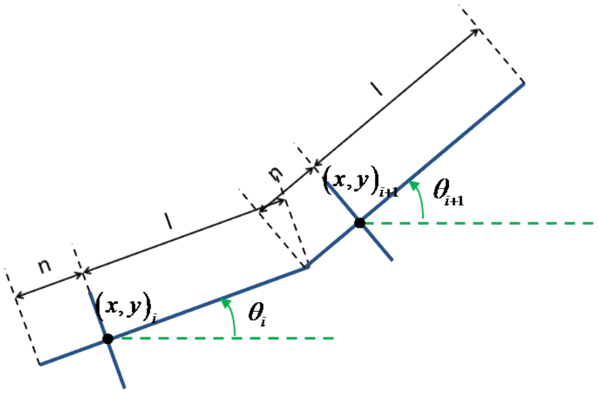
$$f_x := \begin{bmatrix} f_{x1} \\ f_{x2} \\ f_{x3} \end{bmatrix} \quad f_y := \begin{bmatrix} f_{y1} \\ f_{y2} \\ f_{y3} \end{bmatrix} D := \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix}$$

we obtain the next simple expressions:

$$\boxed{\begin{aligned} M\ddot{X} &= D^{'} g_x + f_x \\ M\ddot{Y} &= D^{'} g_y + f_y \end{aligned}} \tag{20}$$

After to have deduced the equations of motion for free link translational motion, we have to write an equation that constrain each link to stay joined to the neighbor ones. For deducing the geometrics equation we have to observe Figure 35.

Geometry equation.



According to the robot geometry it is easy to watch that

$$x_1 - x_2 = n\cos\theta_1 - \left(\left(n+l\right)\cos\theta_1 + n\cos\theta_2\right)$$
$$\Rightarrow x_1 - x_2 = -l\cos\theta_1 - n\cos\theta_2.$$

In the same way

$$y_1 - y_2 = -l\sin\theta_1 - n\sin\theta_2.$$

Figure 35 Link geometry sketch.

For an example with three links, the snake robot geometry equation could be expressed in the following way

$$\begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = -\left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} l & 0 & 0 \\ 0 & l & 0 \\ 0 & 0 & l \end{bmatrix}\begin{bmatrix} \cos\theta_1 \\ \cos\theta_2 \\ \cos\theta_3 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} n & 0 & 0 \\ 0 & n & 0 \\ 0 & 0 & n \end{bmatrix}\begin{bmatrix} \cos\theta_1 \\ \cos\theta_2 \\ \cos\theta_3 \end{bmatrix}\right)$$
$$\begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = -\left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} l & 0 & 0 \\ 0 & l & 0 \\ 0 & 0 & l \end{bmatrix}\begin{bmatrix} \sin\theta_1 \\ \sin\theta_2 \\ \sin\theta_3 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} n & 0 & 0 \\ 0 & n & 0 \\ 0 & 0 & n \end{bmatrix}\begin{bmatrix} \sin\theta_1 \\ \sin\theta_2 \\ \sin\theta_3 \end{bmatrix}\right)$$

Defining the matrices

$$A := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} B := \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} N := \begin{bmatrix} n & 0 & 0 \\ 0 & n & 0 \\ 0 & 0 & n \end{bmatrix} L : \begin{bmatrix} l & 0 & 0 \\ 0 & l & 0 \\ 0 & 0 & l \end{bmatrix} \cos\theta = \begin{bmatrix} \cos\theta_1 \\ \cos\theta_2 \\ \vdots \\ \cos\theta_n \end{bmatrix} \sin\theta = \begin{bmatrix} \sin\theta_1 \\ \sin\theta_2 \\ \vdots \\ \sin\theta_n \end{bmatrix}$$

we could rewrite the previous equations in the next way

$$\begin{aligned} DX &= -\left(\left(AL + BN\right)\cos\theta\right) \\ DY &= -\left(\left(AL + BN\right)\sin\theta\right) \end{aligned} \tag{21}$$

.

Defining $\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$ if we derivate two times (21) we will obtain

$$\begin{aligned} D\dot{X} &= (AL+BN)(S_\theta \dot{\theta}) \Rightarrow \boxed{D\ddot{X} = (AL+BN)\left(C_\theta \dot{\theta}^2 + S_\theta \ddot{\theta}\right)} \\ D\dot{Y} &= -(AL+BN)(C_\theta \dot{\theta}) \Rightarrow \boxed{D\ddot{Y} = (AL+BN)\left(S_\theta \dot{\theta}^2 - C_\theta \ddot{\theta}\right)}. \end{aligned} \tag{22}$$

From equations (20) and (22) we can work out the value of g_x and g_y.

$$\begin{aligned} D\left(M^{-1}D' g_x - M^{-1} f_x\right) &= (AL+BN)\left(C_\theta \dot{\theta}^2 + S_\theta \ddot{\theta}\right) \\ D\left(M^{-1}D' g_y - M^{-1} f_y\right) &= (AL+BN)\left(S_\theta \dot{\theta}^2 - C_\theta \ddot{\theta}\right) \end{aligned}$$

$$\begin{aligned} \Rightarrow &\boxed{g_x = \left(DM^{-1}D'\right)^{-1}\left[(AL+BN)\left(C_\theta \dot{\theta}^2 + S_\theta \ddot{\theta}\right) - DM^{-1} f_x\right]} \\ &\boxed{g_y = \left(DM^{-1}D'\right)^{-1}\left[(AL+BN)\left(S_\theta \dot{\theta}^2 - C_\theta \ddot{\theta}\right) - DM^{-1} f_y\right]} \end{aligned} \tag{23}$$

The equation (23) depends on the θ angles so the next step will be to deduce the link rotation equation that expresses how the angles in the links are affected by the force system.

Rotation equation.

According to the Figure 34 we could write the rotation equations for the links in the following form.

$$\begin{aligned} J_1\ddot{\theta}_1 &= \cos\theta_1 \, l \, \mathrm{g}_{y_1} - \sin\theta_1 \, l \, \mathrm{g}_{x_1} + u_1 + \tau_1 \\ J_2\ddot{\theta}_2 &= \cos\theta_2 \, l \, \mathrm{g}_{y_2} - \sin\theta_2 \, l \, \mathrm{g}_{x_2} + \cos\theta_2 n \, \mathrm{g}_{y_1} - \sin\theta_2 n \, \mathrm{g}_{x_1} + u_2 - u_1 + \tau_2 \\ J_3\ddot{\theta}_3 &= \cos\theta_3 n \, \mathrm{g}_{y_2} - \sin\theta_3 n \, \mathrm{g}_{x_2} - u_2 + \tau_3 \end{aligned}$$

Where $u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$ is the vector which represents the torque we apply between link and the neighbor ones and $\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix}$ the external forces affects in the rotation equation.

Writing this equation with the matrices defined previously the rotation equation will be.

$$\boxed{J\ddot{\theta} = C_\theta(LA^{'} + NB^{'})g_y - S_\theta(LA^{'} + NB^{'})g_x + D^{'}u + \tau} \qquad (24)$$

where J is the moment of inertia matrix.

$$J := \begin{bmatrix} J_1 & 0 & 0 \\ 0 & J_2 & 0 \\ 0 & 0 & J_3 \end{bmatrix}.$$

Substituting (23) into(24), we will obtain

$$J\ddot{\theta} = C_\theta\left(AL + BN\right)^{'}\left(DM^{-1}D^{'}\right)^{-1}\left[\left(AL + BN\right)\left(S_\theta\dot{\theta}^2 - C_\theta\ddot{\theta}\right) - DM^{-1}f_y\right] -$$
$$-S_\theta\left(AL + BN\right)^{'}\left(DM^{-1}D^{'}\right)^{-1}\left[\left(AL + BN\right)\left(C_\theta\dot{\theta}^2 + S_\theta\ddot{\theta}\right) - DM^{-1}f_x\right] + D^{'}u + \tau$$

and grouping the previous equation, we obtain the following expression.

$$\left[J + C_\theta(AL + BN)^{'}(DM^{-1}D^{'})^{-1}(AL + BN)C_\theta + S_\theta(AL + BN)^{'}(DM^{-1}D^{'})^{-1}(AL + BN)S_\theta\right]\ddot{\theta} =$$
$$= -\left[S_\theta(Al + BN)^{'}(DM^{-1}D^{'})^{-1}(AL + BN)C_\theta - C_\theta(AL + BN)^{'}(DM^{-1}D^{'})^{-1}(AL + BN)S_\theta\right]\dot{\theta}^2 -$$
$$C_\theta(AL + BN)^{'}(DM^{-1}D^{'})^{-1}DM^{-1}f_y + S_\theta(AL + BN)^{'}(DM^{-1}D^{'})^{-1}DM^{-1}f_x + D^{'}u + \tau$$

For simplifying the equation we will define the next matrix that we will use for simplifying the previous equation.

$$H := \left(AL + BN\right)^{'}\left(DM^{-1}D^{'}\right)^{-1}\left(AL + BN\right)$$
$$\mathcal{J} := J + C_\theta H C_\theta + S_\theta H S_\theta$$
$$\mathcal{C} := S_\theta H C_\theta - C_\theta H S_\theta$$
$$K := \left(AL + BN\right)^{'}\left(DM^{-1}D^{'}\right)^{-1}DM^{-1}$$

With the matrix dined previously we can define a new mat like

$$L = [S_\theta K \quad -C_\theta K]'$$

and define a new vector as $f = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$.

So with the variables previously defined we could express the rotation equation as

$$\boxed{J\ddot{\theta} = -\mathcal{C}\dot{\theta}^2 + D'u + L'f + \tau} \qquad (25)$$

Center of mass equation

We want to express the motion equations in two decoupled equations referenced to the center of mass. One will be the rotation equation around each link mass center and the other will be a translacion equation that we will deduce it later on.

