Francisco Miguel da Silva Vieira do Coito
Licenciado em Ciências da Engenharia Eletrotécnica e de Computadores

Study on the Development of an Autonomous Mobile Robot

Dissertação para obtenção do Grau de Mestre em Engenharia Eletrotécnica e de Computadores

Orientador: Stanimir Stoyanov Valtchev, Prof. Doutor, FCT-UNL
Co-orientador: Robert Babuska, Prof. Doutor, TU-Delft

Júri:

Presidente: Prof. Doutora Maria Helena Silva Fino
Arguente(s): Prof. Doutor Luís Filipe Figueira de Brito Palma
Vogal(ais): Prof. Doutor Stanimir Stoyanov Valtchev

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Março de 2014

Acknowledgement

It is doubtful that anyone may arrive at the end of his dissertation without a good deal of help along the way, I'm certainly no exception. High on the list of those to thank is my advisor, Professor Stanimir Valtchev, for his patience and precious advice. Next on the list is my co-advisor Professor Robert Babuska, for welcoming me at Technishe Universiteit Delft (TU-Delft), to continue the development of my work, and Professor Gabriel Lopes, for his helpful counseling during my stay at TU-Delft.

It was a great pleasure to work with them. Their aid helped me to think carefully about my work's development. I also thank them for believing in me, for the scientific and technical support, and for the motivation.

I would also like to thank all the members of the Departamento de Engenharia Electrotécnica (DEE) of Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa (FCT-UNL) and of the Department of Systems and Control (Delft Center SC) of Technishe Universiteit (TU) Delft.

All my thanks also to my teachers, colleagues, family and friends, who aren't mentioned individually by name, for helping me to be who I am.

And finally, to my parents for the support given, as well as, for tolerating with my faults.

Abstract

This dissertation addresses the subject of Autonomous Mobile Robotics (AMR). It is aimed to evaluate the problems associated with the orientation of the independent vehicles and their technical solutions. There are numerous topics related to the AMR subject. Due to the vast number of topics important for the development of an AMR, it was necessary to dedicate different degrees of attention to each of the topics.

The sensors applied in this research were several, e.g. Ultrasonic Sensor, Inertial Sensor, etc. All of them have been studied within the same environment.

Employing the information provided by the sensors, a map is constructed, and based on this map a trajectory is planned.

The Robot moves, considering the planned trajectory, commanded by a controller based on Linear Quadratic Regulator (LQR) and a specially made model of the robot, through a Kalman Filter (KF).

Some of the researched topics were implemented in a real robot in an unstructured environment, collecting measurement data. A final conclusion is indicating the future direction of development.

Resumo

O tema desta dissertação é Robótica Móvel Autónoma (AMR). Este trabalho tem como objetivo estudar problemas associados com a orientação dos veículos independentes e as suas soluções técnicas. Existem tópicos numerosos relacionados com o tema AMR. Devido ao vasto número de tópicos importantes para o desenvolvimento de um AMR, foi necessário dedicar diferentes graus de atenção a cada um dos tópicos.

Os sensores aplicados nesta pesquisa eram vários, por exemplo, Sensor Ultrassónico, Sensor Inercial, etc. Todos eles foram estudados no mesmo meio ambiente.

Usando a informação provenientes pelos sensores, um mapa é construído, e uma trajectória é planeada com base neste mapa.

Considerando a trajectória planeada, o Robot move-se seguindo os comandos por um controlador baseado no Regulador Linear Quadrático (LQR) e com um modelo do robot, baseado em Filtro de Kalman (KF).

Parte dos tópicos pesquisados chegaram a ser implementados num robot localizado num ambiente não estruturado. Como conclusão final, é indicado a direcção de desenvolvimento futuro.

Subject Index

Figure Index

Table Index

List of Abbreviations and Symbols

Abbreviations

AMR	Autonomous Mobile Robot
A*	A Star algorithm
DP	Dynamic Programming
EKF	Extended Kalman Filter
IMU	Inertial Measurement Unit
KF	Kalman Filter
LQR	Linear Quadratic Regulator
MC	Monte Carlo
MR	Mobile Robot
RL	Reinforcement Learning
SLAM	Simultaneous Localization And Mapping
TD	Temporal Difference
UGV	Unmanned Ground Vehicle
US Sensor	Ultrasonic Sensor

Variables

α	Half of the Ultrasonic Sensor's rotation angle [°]
β	Step-size Parameter
θ_p	Pendulum's angle [°]
$\sigma(k)$	State Variance at instant k
Φ	Half of the Ultrasonic Sensor detection angle [°]

ϵ	Probabilistic Selection Parameter
γ	Discount Parameter
λ	Decay-rate Parameter
ρ	Observing Distance
π_ϵ	Policy Function
Λ_k	Noise Correlation Matrix between instants t_k and t_{k-1}
$\Phi(t_k, t_{k-1})$	Transition Matrix from t_{k-1} to t_k
∇f	Jacobian of function f
∇h	Jacobian of function h
Δt_k	Time increment between t_k and t_{k-1}
a_i	Available Action or Event taken at iteration i
A	Model's Dynamic Matrix
$A(t)$	System Dynamics Matrix at instant t
A_d	Model's Expanded Dynamic Matrix
A_p	Plant's Dynamic Matrix
b	Cart's Friction Coefficient
B	Model Dynamic's Input Matrix
B_d	Model Expanded Dynamic's Input Matrix
B_p	Plant Dynamic's Input Matrix
c	Speed of Sound $[m/s]$
C	Model Dynamic's Output Matrix
C_p	Plant Dynamic's Output Matrix
D	Model Dynamic's direct Throughput Matrix
D_C	Distance between the Sensor and Obstacle $[m]$