The center of mass X coordinate equation is defined as

$$W_x = \frac{x_1 m_1 + x_2 m_2 + x_3 m_3}{m_1 + m_2 + m_3}$$

$$W_x = \frac{1}{m_1 + m_2 + m_3}\begin{bmatrix}1 & 1 & 1\end{bmatrix}\begin{bmatrix}m_1 & 0 & 0\\ 0 & m_2 & 0\\ 0 & 0 & m_3\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}$$

and in the same way the Y mass equation is defined as the .

$$W_y = \frac{y_1 m_1 + y_2 m_2 + y_3 m_3}{m_1 + m_2 + m_3}$$

$$W_y = \frac{1}{m_1 + m_2 + m_3}\begin{bmatrix}1 & 1 & 1\end{bmatrix}\begin{bmatrix}m_1 & 0 & 0\\ 0 & m_2 & 0\\ 0 & 0 & m_3\end{bmatrix}\begin{bmatrix}y_1\\ y_2\\ y_3\end{bmatrix}.$$

If we define $m := m_1 + m_2 + m_3$ and $e := \begin{bmatrix}1 & 1 & 1\end{bmatrix}'$ we could rewrite the previous equations as

$$E = \begin{bmatrix} & 0\\ e & 0\\ & 0\\ 0 & \\ 0 & e\\ 0 & \end{bmatrix} \qquad W := \begin{bmatrix}W_x\\ W_y\end{bmatrix} = \frac{1}{m}E'\begin{bmatrix}MX\\ MY\end{bmatrix} \qquad (26)$$

Derivating two times the (26) equation, we will obtain

$$\dot{W} = \frac{1}{m}E'\begin{bmatrix} M\dot{X} \\ M\dot{Y} \end{bmatrix} \quad \Rightarrow \quad \ddot{W} = \frac{1}{m}E'\begin{bmatrix} M\ddot{X} \\ M\ddot{Y} \end{bmatrix}$$

substituting with the equation (20) the center of mass equation is expressed in the following form.

$$\ddot{W} = \frac{1}{m}E'\begin{bmatrix} M\ddot{X} \\ M\ddot{Y} \end{bmatrix} = \frac{1}{m}E'\begin{bmatrix} D'g_x + fx \\ D'g_y + fy \end{bmatrix}.$$

Observing that $e'D' = 0_{1x2}$ we will rewrite the dynamic equation for the mass center in the following simply equation.

$$m\ddot{W} = E'\begin{bmatrix} f_x \\ f_y \end{bmatrix} \Rightarrow \boxed{m\ddot{W} = E'f} \tag{27}$$

Link motion equation

From (22) and (27) equations we could fusion both expressions getting a equation for the center of mass translation.

$$TX = \begin{bmatrix} -\left(AL + BN\right)\cos\theta \\ W_x \end{bmatrix} \quad \& \quad TY = \begin{bmatrix} -\left(AL + BN\right)\sin\theta \\ W_y \end{bmatrix} \; where \;\; T := \begin{bmatrix} D \\ e' M/m \end{bmatrix}$$

$$\Rightarrow X = T^{-1}\begin{bmatrix} -\left(AL + BN\right)\cos\theta \\ W_x \end{bmatrix} \quad \& \quad Y = T^{-1}\begin{bmatrix} -\left(AL + BN\right)\sin\theta \\ W_y \end{bmatrix}$$

$$T^{-1} = \begin{bmatrix} M^{-1}D'\left(DM^{-1}D'\right)^{-1} & e \end{bmatrix}$$

Finally, defining the position vector like $z := \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} X \\ Y \end{bmatrix}$ we are able to get the link center of mass equation with the next equation.

$$\boxed{\dot{z} = \mathcal{L}\dot{\theta} + E\dot{W}} \tag{28}$$

Summarizing, the robot motion is defined for the (25) and (28) equations which are independent of the external force system we have chosen for the system.

As we said in the introduction of this chapter, the external forces will derive from the friction forces system between the tire and the terrain in the wheels. According to we said, we will have to design our friction equations.

Friction forces

We mentioned that in our model in contrast to Iwasaki and Saito's multilink kinematic model [11], the friction forces will be generated in the wheels proportionally to the slipping between the tire and the terrain. One point to consider will be that the slipping measurement will be observed by the difference between the longitudinal translacion velocity in the tire and the tire revolution speed.

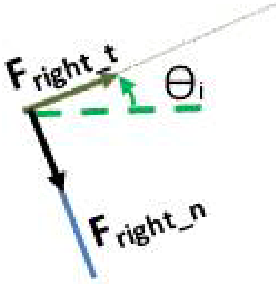


Figure 36 Friction forces in right wheels.

As we can observe in Figure 36, each wheel will be affected by two forces. The tangential forces will come from the wheel spinning and the normal forces will come from Coulomb friction forces.

According to the previous paragraph the normal friction forces will be defined as

$$\left|F_{right\_n}\right| = \left|F_{left\_n}\right| = \mu \cdot m_{link} \cdot g \tag{29}$$

where μ is the coefficient of friction, m_{link} is the link mass and g is the gravity acceleration. The direction of the force will be the opposite of the normal velocity in the wheels V_{norm}.

The definition of the tangential forces will be more complex. The forces are generated due to the friction between the wheel spinning and the terrain. They will be defined as a force proportional to the wheel slipping [4]

$$F_{wheel\_t} = \varphi \frac{V_{long\_wheel} - \omega_{wheel} R_{wheel}}{\left| V_{long\_wheel} \right|} \; . \tag{30}$$

In the previous equation, V_{long_wheel} represents the translation velocity in the wheel where the force is generated and will be expressed in $\left[m/s \right]$, ω_{wheel} is the angular velocity in $\left[rad/s \right]$, R_{wheel} is the wheel radius in $\left[m \right]$and φ will be the force due to Coulomb friction measured in $\left[N \right]$.

Observe that the model gives us the center mass velocity and we will need to calculate the velocity in the wheels according to the relative movement equations.

Considering that the wheel velocities will not be equal to the velocity of the body and that for calculating the longitudinal velocity we first have to translate the frame of reference and then we should apply a rotation matrix and take the x velocity component. This component will be the longitudinal velocity.

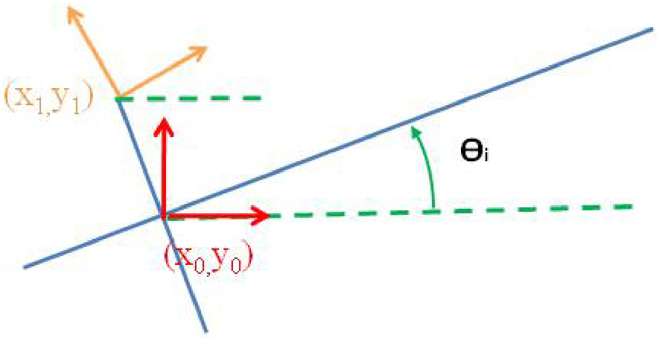


Figure 37 Different coordinate systems in the links.

Observe in Figure 37 the two frames of reference used during the mathematical devlopment. The model equations give us the $\dot{z}$ vector which is expressed in the $\left(x_0, y_0 \right)$ frame of reference. The longitudinal velocity is the X velocity component expressed in the $\left(x_1, y_1 \right)$ frame of reference and the normal velocity will be the Y velocity component.

If we express the previous sentence in a vectorial equation we have

$$\vec{V}_{wheel(x_1,y_1)} = \underbrace{\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}}_{\text{rotation matrix}} \left(\vec{V}_{body(x_0,y_0)} + \vec{l}_{wheel} \times \vec{\dot{\theta}} \right)$$

in our particular case this equation is expressed like

$$\vec{V}_{wheel\_right(x_1,y_1)} = \begin{bmatrix} V_{long\_right} \\ V_{norm\_right} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} V_{body\_x} \\ V_{body\_y} \end{bmatrix} + \begin{bmatrix} -\dot{\theta} l_{wheel} \\ 0 \end{bmatrix}$$

$$\vec{V}_{wheel\_left(x_1,y_1)} = \begin{bmatrix} V_{long\_left} \\ V_{norm\_left} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} V_{body\_x} \\ V_{body\_y} \end{bmatrix} + \begin{bmatrix} \dot{\theta} l_{wheel} \\ 0 \end{bmatrix} \tag{31}$$

Analyzing (31), it is easy to deduce that when the link movement is a straight line, the angle velocity $\dot{\theta}$ is zero so the longitudinal velocity in both wheels is the same and the forces that affect to each wheel is the same.

As it is mentioned in.[4] the expression of the forces shown in the equation (30) will not work when the tire slipping is higher than ±10% and then this expression is affected by a saturation effect.

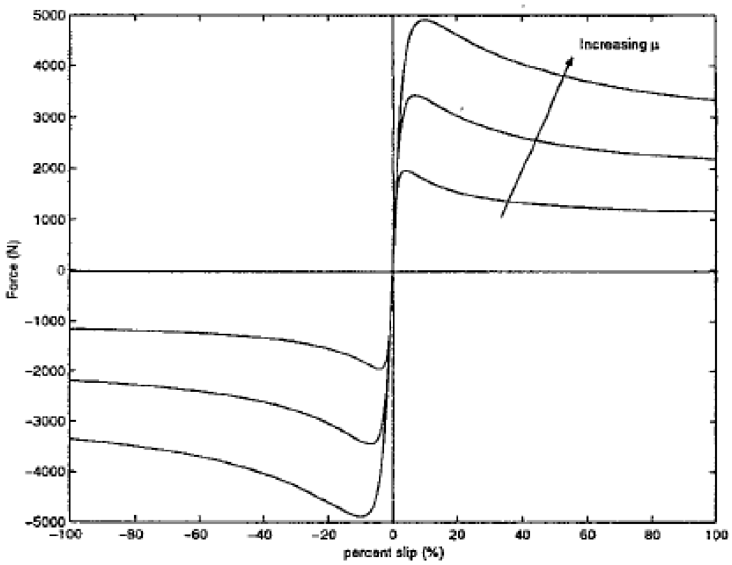


Figure 38 **Forces saturation diagram. [4]**

Substituting the equation (31) in equation (30) we can arrive to the following expression for the forces.

$$\begin{aligned} F_{left\_t} &= -\varphi \frac{\omega_{wheel} R_{wheel} - \left([C_\theta \ S_\theta] \dot{z} + \dot{\theta} l_{wheel} \right)}{\left\| [C_\theta \ S_\theta] \dot{z} + \dot{\theta} l_{wheel} \right\|} \\ F_{right\_t} &= -\varphi \frac{\omega_{wheel} R_{wheel} - \left([C_\theta \ S_\theta] \dot{z} - \dot{\theta} l_{wheel} \right)}{\left\| [C_\theta \ S_\theta] \dot{z} - \dot{\theta} l_{wheel} \right\|} \end{aligned} \tag{32}$$

Defined the forces expressions in the wheels we have to deduce how the external forces affect the link rotation equation (25). Remembering that the rotation equation is

$$\mathcal{J} \ddot{\theta} = -\mathcal{C} \dot{\theta}^2 + D' u + L' f + \tau$$

where still f and τ are not defined.

If f is defined as $f = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$, the first step will be to define f_x and f_y in relation to F_{left} and F_{right}. Observing Figure 34, the forces definitions are the next one.

$$\begin{aligned} f_x &= C_\theta \left[F_{left\_t} + F_{right\_t} \right] + S_\theta \left[F_{left\_n} + F_{right\_n} \right] \\ f_y &= S_\theta \left[F_{left\_t} + F_{right\_t} \right] - C_\theta \left[F_{left\_n} + F_{right\_n} \right] \end{aligned}. \tag{33}$$

Observing Figure 34 we can deduce the value of. τ As we said, τ is the direct effect that the friction forces in the wheels generate in the rotation equation (25) generation a torque in the center of mass. Observe in Figure 34 that each force generates one torque in opposite directions with respect the other. So the effect of the external force is

$$\tau = F_{left\_t} l_{wheel} - F_{right\_t} l_{wheel}$$

substituting the forces values from (32) we will be able to calculus the τ value.

As it is said at the beginning of this chapter, the equations deduction was done for a three links robot trying to make easiest for the reader to understand the equation development.

In this point we will show the expression for a n links robot. If we want to express the equations for n links, we will have to redefine the matrix that we have use in the previous deduction

Equilibrium equations (n links robot).

The equilibrium equation aspect will not be different to the equation of motion (20), but we have to redefine each matrix that appears in the equations. Remember that the equilibrium equations have this form

$$M\ddot{X} = D^{'} g_x + f_x$$
$$M\ddot{Y} = D^{'} g_y + f_y$$

where for the n link robot the matrices will be redefined as

$$M = \begin{bmatrix} m_1 & 0 & \cdots & 0 \\ 0 & m_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & m_n \end{bmatrix} \quad D = \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 & -1 \end{bmatrix} \left[n-1, n\right]$$

$$\ddot{X} = \begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \vdots \\ \ddot{x}_n \end{bmatrix} \ddot{Y} = \begin{bmatrix} \ddot{y}_1 \\ \ddot{y}_2 \\ \vdots \\ \ddot{y}_n \end{bmatrix} f_x = \begin{bmatrix} f_{x_1} \\ f_{x_2} \\ \vdots \\ f_{x_n} \end{bmatrix} f_y = \begin{bmatrix} f_{y_1} \\ f_{y_2} \\ \vdots \\ f_{y_n} \end{bmatrix} g_x = \begin{bmatrix} g_{x_1} \\ g_{x_2} \\ \vdots \\ g_{x_{n-1}} \end{bmatrix} g_y = \begin{bmatrix} g_{y_1} \\ g_{y_2} \\ \vdots \\ g_{y_{n-1}} \end{bmatrix}.$$

Geometry equation (n links robot).

The geometry equation and its derivates had the following aspect

$$DX = -\left(\left(AL + BN\right)\cos\theta\right) \xrightarrow{d/dt} D\dot{X} = \left(AL + BN\right)\left(S_\theta \dot{\theta}\right) \xrightarrow{d/dt} D\ddot{X} = \left(AL + BN\right)\left(C_\theta \dot{\theta}^2 + S_\theta \ddot{\theta}\right)$$
$$DY = -\left(\left(AL + BN\right)\sin\theta\right) \xrightarrow{d/dt} D\dot{Y} = -\left(AL + BN\right)\left(C_\theta \dot{\theta}\right) \xrightarrow{d/dt} D\ddot{Y} = \left(AL + BN\right)\left(S_\theta \dot{\theta}^2 - C_\theta \ddot{\theta}\right)$$

where we will rewrite the matrix as

$$A = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & & & \vdots \\ \vdots & & \ddots & & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix} \left[n-1, n\right] B = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & & \vdots \\ \vdots & & & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix} \left[n-1, n\right] \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \dot{\theta} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \dot{\theta}_n \end{bmatrix}$$

$$\ddot{\theta} = \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \vdots \\ \ddot{\theta}_n \end{bmatrix} L = \begin{bmatrix} l_1 & 0 & \cdots & 0 \\ 0 & l_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & l_n \end{bmatrix} N = \begin{bmatrix} n_1 & 0 & \cdots & 0 \\ 0 & n_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & n_n \end{bmatrix} \cos\theta = \begin{bmatrix} \cos\theta_1 \\ \cos\theta_2 \\ \vdots \\ \cos\theta_n \end{bmatrix} \sin\theta = \begin{bmatrix} \sin\theta_1 \\ \sin\theta_2 \\ \vdots \\ \sin\theta_n \end{bmatrix}$$

Rotation equation (n links robot).

The final rotation equation was defined as

$$\mathcal{J}\ddot{\theta} = -\mathcal{C}\dot{\theta}^2 + D'u + L'f + \tau$$

where the matrices defined previously were

$$\begin{aligned}
&H := (AL+BN)'\left(DM^{-1}D'\right)^{-1}(AL+BN)\\
&\mathcal{J} := J + C_\theta H C_\theta + S_\theta H S_\theta\\
&\mathcal{C} := S_\theta H C_\theta - C_\theta H S_\theta\\
&K := (AL+BN)'\left(DM^{-1}D'\right)^{-1} DM^{-1}\\
&L = [S_\theta K \quad -C_\theta K]' \qquad .
\end{aligned}$$

So we have to define the previous matrices for n links robot.

$$J = \begin{bmatrix} J_1 & 0 & \cdots & 0 \\ 0 & J_2 & & \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & J_n \end{bmatrix} C_\theta = \begin{bmatrix} \cos\theta_1 & 0 & \cdots & 0 \\ 0 & \cos\theta_2 & & \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \cos\theta_n \end{bmatrix} S_\theta = \begin{bmatrix} \sin\theta_1 & 0 & \cdots & 0 \\ 0 & \sin\theta_2 & & \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \sin\theta_n \end{bmatrix}$$

Mass center equation (n links robot).

The mass center equation had the following expression

$$m\ddot{W} = E' f$$

where the matrices E, f and the m variable are rewritten as

$$m = m_1 + m_2 + \cdots + m_n \qquad e = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}[n,1] \qquad E = \begin{bmatrix} & 0 \\ e & \vdots \\ & 0 \\ 0 & \\ \vdots & e \\ 0 & \end{bmatrix}[2n,2].$$

Link motion equation (n links robot).

The link motion equation when we have a n links robot. is the one expressed in the equation (28).

$$\dot{z} = \mathcal{L}\dot{\theta} + E\dot{W}$$

In this case it is not necessary to redefine any new matrix because all of them have been defined before.

Defined the equations for i links the shown simulations a posteriori will be done for a robot with six link robot.

5.2. Link angle controller

After developing the robot physical model for n number of links and if we now simulate the robot trajectory without any controller acting in the link angle, in the moment the robot has not every of its links aligned an oscillating and unstable behavior will appear. We illustrate this behavior in Figure 39.

The simulation in Figure 39 has been done for a n=6 links applying a constant velocity in the wheels and from the second 9 to second 13 we have applied a torque in one of the robot joins.

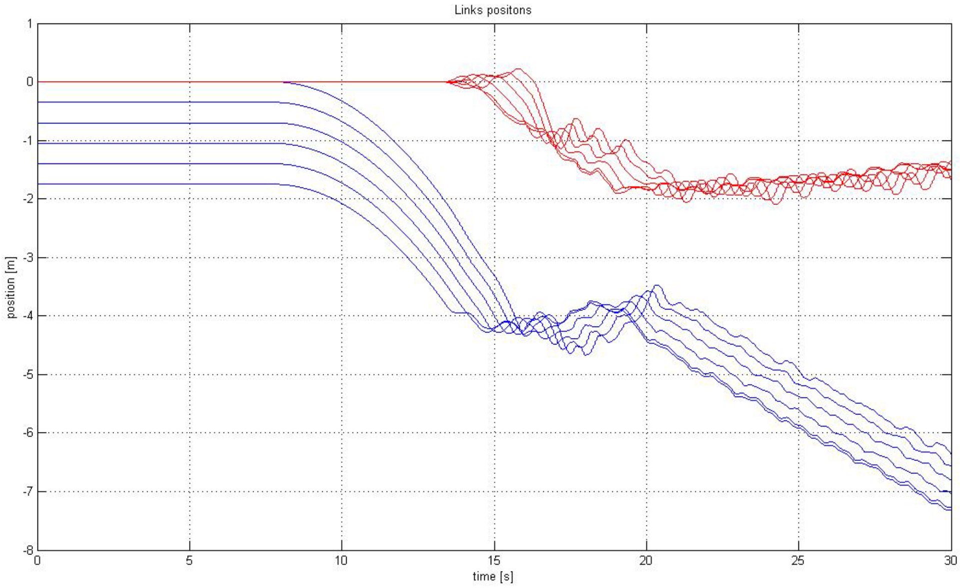


Figure 39 **Links position without link angle controller.**

In Figure 39 we can observe the positions of the links over the time. The X position and the Y position are plotted in the previous chart in blue color and red color respectively. It is easy to observe how the robot was stable. But around the second 13 a torque in the join was applied and the robot started to have an oscillating behavior, mostly in the Y position, due to there is no control applied in the link angle.

For a link angle controller development we have to observe how the entrance affects the angle acceleration equation. In the angle acceleration equation (25), the input u, which represents the joint torque, is multiplied per D'. That matrix multiplication entails that each n join torque affects the i and i+1 links angle.

For controlling the link angle, we have designed a simply link angle controller which controls the link robot angle in the equilibrium point. The controller is based on a PD controller (proportional, derivative) which is able to turn in a corner and continue straight in a linear trajectory.

In Figure 40 is easily observable how the angle controller controls the robot trajectory and the robot executes a kink of 90 deg. That could be a good example if the robot is working inside a pipe.