D_p	Plant Dynamic's direct Throughput Matrix
D_L	Distance between the side limit of the acoustic cone and its center $[m]$
D_O	Distance between the acoustic cone's point of origin and Obstacle $[m]$
D_R	Distance between the side limit of the acoustic cone and its center $[m]$
D_{S-O}	Distance between the acoustic cone's point of origin and the US Sensor $[m]$
$e(S_i)$	Eligibility Trace at state S_i
$f(\cdot)$	Function describing the vehicle's Kinematics
F	Horizontal Force applied to the Cart $[N]$
g	Gravity Constant $[m/s^2]$
$h(\cdot)$	Function describing the expected Observations
H_k^i	Measurement Matrix correspondent to the i^{th} measurement at instant t_k (z_k^i)
$\overline{H}_k^i$	Measurement Matrix correspondent to the i^{th} measurement (z_k^i), estimated from instant t_k to t_k^i
H_k	Measurement Matrices Collections between the instants t_k and t_{k-1}
I	Pendulum's Mass Inertia $[kg\ m^2]$
J	LQR's Cost Function
k	Instant or Iteration
k_m	Moving Distance
K	LQR Controller Gain
K_d	Expanded LQR Controller Gain
K_k	Kalman Gain at instant k
K_R	Set point Matrix
l	Pendulum's Length $[m]$
L	Environment map, Landmarks States

$\hat{L}_k$	Landmarks Estimate States at instant k
m	Pendulum's mass $[kg]$
M	Cart's Mass $[kg]$
N_x	Horizontal Forces applied by the Pendulum to the Cart $[N]$
P_k	Covariance Matrix at instant k
P_{LL_k}	Covariance Matrix of the Landmarks at instant k
P_{XL_k}	Covariance Matrix between the Robot's State and Landmarks at instant k
P_{XX_k}	Covariance Matrix of the Robot's State at instant k
P_y	Vertical Forces applied by the Pendulum to the Cart $[N]$
$q(t_k)$	State Noise Covariance Matrix of the continuous system at instant t_k
$Q(t_k, t_k^i)$	State Noise Covariance Matrix of the measurements between the instants t_k and t_k^i
Q_d	Expanded LQR's State weight Matrix
Q_k	Covariance of the State disturbance W_k
Q_{LQR}	LQR's State weight Matrix
r	Output's Set point
R	Robot's Orientation [°]
R_σ	Output weight Matrix affected by State Variance
R_d	LQR's Expanded weight Matrix
R_k	Measurement Noise Covariance Matrix of the measurements between the instants t_k and t_{k-1}
R_{LQR}	LQR's Output weight Matrix
R_w	Reward Function
S_c	Current Position State
S_i	Position State at iteration i
S_k	Innovation Covariance at instant k

t_k	Sampling instant at iteration k
T	Room Temperature [$°C$]
U_k	Control Vector at instant t_k
v_f	Variance Factor
v_k^i	Zero mean Gaussian Measurement noise from measurement z_k^i at instant t_k^i
v_R	Robot's Rotation Speed [$°/s$]
v_T	Robot's Forward Speed [m/s]
V	Value Function array
V_f	Value Function
V_k	Zero mean Gaussian Observation error at instant k
$w(t_k, t_{k-1})$	Estimated zero mean Gaussian State noise from instant t_{k-1} to t_k
W_k	Zero mean Gaussian State disturbance at instant k
x	Robot or Cart's Horizontal Position [m]
$X(t)$	Continuous System's State at instant t
X_k	Robot or Plant State at instant k
$\hat{X}_k$	Robot or Plant Estimated State at instant k
y	Robot's Vertical Position [m]
Y_k	Plant's Output
$\hat{Y}_k$	Estimated Plant's Output
$\underline{z}_j^l$	l^{th} measurement provided by the sensor j at instant t_k^i
z_k	Measurements Collection between the instants t_k and t_{k-1}
z_k^i	Measurement provided by the i^{th} sensor at instant t_k
Z_k	Robot's Observations at instant k

1 Introduction

1.1 Motivation

Everyday life incorporates an increasing number of intelligent machines capable of performing complex and diversified tasks. Purchases and sales, bank transactions, doubt clarification, or driving support, are examples of activities executed by such machines.

In what regards autonomous mobile platforms, these are often found in industrial environments, performing transport functions of materials or components. However, there are areas where the use of automatic machines is still incipient. One example is the execution of tasks requiring mobility in humanized environments. The use of Mobile Robots (MR) to perform autonomous tasks in offices, commercial areas, or even in households, is still at an early stage.