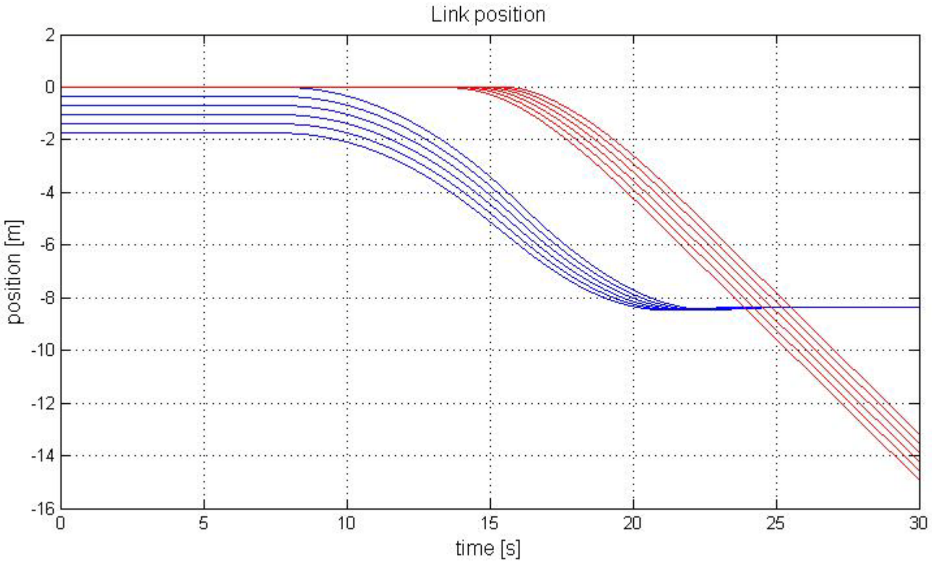


Figure 40 **Link position with link angle controller.**

Another time, in Figure 40 the X position and the Y position are plotted in blue and red color respectively.

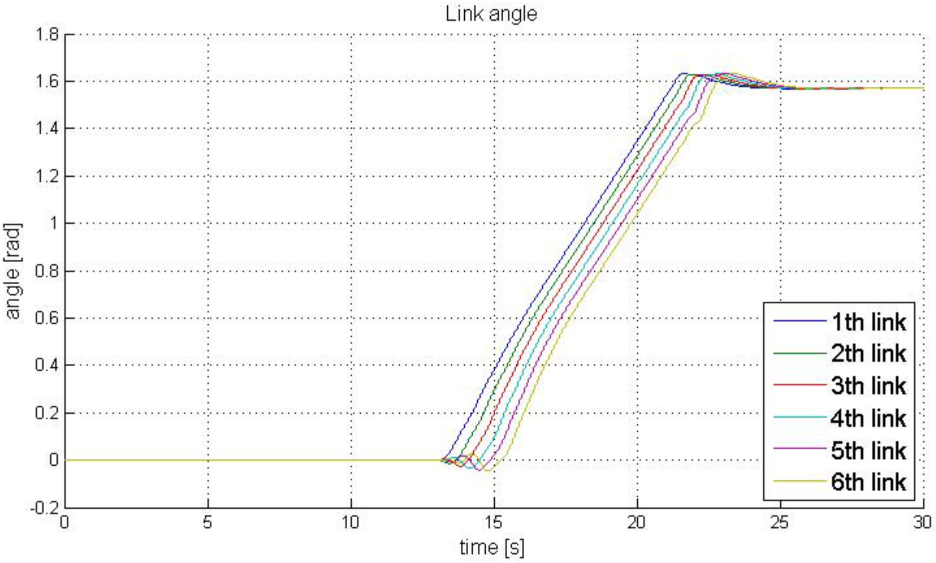


Figure 41 **Link angle.**

In Figure 41 we can observe how the controller stabilizes the links angles progressively starting by the first one and finally aligning all of them.

5.3. Physic model and inertial sensor fusion

Explained the physical model that we have deduced in Chapter 5.1, in this chapter we will observe the different errors suffered by the physical model due to the assumed simplification done during the design. Subsequently we will implement or at least analyze some possible techniques for improving the position estimation given by the designed physical model.

As it is said in the introduction, the error sources come mostly from the simplification done during the modelization of several physical effects. In our case the most important errors sources will come from the coefficient of friction and if a robot link looses contact with the terrain and the wheels are spinning in the air.

In the case of the coefficient of friction, the error comes from the assumption that this coefficient is a constant in the terrain. In the model this constant is an averaged value typically of each terrain material but it does not mean that in reality the coefficient is constant over the time.

Our first goal will be to observe what kind of error causes these simplifications in the designed observer and try to obtain a better estimation with the information which will be given by the inertial sensor the robot has installed. Because of the strong system nonlinearity the implementation of a Kalman Filter will be difficult.

Due to we did not have access to empirical data, we had to look for another way for checking the quality of the developed model and analyzing the committed error caused by the simplifications done in the model. For this propose, we created a model which will be called real physical model where we emulated these unexpected several random error sources. This model will be compared to another one which we have simulated without these error sources. This last model will be called ideal physical model. These names will be used during the rest of this thesis.

Towards a better understanding of the committed errors in the navigation system based on the ideal physical model developed previously, we will simulate different real situations and observe how the ideal physical model will estimate the trajectory in compassion to the trajectory given by the real physical model.

5.3.1. Error caused by wheel traction loss

In the first error analysis, we will simulate a real situation as a situation where a couple of wheels of a robot link are spinning in the air and this link, during a period time, has no traction with the terrain.

As we did in previous chapters, we have simulated a trajectory where the robot executes a kink of 90 deg. In the following graph we will observe the trajectory of each link in a XY graph which will be more intuitive than to show both components over the time as we did in Figure 40.

The traction losing has been simulated just in one link during the 9th and 16th second when the robot is moving with all its links aligned. During that period of time, for the real physical model, the first link lost completely the contact with the terrain and in it wheels did not appear any friction forces. During this example in both physical models it is supposed that the coefficient of friction is a constant value.

In Figure 42 we can observe the trajectory followed by the robot. If we want to difference the trajectory of each link we should zoom into the graph.

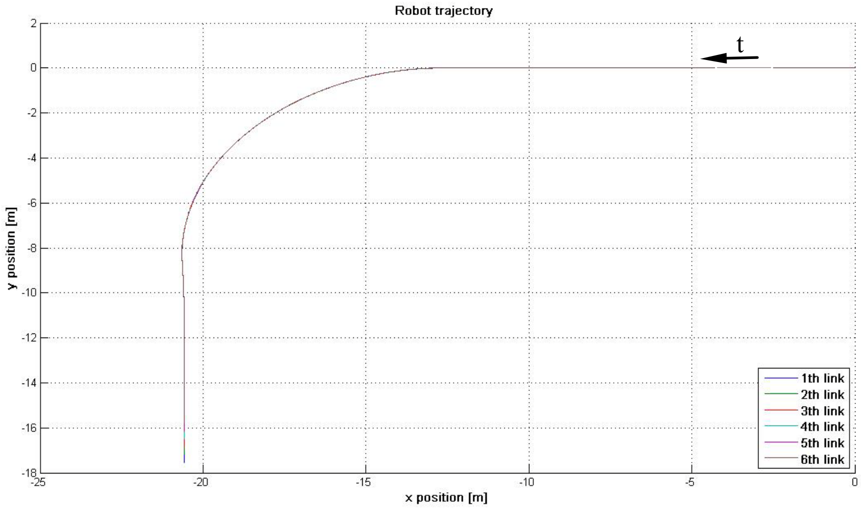
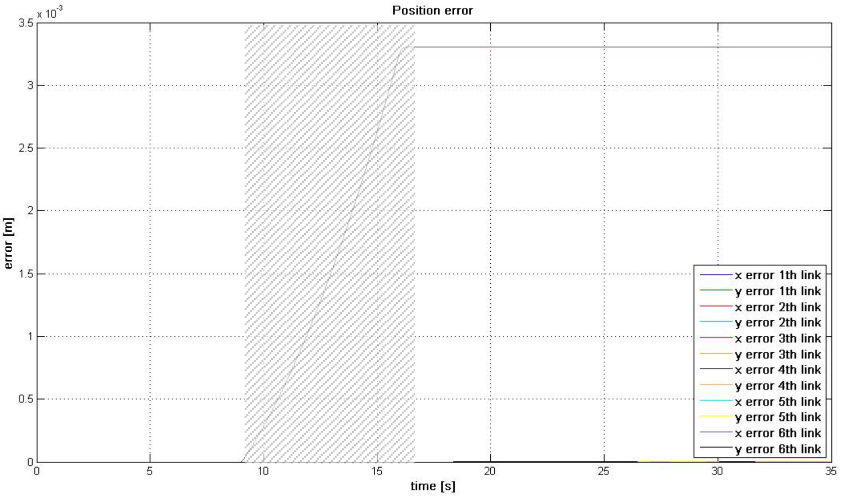


Figure 42 Robot trajectory.

The trajectory obtained by the ideal model which is working as our trajectory observer will not be shown because the difference between the real trajectory graph and the one obtained by the ideal model is not perceptible watchable.

Instead of posting the estimated trajectory in a XY graph where it is difficult to appreciate the error between the real trajectory and the estimated trajectory, we will show how the error between the real and ideal trajectory evolves over the time and analyze the effect of this traction loss



.

Figure 43 **Position error. Traction loss case.**

In Figure 43, we can observe the error between the real and the estimated trajectory for each component. It is impossible to see the difference in each link because they suffer almost the same error on each component so due to the line overlap we just observe a gray line which represents the X component error and an orange line which represents the Y component error. It is easily to observe how during the traction loss time (shady area) the error increments and after that the error is stable in the reached value.

The results shown before could be checked executing the Matlab script called *traction_lost_model.m* attached to this thesis.

5.3.2. Error caused by heterogeneous coefficient of friction

In this point our real case will be a simulation where the coefficients of friction will not be a homogeneous averaged constant as in the ideal physical model. In the ideal physical model (estimator) the coefficient value has been fixed in $\mu = 0.5$ and instead of the real physical model we have defined the coefficient of friction as $\mu = 0.5 + w$, where w is a white noise which we have fixed with the amplitude of ±0.1 trying to emulate the nature.

The trajectory will be almost the same like the one which is shown in Figure 42. The trajectory estimated by the ideal estimator is similar to the real trajectory so it has no sense to plot the trajectory because we could not observe the difference. Instead, another time, we will plot the error per component between the real trajectory and the estimated by the ideal physical model.

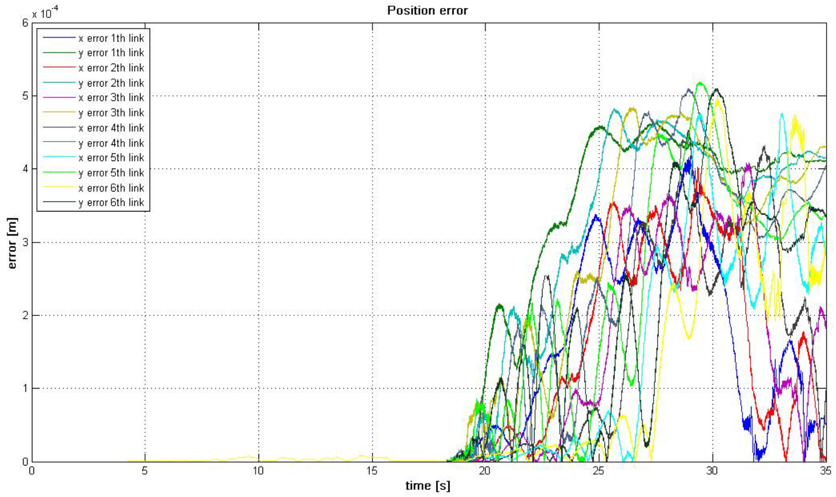


Figure 44 **Position error. Heterogeneous coefficient of friction case.**

Observe that the estimation is quite good and the error has an order of 10^{-4} but the error is accumulative and will be increased step by step to higher orders.

The obtained results shown previously could be checked executing the Matlab script called *het_coeff_model.m* attached to this thesis.

5.4. *Physical model/inertial sensor fusion*

After analyzing in Chapter 5.1 two further probable errors, which could be suffered by the ideal physical model, we will study how the fusion between the physical model and the inertial sensor works.

5.4.1. *Real case simulation*

First of all we have to simulate a real case for comparing the estimation given by the ideal model. Then we will compare it with the error committed by the one that we obtain from applying a fusion technique and in this way we will study if the implement of this technique is worthwhile.

Due to we have no empirical data of the robot running, we have to create a simulation which could be compared with an emulation of a real situation. In our case this simulation has been introduced in Chapter 5.3. The heterogeneous coefficient of friction will be simulated by introducing a white noise in the averaged value which we have estimated in the ideal physical model. The loss traction effect will be simulated during the whole simulation by decreasing the coefficient of friction in some link wheels intermittently to 0. So in other words, it is to say that while the robot is working, there are some moments in which the links wheels have no traction with the terrain where they are turning.

First of all we have to define the robot characteristics of the multilink robot physical model defined in Chapter 5.1. The characteristics that have been defined for the robot in the simulation will be the following ones.

Robot properties.

$$n = 0.15\ m \qquad l = 0.2\ m \qquad l_{wheel} = 0.08\ m$$
$$l_{wide} = 0.10\ m \qquad R_{wheel} = 0.05\ m \qquad m_{link} = 1.2\ Kg$$
$$J = 2.18 \cdot 10^{-3}\ Kg \cdot m^2 \qquad \mu = 0.5$$

These robot characteristics are defined in the simulation in the Matlab script called *modelini.m* attached to this thesis.

Another time the robot trajectory will be the one where the robot turns a kink of 90 degrees. The obtained trajectory is shown in Figure 45.

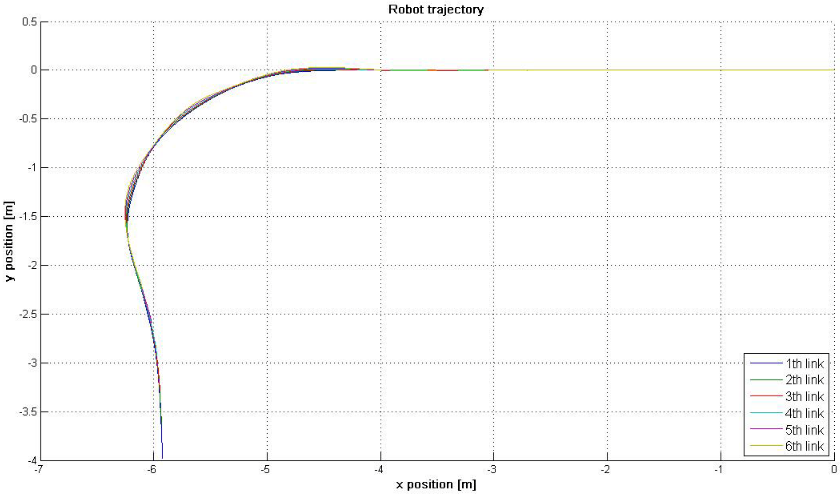


Figure 45 **Robot real trajectory.**

As we said before, the real trajectory shown in Figure 45 has been simulated by introducing some error sources explained in Chapter 5.3. Figure 46 represents the traction loss suffered in each link.

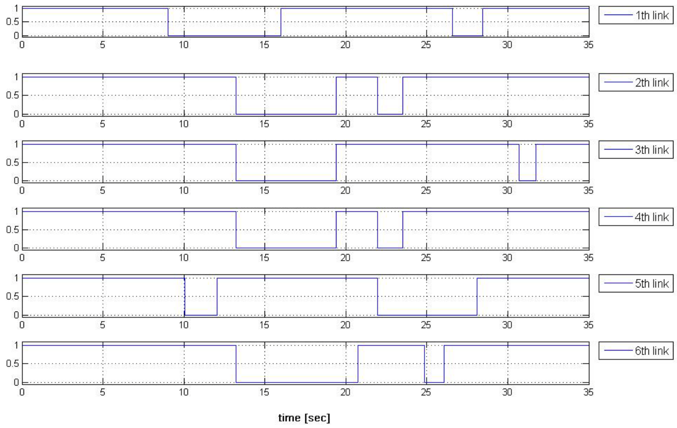


Figure 46 **Wheels traction loss**

In Figure 46 it is shown the time periods where the link wheels are losing contact with the terrain. When the line has a value of 1 the wheels in this link have a perfect contact with the terrain and when the line has a value of 0, the wheels in the link will be spinning in the air.

The second source of error will be the non constant or heterogeneous coefficient of friction. In our simulation we have supposed a coefficient of friction $\mu = 0.5 + w$ where w is a white noise with an amplitude of 0.2.

The estimated trajectory of the ideal model where we have not considered any traction loss period and a coefficient of friction will be presented in Figure 47.

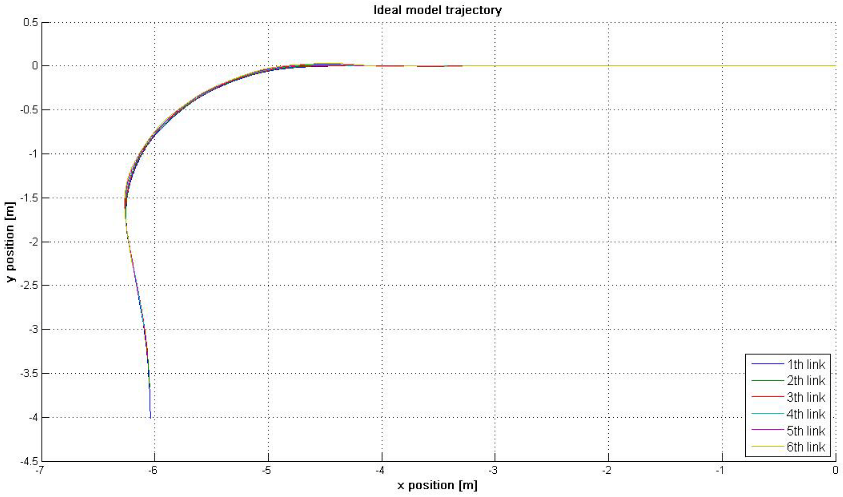


Figure 47 **Trajectory given by the ideal physical model.**

In Figure 47 is not easy to difference the error between the link trajectories given by the ideal physical model and the real trajectory shown in Figure 45. To illustrate this error we can observe Figure 48 where it is plot the error for each link component.

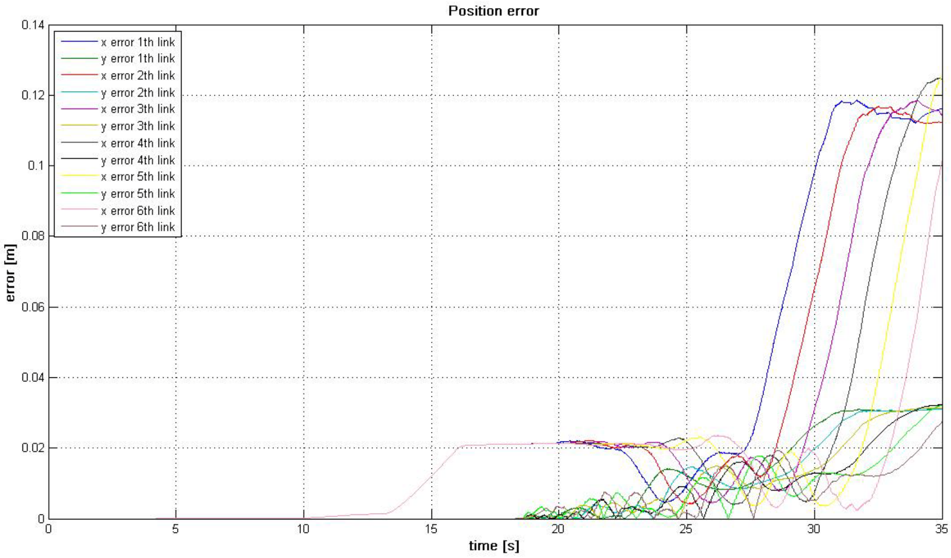


Figure 48 **Error between the real trajectory and the one given by the ideal model.**

Observe that the ideal estimator will not suffer a large error but the error will be accumulative over the time. As long as the ideal model is navigation the system the error will increase more and more. Anyway in this simulation the committed error for the ideal physical model navigation the trajectory has an order of centimeters so, that it will be difficult to reach a most accurate navigation system by the fusion of the ideal physical model trajectory and the inertial sensor measurements.

5.4.2. Fusion implementation

As we said in the introduction, this point will attempt to fusion the data which come from the ideal physical model, the inertial sensors and the angle information from the gyroscope. The ideal physical model will give us the position estimated with all the simplifications that we have supposed during the design, meanwhile the information that the sensor and gyroscope one will give us will be from the real acceleration and angle velocity that is acting in each link but these measurements will be affected by a white noise characteristic of each sensor.

Another time, the technique used in this thesis will be the Kalman Filter estimator. Due to the sensor will be located in each link, the acceleration measurement given for this sensor will not be according to the absolute frame of reference and these measurements will be referenced with the relative frame of reference that the IMU (Inertial Measurement Unit) has. These relative measurements will make the system nonlinear ergo we will use an Extended Kalman Filter (EKF) fusion technique for which the mathematical theory is explained in Chapter 3.2. Illustrating these differences which exist in the frames of reference we have in Figure 17 which enables the reader to understand the case easiest.