A reason for this fact may be attributed to present technological limitations. Homes or offices are less structured than factories, and are subjected to frequent changes. In particular, the available area for movement and position of objects undergoes frequent alterations. On the other hand, the installed infrastructure is very different from the manufacturing environment:

- Specific support elements for the mobile robot's localization and movement (beacons, sensors, etc.) are in small number or non-existent;
- Also the way humans relate themselves with the Unmanned Ground Vehicles (UGVs) is distinct. On a manufacturing plant moving machines are provided priority, while in an office or commercial space, humans always have priority.

This reality creates specific requirements to UGV development.

This dissertation addresses the development of an autonomous mobile platform capable of moving between different points of an environment with a (map) structure which is just partially known and suffers changes along the time. It is assumed that the tasks associated to localization and displacement of the mobile unit should be performed almost exclusively based on the sensor capabilities installed onboard.

The dissertation aims to study a set of capabilities that need Figure 1.1 to be implemented so the UGV may carry out its mission, and to identify the difficulties associated with it. The dissertation aims also at study of and the implementation of the identified functionalities.

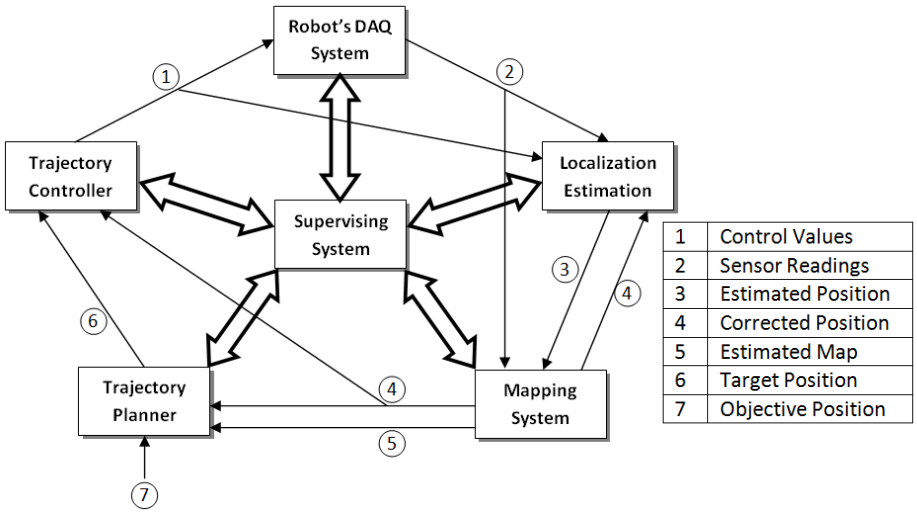

Figure 1.1 – Interdependency scheme of the subjects approached in this work.

1.2 Problem Formulation

What does a mobile robot need to be autonomous?

For a person to be able to give the first step it requires seeing where he (she) wants to go or, more importantly, where that first step is to be placed. Following this idea, before a mobile structure should start to move it is required to analyze the surrounding environment. Sensors, can either be located on the mobile structure or external to it, are what gives the robot the perception of its environment.

To have a better grasp of the mobile robot's surrounding environment, the information from the different sensors must be combined and integrated with the knowledge already available. The data provided by sensors may be used, for example, to update the environment information by adding new obstacles or removing obstacles no longer detected.

For the Mobile Unit it is not enough to know the location of every obstacle within the working area. It also needs to know how to perform the assigned mission. For that purpose it must be able to assembly a sequence of simple tasks to be performed, in order to complete the mission. A classic example is the planning of a trajectory for a MR to reach a specific point on the working area.

To implement a Task, like moving from one point to another, it is not enough just to order it. The robot movement is subjected to disturbances and uncertainties so its displacement must be done in a controlled way. This usually requires additional sensing information on the MR's location and movement.

This set of abilities has to be integrated by means of a supervising system capable of coordinating them harmoniously with the purpose of concluding its missions.

This dissertation addresses several of the elements afore mentioned. It is aimed to evaluate problems associated to the different subjects as well as the technical solutions. Due to this widespread nature the degree of attention dedicated to each subject is varied.

To gather relevant data from its environment the robot uses sensors. Both the sensors and the type of data provided are diversified: cameras provide images that need to be analyzed for extraction of features; long range obstacle detectors, like laser sensors and ultrasonic sensors, can give the distance to obstacle that are within line of sight; touch sensors detect obstacles when the robot gets in direct contact with them.

Whatever the type of sensor, the data provided must be analyzed to produce relevant information. The effort required to perform this analysis varies with the type of data, so the resources needed are different depending on the sensors installed on the robot.

One of the Mobile Platforms used for this work carries an Ultrasonic Sensor. One of the subjects addressed by the dissertation is the use of the Ultrasonic Sensor to detect obstacles, and to determine its location (distance and direction).

To plan a robot's route towards a goal position requires the use of a map, identifying the obstacles and the free paths. Humans create mental maps almost unwarily, that is not the case of an Autonomous Mobile Robot (AMR), where sensor measurements must be processed, giving raise to a map of the working environment. Maps besides being used to plan the UGV's path, can also help to determine the location within the working area. This subject is usually named by Simultaneous Localization And Mapping (SLAM), that is also discussed in this dissertation. The dissertation also deals with the problem of using existing map information to plan the robot trajectory over the free space within the working area.

For a mobile robot to know its current position along the pre-defined path, there is a number of solutions that complement the use of maps. Triangulation techniques using beacons, as well

as, inertial information and odometry are some of the more frequently used methods to determine the MR's position. Each of these approaches relies on its own type of sensors. It is possible to combine the information from several sensors, giving different types of data, to provide a better knowledge of the surrounding environment and improve the precision of the robot location. This subject is covered within the sensor fusion framework.

An UGV also needs a trajectory controller to allow it to follow the predefined trajectory. While it is following the trajectory, the UGV may be subjected to disturbances of different nature that may lead it off track. The trajectory controller's purpose is to compensate for disturbances and ensure an adequate tracking of the planned trajectory. The controller's performance is strongly connected to the ability to determine the robot movement attributes (such as position, speed, and orientation).

An important controller feature is the ability to comply with different degrees of precision from the movement attributes data. The Mobile Platform's controller needs to be able to tolerate some degree of error and to adapt the speed accordingly. In this dissertation an approach for the development of such a Controller is studied.

Figure 1.2 ilustrates the hierarchical dependencies between each subject approached in this dissertation.

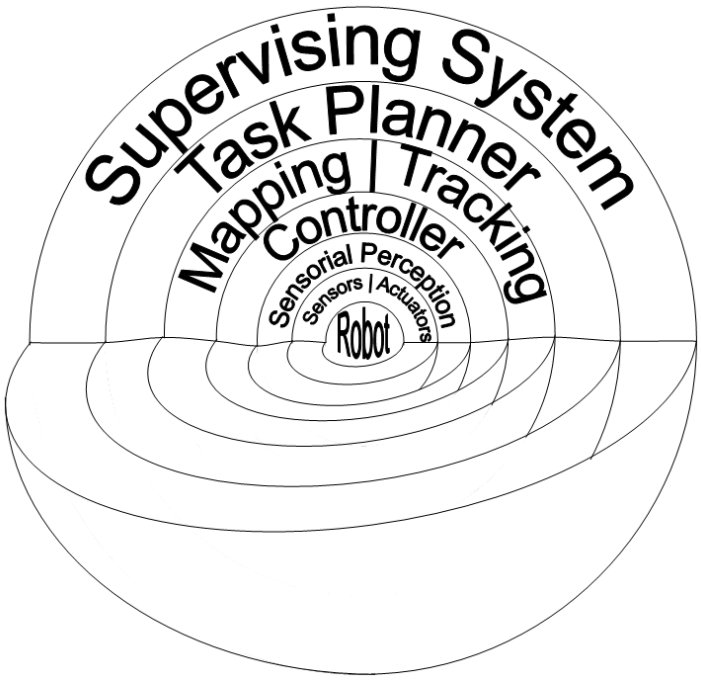

Figure 1.2 – Scheme of an Autonomous Mobile Robot's Systems
hierarchical dependencies between the various elements

1.3 Contributions

Development of an asynchronous sensor fusion algorithm and its integration with a trajectory controller. This controller uses a modified version of the LQR's algorithm. This modification consists on the use of a cost function, that incorporates the uncertainty associated to the robot's movement.

A trajectory planning algorithm was developed, based on a Reinforcement Learning scheme. This algorithm was implemented as an interactive program.

An ultrasonic sensor was studied, and its use for measuring the distance to an obstacle was analyzed.

A small autonomous robot was implemented. The robot is able to able perform a predefined trajectory without the help from external sensors or beacons.

1.4 Dissertation Overview

The dissertation is organized with the following structure.

The first chapter introduces the Motivation for the Dissertation's subject and presents the Problems that will be covered. The dissertation structure is also presented.

Chapter 2 addresses the study of Obstacle perception sensors, focusing on the Ultrasonic Sensor (US). It presents the basic functions of US sensors. It studies the sensor's sensing Area, that is, the sensing cone width and the relation between the true and the computed distance. Attempts are made to obtain the position and profile of the obstacles within the sensor's range. Throughout this chapter the problems encountered are also depicted.

Chapter 3 covers the trajectory planning problem using maps. Starts by dealing with the mobile robot's mapping problem, focusing on Simultaneous Localization And Mapping (SLAM).The basic reasoning behind is explained. A generic algorithm for Featured Based Mapping using Extended Kalman Filter (EKF) is also presented. An approach for the mobile robot's path planning is proposed through a Learning Algorithm. Reinforcement Learning is used to development a simple decision making system, dedicated for this problem. An

Occupancy grid-based map is used to compute the robot's trajectory, from the original position to its desired destination.

Chapter 4 deals with the problem of finding the mobile Platform's Localization, using only its own on-board sensors. It is assumed that each sensor has its own independent sample rate, although the sensor's sampling may be asynchronous. As an example, the delay associated with processing image information is usually random. Experiments are presented using two different Mobile Platforms, and the estimation of the platforms' position is made in a number of different fusion situations.