Our system that we will define for implementing the Extended Kalman Filter fusion will be equal to the one used in Chapter 4.4.

$$x(t)=\begin{bmatrix} x_{pos}(t) \\ y_{pos}(t) \\ v_x(t) \\ v_y(t) \\ a_x(t) \\ a_y(t) \\ \theta(t) \\ \dot{\theta}(t) \end{bmatrix} \Rightarrow \dot{x}(t)=\begin{bmatrix} 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0 \end{bmatrix} x(t)+w(t) \tag{34}$$

The output vector which will represent the sampled measurements will be

$$z(t)=\begin{bmatrix} z_1(t) \\ z_2(t) \\ z_3(t) \\ z_4(t) \\ z_5(t) \end{bmatrix} \Rightarrow \begin{cases} \text{x ideal model position estimation.} \\ \text{y ideal model position estimation.} \\ \text{x IMU sensor acceleration measurement.} \\ \text{y IMU sensor acceleration measurement.} \\ \theta \text{ IMU sensor velocity measurement.} \end{cases} \tag{35}$$

being the output equation

$$z(t)=\begin{bmatrix} x_{pos} \\ y_{pos} \\ \cos(\theta)a_x+\sin(\theta)a_y \\ -\sin(x_7)a_x+\cos(x_7)a_y \\ \dot{\theta} \end{bmatrix}+v(t). \tag{36}$$

The definition of the Q and R matrices will be the same as in Chapter 4.4. One important point will be to tune the filter selecting the noise in each measurement which we defined in the filter. In the inertial sensor noise, the noise will be defined according to the proprieties of the sensors. As we did in the Chapter 4.4 the error in the acceleration measurements will be a white noise of $0.02\, m/s^2$ In the case of the ideal physical model, as big value we select for the position noise less credibility the data will have into the filter.

The real trajectory will continue being the one shown in the Figure 48 and the ideal model estimation will be the one presented in Figure 51. Remember that the difference between both trajectories has been explained in Chapter 5.4.1.

Observe that the filter has just been applied to the first link due to the computational cost is so high and the results and conclusions obtained for the first link could be extrapolated to the rest of them.

One important point that the developer has to decide will be the error covariance which we assign to the position data given by the ideal physical model developed in Chapter 5.1. As higher we choose this error covariance, less is the importance of the position given by the ideal physical model in the EKF (Extended Kalman Filter) for the state estimation.

Tuning the position error given by the ideal physical model as

$$v_1(t)=0.001m \quad v_2(t)=0.001m$$

and the inertial and gyroscope errors characteristics of the sensor used in this thesis

$$v_3(t)=0.02\, {}^{m}\!/_{s^2} \quad v_4(t)=0.02\, {}^{m}\!/_{s^2} \quad v_5(t)=1\, {}^{\deg}\!/_{s}$$

the estimation given by the Extended Kalman Filter for the first link will be the one shown in Figure 49.

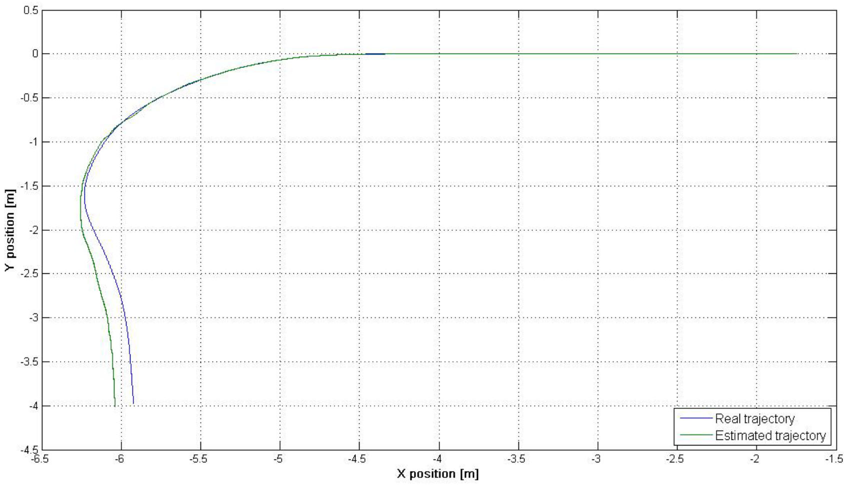


Figure 49 **EKF estimation** $v_1(t)$**=0.001 m** $v_2(t)$**=0.001 m**

For a better understanding we have to observe in Figure 50 the error committed between the real trajectory and the one estimated by the Extended Kalman Filter (EKF).

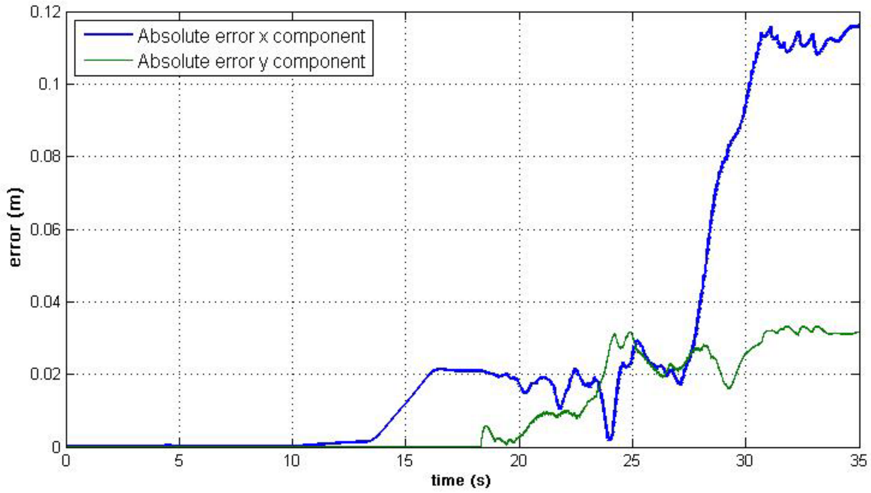


Figure 50 **Error committed by the EKF.**

As we can observe in Figure 50, to have chosen a low position error value in comparison to the acceleration error will mean that for the observer, the position measurement given for the ideal physical model will have a high importance in comparison to the information given by the acceleration sensors. For that reason, the error committed by the EKF estimator for the first link is similar to the error committed by the ideal physical model illustrated in Figure 48.

The next step will be to study how the filter works when we increase the weight of the position measurement and the position error has been defined as. $v_3(t) = 0.02\,m \quad v_4(t) = 0.02m$

The obtained trajectory applying the EKF estimator will be the one presented in Figure 51.

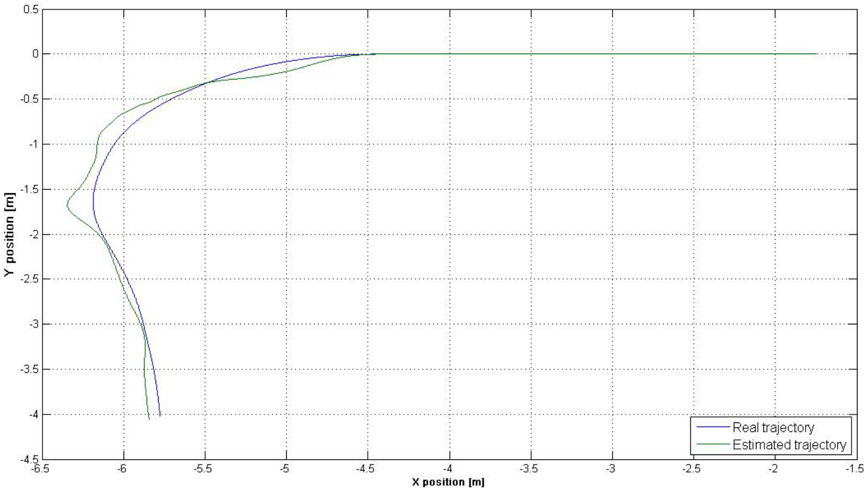


Figure 51 **EKF estimation $v_1(t)=0.02\,m$ $v_2(t)=0.02\,m$**

Illustrating how good the estimation given by the EKF (Extended Kalman Filter) estimator works it is plotted Figure 52 where the absolute error between the real trajectory and the estimation given by the EKF is shown.

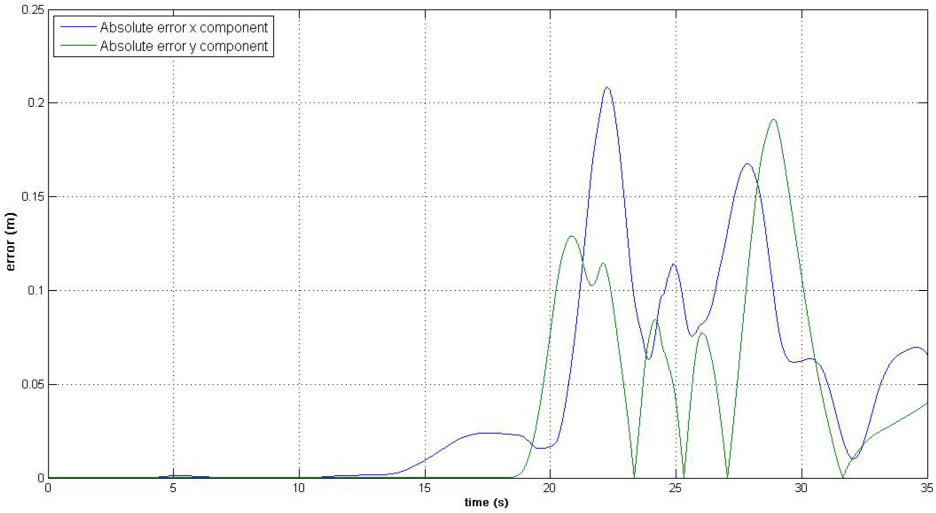


Figure 52 **Error committed by the EKF.**

Analyzing Figure 52 we can observe that between second 19 and second 30 the error is bigger than the one committed for the ideal physical model. Afterwards the error decreases and we obtain an acceptable estimation. That phenomenon is due to when the robot is turning its links high frequencies will appear in the acceleration which will not be synchronized with the accelerations that came from the ideal model. This will increase the error during the robot turning but ones the robot has again the links aligned the estimation will reach an accurate and precise estimation.

One problem that appeared during the observer development was the high computational cost which the filter has. The high dynamics caused by the forces model will oblige us to choose a small time step in the filter trying to avoid aliasing problems. In each time step the operational cost will be high due to the matrix linearization and several matrix inversions that the filter will calculate. This high computational cost will make impossible to apply the filter in real time.

All the graphs and results obtained could be checked running the Matlab Simulink model *definitive_model.mdl* and then executing the Matlab script *kf_extended.m.*

Advertisement. Due the small step size chosen, the Extended Kalman Filter script *kf_extended.m* has a huge computational cost and Matlab could cause some problems in old or not powerful computer related to the memory that Matlab uses for the calculations.