Chapter 5 addresses the development of a controller for the AMR. The planned controller is a Linear Quadratic Regulator (LQR) based adaptive speed regulator depending on the uncertainty of the robot's state. This controller is applied to one of the MR's, mentioned on the second part of Chapter 3, to follow a pre-planned trajectory.

Chapter 6 summarizes the results drawn from the work developed for the dissertation. It also proposes a set of research topics to be developed within this line of work.

During the development of this work, it was used two distinct robots to implement and collect data. One was a robust 4-wheeled robot, and the other one was a small 2-wheeled robot.

2 Environment Perception – Ultrasonic Sensor

An AMR requires sensing capabilities in order to avoid collision with static or moving obstacle. A number of sensors using different technologies exist for this purpose, such as lasers, infrared detectors and contact switches.

This chapter addresses the study of an Ultrasonic Sensor, its features and the ability to detect and evaluate the obstacle position.

2.1 The Ultrasonic Sensor for distance measurements

Ultrasonic sensors may be categorized as active sonars with monostatic operation. In other words, it emits sound pulses and receives the echoed signal. The sonar calculates obstacle's distance using the time interval between the signal emission and reception, as the distance is proportional to the time, knowing the speed of the traveled sound. Ultrasonic (US) sensors generate high frequency sound waves that are above the human hearing range (20 - 20 000 Hz).

Sonar is one of the most frequently used type of sensors for obstacle detection, mostly because they are cheap, easy to operate and make no physical contact, resulting no wear due to friction, and it doesn't alter the environment.

Ultrasonic sensors don't only carry the functionality of obstacle detection and avoidance. They are also used for localization and navigation. Figure 2.1 presents the mobile robot platform and a it's ultrasonic sensor mounted on a rotating servo.

Unfortunately, the simple operation of the Ultrasonic sensor also has its limitations and drawbacks on performing its tasks.

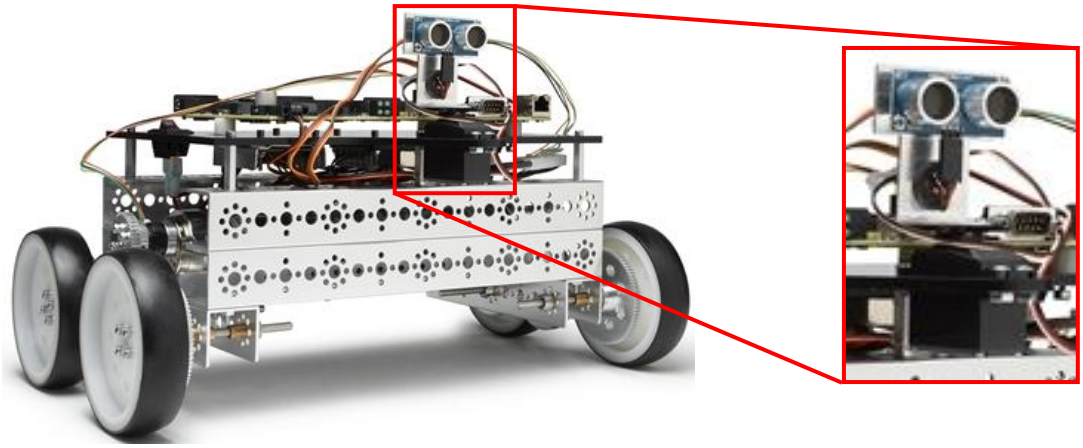

Figure 2.1 – *NI Robotics Starter Kit*

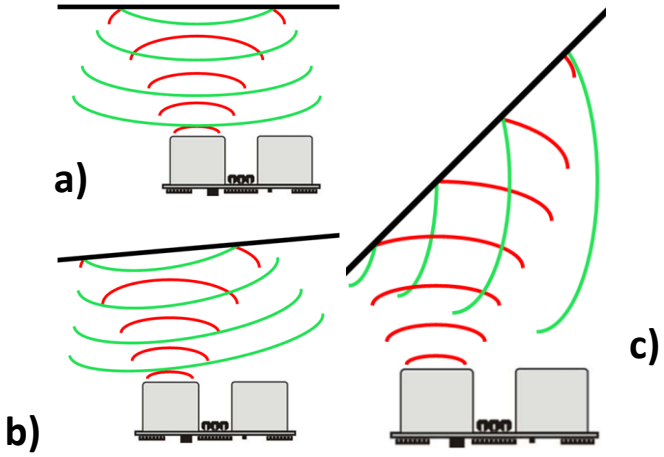

Figure 2.2 – Example of the sound reflection produced by the sensor. It shows two examples of obstacle detection (a and b), and an example of obstacle's misdetection (c).

The main problems with US Sensors, commonly mentioned by various authors [Cardin07; Ayteki10; Borens88], are:

- When the sound emitted by the sensor hits an obstacle's surface, it works almost like a light beam hitting a mirror surface. If the surface is perpendicular to the emitted signal, then it goes straight back to the sensor. However, the surfaces encountered aren't always parallel to the sensor, meaning, the obstacles aren't always detected, as the sound may be reflected away from the sensor (Figure 2.2.(c)).

- The signal emitted by the US sensor is in a form of an arch (acoustic cone), which gives the possibility that the obstacle detected might not be located right in front of the sensor. There might even be more than one obstacle at range and only the closest be detected (Figure 2.3).