6. Conclusions

During this thesis we have studied a set of techniques for developing an accurate and precise navigation system for our multilink snake robot.

Our robot had two working modes. In the first one the robot was on the surface and it was able to receive the GPS (Global Positioning System) signal and the inertial measurements taken by the IMU (Inertial Measurement Unit) sensor. During the second case, the robot was inside a pipe or in an environment where it was completely impossible to receive the GPS signal.

In the first mode the availability of the GPS signal and the inertial measurements from IMU gave us several options to observe the position in the system.

The obvious option was to base the position navigation system on the trajectory given by the GPS receptor that the robot has installed. In this option we found two problems. First, the precision which the GPS system has and second, the slow sample frequency with which the GPS system runs. The trajectory given by the GPS system was useful for testing the quality of the other techniques applied for the first working mode. If the technique improves the GPS precision, the implementation of this technique will be reasonable.

Another possibility which we analyzed in the first working mode was to obtain the trajectory navigation integrating two times the acceleration measurements given by the IMU sensors. We demonstrated in Chapter 4 that this technique worked perfectly if the sensors were just affected by a pure white noise. The effect of a pure white noise is eliminated by the integration. The reality is that the inertial sensors are never affected by a pure white noise and are affected by other error sources like an offset signal. In this case of the IMU measurements affected by offset signals, we demonstrated how the offset error caused a position error which was accumulative and increased over the time concluding that the double integration of the inertial sensor signals is not an acceptable navigation technique.

After having rejected the two previous navigation attempts, during this thesis we applied a Kalman Filter multisensor fusion to GPS and IMU signals trying to reach a better position estimation. We observed that having an offset error signal in the IMU sensor, the Kalman Filter fixed the offset problem and we reached an accurate navigation system which reduced the error inherent to the GPS navigation system.

In the second working mode it was supposed that the robot ran in an environment where the GPS signal reception was not able. In that point, being impossible to receive the GPS signal, we developed a kinematic physical model which simulated the robot motion and estimated the robot trajectory. We studied what kind of errors the model suffered due to the simplification assumed in the model.

We observed that when the model was not running during a long time, the model gave us an accurate trajectory estimation and it could be used as navigation system. Instead, when the robot was working during a long time, the error increased and it was logic to investigate new techniques.

In the last part of the thesis we tried to implement Kalman Filtering for estimating the trajectory. The inputs in the filter were the trajectory given by the physical model, the acceleration and the head angle velocity given by the IMU sensor. After trying to implement the Kalman Filter we concluded that it was able to reduce the error in the estimation but the possibility to implement the filter on real time was low due to the high computational cost that we observed. The robot dynamics forced us to select a small step size in the simulation which increased the computational cost a lot.

7. Future work

During the thesis development we found several points which new thesis students, PhD students or research teams could continue researching more intensively and extend the results presented in this thesis.

This thesis was a theoretical work where we presented, explained, developed and programmed mathematical algorithms for GPS/IMU fusion guided to reach an accurate navigation system. Future works which will continue this theoretical work and try to implement these algorithms in the robot for analyzing empirically how well these algorithms work in a real situation could be interesting.

Talking about the model, we observed that really high frequencies appeared in the acceleration. These high frequencies were probably caused by the friction forces model we chose. It would be logical to expect that in reality, theses high frequencies do not exist and due to tire deformation or other reasons, the robot behavior would be running in slow frequencies. Explained this possible discrepancy between the model and reality, it would be interesting future works will try to refine the friction model or at least will research about this discrepancy.

In the future if we want to implement the algorithm in a microprocessor we should translate the algorithm to a high level programming language implementable in commercial microprocessors. Usually these microprocessors work with fixed point arithmetic when Matlab does it in floating point. So before the implementation, the code has to be evaluated trying to prevent some numerical analysis errors.

8. References

[1] SINTEF (2009) The Pipe Inspection robot PiKo. Available from: <http://www.sintef.no/Home/Information-and-Communication-Technology-ICT/Applied-Cybernetics/Projects/The-Pipe-Inspection-robot-PiKo> [Accessed: July 26, 2009].

[2] Caron, F., Duflos, E. et al. (2004). GPS/IMU Data Fusion using Multisensor Kalman Filtering: Introduction of Contextual Aspects. 6-11.

[3] Rios, J. A., White, E. (-). Fusion Filter Algorithm Enhancements for a MEMS GPS/IMU. Crossbow Technology, Inc. 1-2

[4] Gerder, J. C., Miller, S. L. et al. (2001). Calculating Longitudinal Wheel Slip and Tire Parameters Using GPS Velocity. Department of Mechanical Engineering. Stanford University. 1-2

[5] IEEE Biographies. (1974). Available from: <http://www.ieee.org/web/aboutus/history_center/biography/kalman.html> [Accessed July 26, 2009].

[6] Ballaguer, C., Garrido, S. and Moreno, S. (2003) Ingeniería de Control; Modelado, Análisis y Control de Sistemas Dinámicos. Madrid: Ariel editorial 372-377

[7] Bishop, G. and Welch, G. (2004) An Introduction to the Kalman Filter. Department of Computer Science. University of North Carolina at Chapel Hill

[8] Abbott, A. (2002). GPS and Inertial Navigation for Precision Weapon Delivery. Available from: <http://www.aero.org/publications/crosslink/summer2002/05.html> [Accessed: July 30, 2009]

[9] Fevig, R. A., Schultz, R. R and Wang, Y. (no date). Sensor Fusion Method Using GPS/IMU Data for Fast UAV Surveillance Video Frame Registration. Department of Electrical Engineering and Department of Space Studies. University of North Dakota.

[10] Musoff, H. and Zarchan, P. (2005) Fundamentals of Kalman Filtering: A Practical Approach. Publisher: AIAA (American Institute Of Aeronautics & Ast). 155-156

[11] Fukaya, M., Iwasaki T. and Saito M. (2002) Serpentine Locomotion with Robotic Snakes. Publisher: IEEE Control System Magazine. 64 -70

9. Appendix

MTi-G

MINIATURE AHRS WITH INTEGRATED GPS

xsens

MTi-G DEVELOPMENT KIT

The MTi-G DK contains the following:

- MTi-G (any configuration)
- Antenna
- USB cabling
- MT Software Development Kit (see below)
- Hardcopy documentation
- Optional: serial cabling
- Suitcase

MT Software Development Kit (MT SDK)

The MT SDK is an extensive set of tools for every level of interfacing, which allows configuring the MTi-G to the user needs, reading out and storing data and (re-)processing MTi-G data previously recorded. It also allows the user to extend own user source code with the MTi-G communication, using commands and code examples provided.

The MT SDK contains:

MT Manager Software

A specially developed, easy-to-use graphical user interface with possibilities to configure Xsens' sensors, reading out, store and show data in real-time charts and visualizations.

MT COM-object API and DLL API for Windows

Integrating the MTi-G in Windows programs, such as Matlab, C++ and Excel is made easy with the MT COM-object API and the DLL API. User-modifiable example code for programs Matlab, C++ and Excel (VBA) is included.

C++ Class and binary communication for any (RT)OS

A C++ class is available for users who want to use the MTi-G on a binary level. Direct communication without using the C++ class is possible, following the fully documented communication protocol.

Magnetic Field Mapper plug-in

The Magnetic Field Mapper plug-in enables compensation for hard and soft iron effects.

ACCESSORIES

Cable options

CA-USB2G

USB cable

CA-SERi-2,5

Serial cable
RS232 + pigtail

CA-DB9iG

RS232, DB9
power

PA-MP

Power adapter
(for CA-DB9iG only)

ANT

Antenna

The MTi-G is a miniature size and low weight 6DOF Attitude and Heading Reference System (AHRS). The MTi-G contains accelerometers, gyroscopes, magnetometers in 3D, an integrated GPS receiver, a static pressure sensor and temperature sensor. Its internal low-power signal processor provides real time and drift-free 3D orientation as well as calibrated 3D acceleration, 3D rate of turn, 3D earth-magnetic field, 3D position and 3D velocity data.

The MTi-G is an excellent measurement unit for stabilization and control of air and ground objects, even during situations of long term accelerations.

Highlights

- Real-time computed GPS-enhanced attitude/heading and inertial enhanced position/velocity data
- GPS integration overcomes typical IMU challenges
- Integrated AHRS, GPS and static pressure sensor
- On board DSP, running sensor fusion algorithm
- High update rate (120 Hz), inertial data at max 512 Hz
- Individually calibrated for temperature, 3D misalignment and sensor cross-sensitivity
- UTC referenced output

Compact design

- Compact and robust design
- Easy integration in any system or application (OEM)
- Low weight, ultra-low power consumption

High performance

The MTi-G is a combination of a MEMS IMU, GPS and barometer. Yet, the MTi-G is more than just a sensor assembly. The IMU, GPS and barometric information is blended together in Xsens' sensor fusion algorithm to estimate the most accurate orientation and position possible. Because of this fusion, the output is more accurate than the output from the IMU or GPS receiver only. For example, the MTi-G copes with transient accelerations; a typical error source for any AHRS using the gravity as its reference estimating roll and pitch. The loose coupling works both sides: double-integrating the accelerometers for short periods, the MTi-G is able to calculate position and velocity even during short GPS outages. There are several more corrections realized to aid the IMU functionality and to enhance the GPS measurements.

User friendliness

The MTi-G is a sensor which can be used in a wide range of applications. Because of the specific requirements for all these applications, the MTi-G uses different filter settings and constraints, implemented in scenarios. Among others, there are scenarios for use in automotive and aerospace applications.

Output

- 3D orientation (360°)
- 3D position and velocity (aided and unaided by inertial sensors)
- 3D acceleration, 3D rate of turn,
- 3D magnetic field

MTi-G TECHNICAL SPECIFICATIONS

Attitude and heading

Static accuracy (roll/pitch)	<0.5 deg
Static accuracy (heading)[1]	<1 deg
Dynamic accuracy[2]	1 deg RMS
Angular resolution[3]	0.05 deg
Dynamic range:	
- Pitch	± 90 deg
- Roll/Heading	± 180 deg
Maximum update rate:	
- Onboard processing	120 Hz
- External processing	512 Hz

Position

Accuracy position:	
- SPS	2.5 m CEP
Maximum update rate:	
- Onboard processing	120 Hz
- External processing	512 Hz

Interfacing

Digital interface	RS-232(max 921k6 bps) and USB (ext. converter)
Operating voltage	5 - 30V
Power consumption	610 mW (typical) - 910 mW (max)
Interface options i/o	SyncOut, AnalogIn (2x),
GPS antenna	SMA connector, active

Maximum operational limits

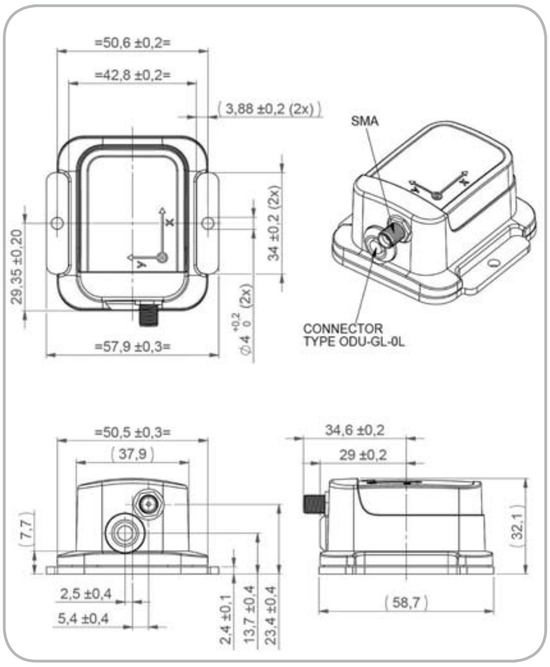
Altitude	18 km
Velocity	600m/s (2160 km/h)
Ambient temparature	
Operating range[4]	-20...+60 °C

INDIVIDUAL SENSOR SPECIFICATIONS

Sensor performance	Rate of turn	Acceleration	Magnetic field	Static pressure
Dimensions	3 axes	3 axes	3 axes	-
Full Scale (standard)	± 300 deg/s	± 50 m/s^2	± 750 mGauss	30-120 kPa
Linearity	0.1% of FS	0.2% of FS	0.2% of FS	0.5% of FS
Bias stability[4]	1 deg/s	0.02 m/s^2	0.1 mGauss	100 Pa/yr
Scale Factor stability[5]	-	0.03%	0.5%	-
Noise	0.05 deg/s/√Hz	0.002 m/s^2/√Hz	0.5 mGauss	4 Pa/√Hz (0.3 m/√Hz)
Alignment error	0.1 deg	0.1 deg	0.1 deg	-
Bandwidth	40 Hz	30 Hz	10 Hz	-
Max update rate	512 Hz	512 Hz	512 Hz	9 Hz

GPS

Receiver type	50 channels L1 freqency, C/A code Galileo L1 Open Service
GPS update rate	4 Hz
Start-up time cold start	29 s
Tracking sensitivity	-160 dBm
Timing accuracy	50 ns RMS



HARDWARE SPECIFICATIONS

Housing

Dimensions (WxLxH)	58x58x33 mm
Weight	68 g

Options

Full scale acceleration:		Full scale rate of turn:	
1.7g (17 m/s^2)	A33	150 deg/s	G15
5g (50 m/s^2)	A53	300 deg/s	G35
18g (180 m/s^2)	A83	1200 deg/s	G25

Product code:	MTi-G-28 A## G##
Standard version:	MTi-G-28 A53 G35

Note: Specifications subject to change without notice

[1] depends on usage scenario. In case the Earth magnetic field is used, it must be homogeneous
[2] under condition of a stabilized Xsens sensor fusion algorithm and good GPS availability
[3] standard deviation of zero-mean angular random walk
[4] non-condensing environment
[5] deviation over operating temperature range

TYPICAL USAGE APPLICATIONS

Automotive

- Vehicle dynamics analysis
- Racing cars and motorbikes
- Performance testing

Full access to valuable data is available for engineers working in any level of sports car or motorbike competitions. The MTi-G outputs are suitable to test and analyze the dynamic behavior of e.g. automobiles. Non-holonomic constraints can be used to further enhance accuracy. The roll and pitch remains accurate even during long term accelerations typical for this application.

Marine

- In-competition optimization of racing yachts
- State estimation of leisure yachts
- Backup system for high-grade GPS systems

For racing yachts, the MTi-G provides orientations of several parts of the ship, such as the roll angle of the hull or the movement of the mast. This allows a sailing team to refine the ship’s performance during a race, or include the data in the autopilot. Another major application is commercial shipping. Measuring roll and pitch as well as heading is important for container ships, cargo ships and surveying vessels. Installing the MTi-G together with high grade GPS systems is a logical choice to enhance accuracy and to reduce costs, especially in situations with limited GPS reception.

Unmanned ground vehicles and robotics

- Autonomous control for driving and walking robots
- Military and civil ground vehicles
- Camera/LIDAR stabilization and correction

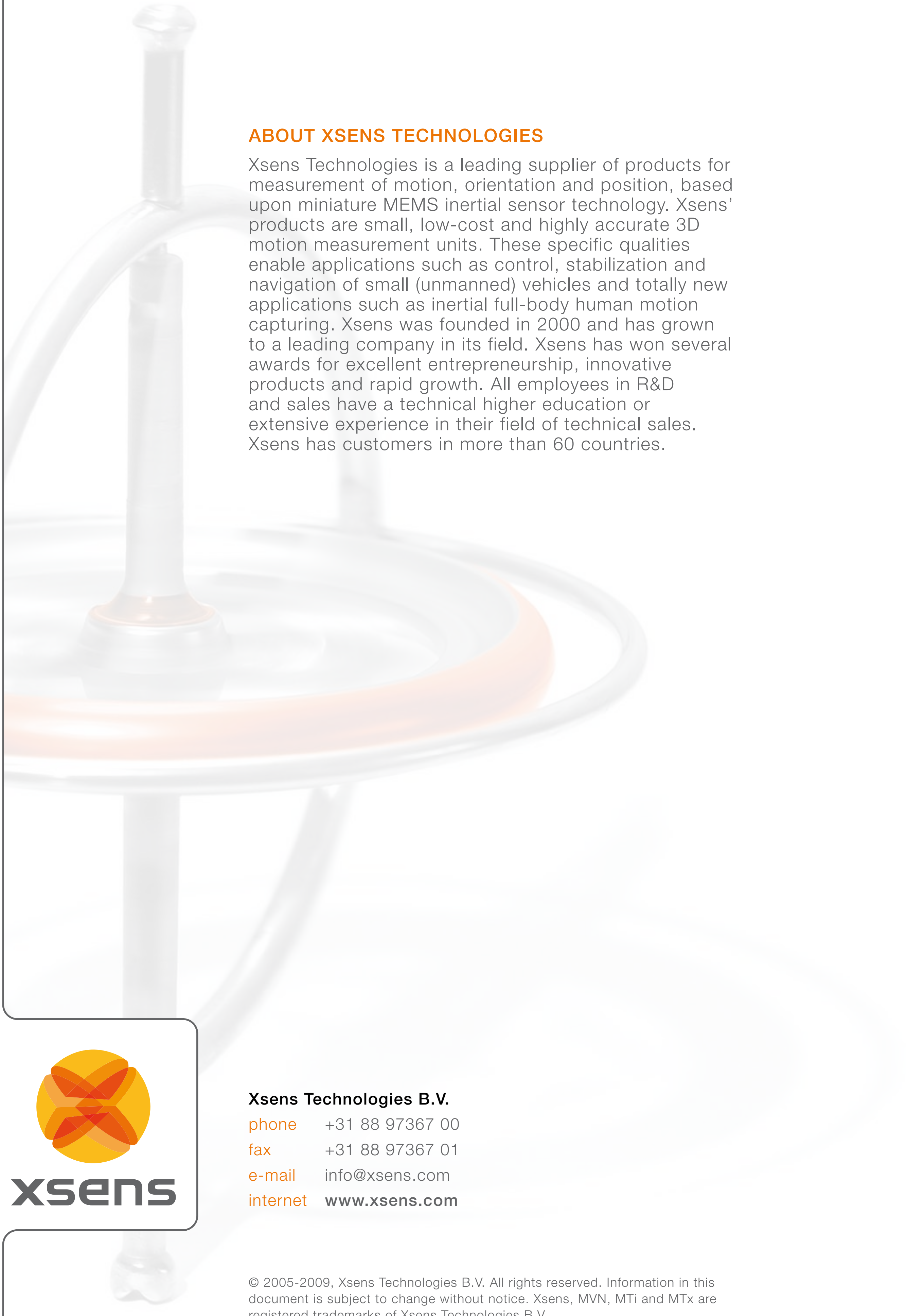
The MTi-G is an excellent sensor for driving and walking robots. The main functionality of the MTi-G in robotics applications is attitude control, even under dynamic conditions. Accurate position and orientation makes autonomous navigation possible for Unmanned Ground Vehicles (UGV’s), even on rough terrain or during short GPS outages. The MTi-G has been used in DARPA Grand Challenges for these purposes. These applications are typically out of reach for conventional mems imu’s.

Aerospace

- Autonomous attitude and navigation control
- Dynamics of (aerobatics) planes
- Camera/LIDAR stabilization and correction
- Head-up display

The MTi-G is the ideal choice for control and stabilization for any type of small to medium sized fixed-wing and rotary-wing aircraft. Because of the low latency, autonomous control can be designed in a simple and robust manner. The MTi-G provides a wide variety of dynamic data, suitable for dynamics analysis of (un)manned airplanes. An easy software/hardware interface makes data processing possible post-flight or even real-time. The MTi-G is easy to install and computes all the data required for e.g. an accurate artificial horizon or a digital map.

xsens



ABOUT XSENS TECHNOLOGIES

Xsens Technologies is a leading supplier of products for measurement of motion, orientation and position, based upon miniature MEMS inertial sensor technology. Xsens' products are small, low-cost and highly accurate 3D motion measurement units. These specific qualities enable applications such as control, stabilization and navigation of small (unmanned) vehicles and totally new applications such as inertial full-body human motion capturing. Xsens was founded in 2000 and has grown to a leading company in its field. Xsens has won several awards for excellent entrepreneurship, innovative products and rapid growth. All employees in R&D and sales have a technical higher education or extensive experience in their field of technical sales. Xsens has customers in more than 60 countries.

Xsens Technologies B.V.

phone	+31 88 97367 00
fax	+31 88 97367 01
e-mail	info@xsens.com
internet	**www.xsens.com**

© 2005-2009, Xsens Technologies B.V. All rights reserved. Information in this document is subject to change without notice. Xsens, MVN, MTi and MTx are registered trademarks of Xsens Technologies B.V.