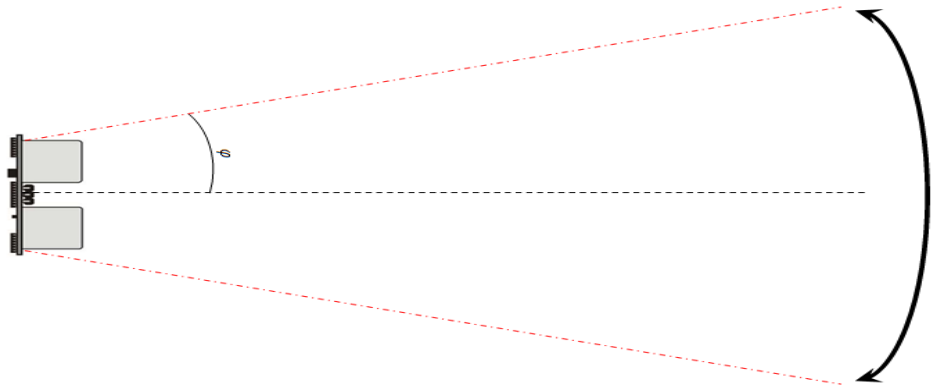

Figure 2.3 – Ultrasonic Sensor's acoustic cone, with a 2 φ angle of detection.

2.2 Ultrasonic Sensor Features

The ultrasonic sensor used in this work is a *Parallax Ping)))™ Ultrasonic Distance Sensor*. According to the sensor's datasheet [Ping13a; Ping13b], it can detect obstacles from $2cm$ up to $3m$, as well as measure the distance at which the obstacle is located. The acoustic cone has an angle of 40° (20° for each side Figure 2.3).

Although these features are indicated by the manufacturers, it is still necessary to verify them with a few tests.

2.2.1 Width of the Ultrasonic Sensor's Acoustic Cone

The ultrasonic sensor emits a sound, and during the journey from the sensor to the obstacle and back to the sensor, the sound disperses almost in a form of a cone (acoustic cone), as it loses energy (intensity). Any obstacle(s) located inside this cone may reflect the sound back to the sensor, allowing the sensor to compute the distance to the target, but not the position within the cone. To have a notion of the sensor's sensing cone, it is required to know the Sensing Angle (2 φ), the angle of detection, and the distance between the sensor and the origin of the cone (D_{S-O}).

An experiment was made to extract this information. For the tests an aluminum obstacle with a round surface, like a pipe, with a diameter of $2cm$ was used. In the experiment (Figure 2.4), the pipe is positioned in front of the sensor, at a distance of $28.5cm$ (D_C). The aim was not on evaluating the sensor readings, but on the detection of the obstacle. The obstacle is

placed over a parallel line to the sensor, located at a distance D_C from the sensor. The obstacle starting position is located outside of the sensor's detection range.

The target is moved over the parallel line to the sensor, and the region where detection occurs is registered. The sensor detects the obstacle from a distance of $12.3cm$ (D_L) to the left of the center until $12.7cm$ (D_R) to the right.

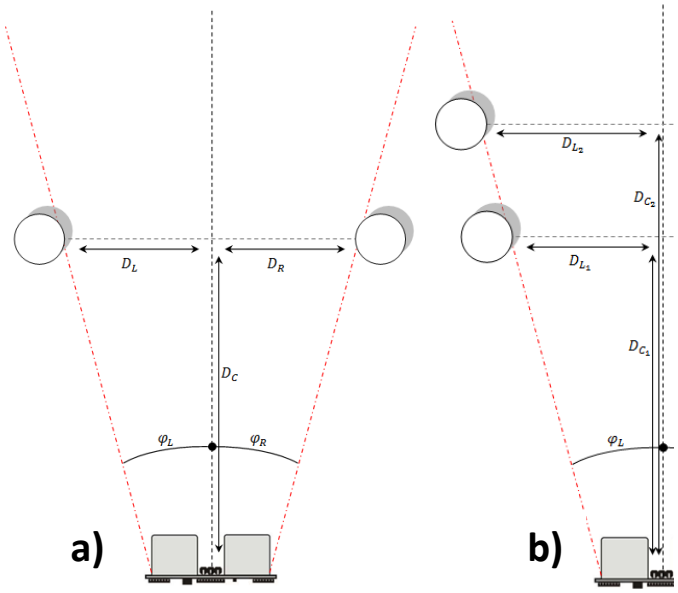

Figure 2.4 - Representation of the experiment performed to obtain the angles of the acoustic cone.

The procedure was repeated at a different distance D_C from the sensor. With the target placed at $90cm$ (D_{C_2}) the range of the obstacle detection is between $25.5cm$ (D_{L_2}) to the left and $24.5cm$ (D_{R_2}) to the right.

$$\varphi_L = \sin^{-1}\left(\frac{D_{L_2} - D_{L_1}}{D_{C_2} - D_{C_1}}\right) \tag{2.1}$$

$$\varphi_R = \sin^{-1}\left(\frac{D_{R_2} - D_{R_1}}{D_{C_2} - D_{C_1}}\right) \tag{2.2}$$

$$D_O = \frac{D_L}{\sin(\varphi_L)} = \frac{D_R}{\sin(\varphi_R)} \tag{2.3}$$

From these measurements the angles of detection (φ_L and φ_R) were computed, resulting in $12.7°$ and $11.7°$, respectively. The total detection angle of the used sensor was about $24.4°$ (φ_L

+ φ_R), but knowing the acoustic cone's angular width isn't enough, it is required to know how far is actually the point of origin (D_O, distance between the obstacle and the sensor's parallel line), as it is not at the same distance as is the distance to the sensor, D_C. The average distance D_{S-O} is $28.4cm$ from the sensor to the point of origin to the sensor. Knowing this angle can allow for a more precise location the obstacles, or to delimit the obstacles geometry.

2.2.2 Ultrasonic Sensor's Measurement Readings

The second experiment aims to assess the accuracy of the distances to target, measured by the sensor. For this purpose it an obstacle with a flat surface (turned to the sensor) positioned around the central position is used. The test consists on placing the obstacle as far as possible from the sensor and recording the true distance and the measure provided by the sensor. The target is then moved closer to the sensor and new measures are obtained. This procedure is repeated several times until the obstacle gets as close as $10cm$ from the sensor. The collected data is used to compare the sensor measures with the true distances and to compute any necessary corrective actions (gain or off-set error correction) and to evaluate the consistency of results.

At the beginning of this experiment, it was noticed that the sensor was providing incorrect values, indicating that the target was far closer than it really was. As an example, with the obstacle at $1m$, the sensor presented measured values between $0.10m$ and $0.15m$. This error occurs systematically. The explanation for this problem appears to be related with echoes occurring within a closed environment. The generated sound has enough intensity so that, after being reflected, it reaches another obstacle, from which it bounces back returning to the US sensor and confusing it. During the experiments, the echo was from the wall behind the robot or the robot itself (Figure 2.5).

Following equation (2.4), with the room temperature (T) about $20°$, the speed of sound, during the experiment, was $343.4m/s$.

$$c = (331.3 + 0.606\,T) \tag{2.4}$$

According to the data sheet [Ping13a; Ping13b], the burst of pulses can be detected from $115\mu s$ and $18.5ms$, in other words, it can detect obstacles from about $2cm$ up to $3.2m$. If the

burst of sound took longer than $18.5ms$ to reach back to the sensor, it could confuse the sensor by making it think the obstacle is at a few centimeters away, by associating the sound burst to the next cycle.

In the present situation, the echo from the wall, which was about $1.8m$, took about $28.2ms$, $3.6ms$ longer than the maximum detection time, with the obstacle at $1m$ (total distance traveled by the sound was $3.8m$). When the echoed sound reaches the sensor, it assumes the obstacle is closer than it is.

As the sensor does not allow the intensity of the generated sound to be adjusted, it was not possible to reduce the intensity of the produced sound.

The same experiment was performed with the obstacle covered by a sound absorbing material, such as cloth. This has an effect similar to reducing the sound intensity and allows the sensor to provide adequate measures. Measures were recorded with the covered obstacle positioned at distances ranging from $1.625m$ down to $8.5cm$ away from the sensor. The results are summarized on the graph from Figure 2.6. All the measurements provided by the sensor had an average value that is $2.5cm$ larger than the actual distance.

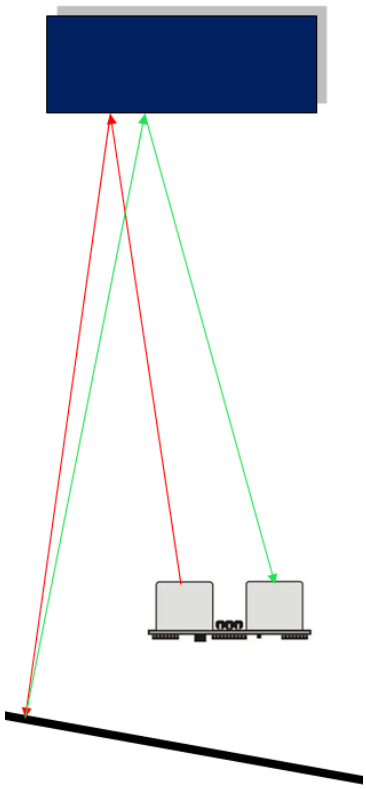

Figure 2.5 – Example of the occurrence of an Echo.

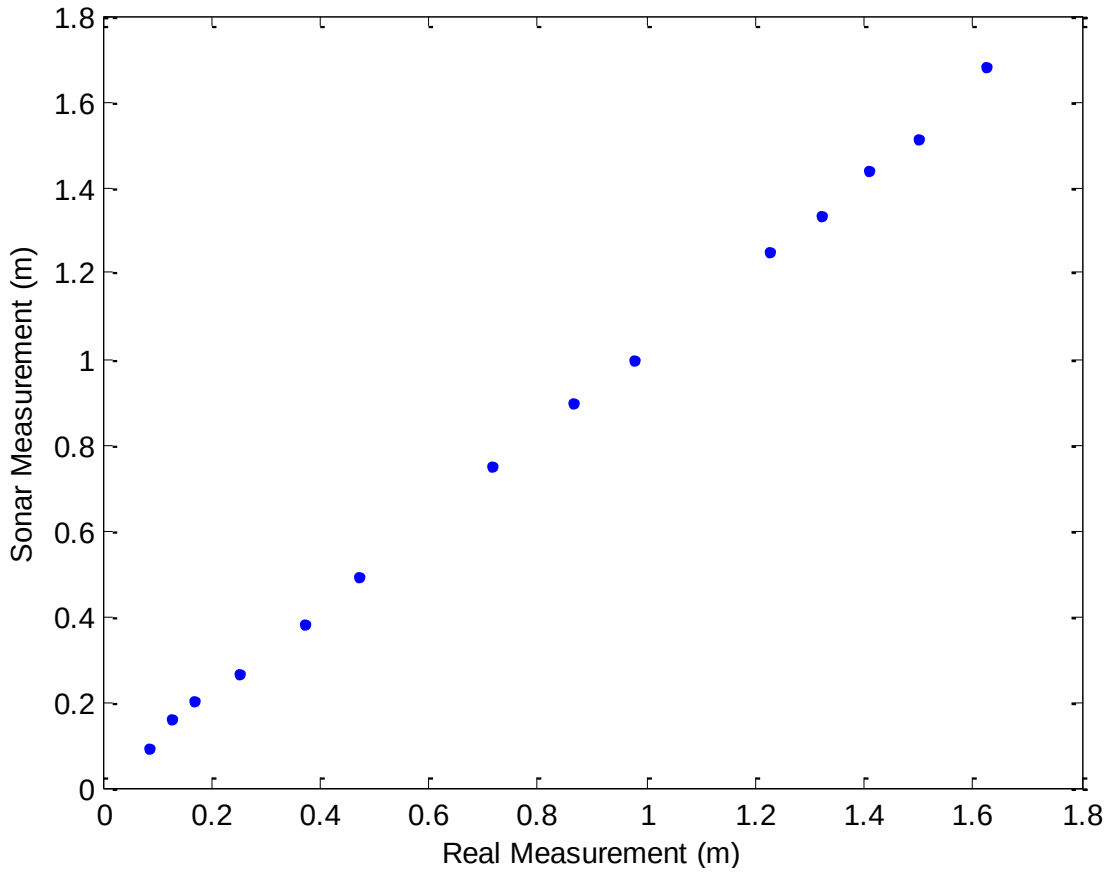

Figure 2.6 – Relationship of the measurement provided by the sensor and the actual value. The measurements are the distances between the detected obstacle and the sensor.

2.3 Target Detection

2.3.1 Use a Rotating Ultrasonic Sensor

On another experiment the ultrasonic sensor was rotated (Figure 2.7 and Figure 2.8), to evaluate the obstacle's profile obtained from using a rotating sensor. As the rotation axis is not placed at a symmetry axis of the sensor geometry, the rotation produces an asymmetry when comparing the readings. The readings obtained when scanning from left to right are not the same as the readings scanned from right to left. This variation can also be seen as another problem that isn't mention by many authors.

The results of this test are presented on the Figure 2.9, which shows the temporal progress of the measured distance by the sensor, as with its angular position. The Target used has a flat surface, $29cm$ wide, positioned about $88.5cm$ away from the sensor, parallel and centered to

it.The sensor turns with a speed of $50°/s$, within a range $[-25°, +25°]$. The target surface was covered with a sound absorbing cover to reduce the burst intensity emitted by the sensor.

The test results are summarized on the graphs from Figure 2.10 to Figure 2.13. The first graph (Figure 2.10) shows the measured distance as a function of the rotation angle. As expected it may be noticed a slight increase of the distance towards the edges of the surface. From the information used to produce Figure 2.9, the instant when the obstacle's limit was detected and the corresponding sensor's orientation, were extracted. This information was assembled on the

Table 2.1 is divided between the reading made from right to left and the readings made from left to right. As the table shows, there is a significant difference between the detected angles.

The second graph plots (Figure 2.11) the target surface on a $< x, y >$ plane. The average distance, provided by the sensor is $88.1cm$. Without taking into account the limits of the sensor's acoustic cone, the obstacle is seen as having an average width of $44.4cm$.

At this distance, according to the equations presented (2.1**)** to (2.3), the sonar cone, obtained with the round tubular target, has a width of $48.3cm$, wider than this new target estimated width. This shows that the obstacle surface geometry is very important as it affects the measures produced by the sensor. A round surface obstacle, such as a tube, has the capability to reflect the burst of sound back to the sensor, wherever it is located within the acoustic cone. Flat surface obstacles are different, because of their relative orientation to the sensor affects the detection. The obstacle angle may either allow it to be seen, by reflecting the signal back to the sensor, or not seen, by reflecting the signal away from the sensor. The echoes resulting from secondary reflections on other obstacles also contribute to confuse the sensor measurements. A more extensive research on this subject is required, which is beyond the scope of this document.

The results of this experiment are also shown in Figure 2.12, which separates the measurements read by the sensor from Right to Left from the ones made from Left to Right. The measurements read by the sensor from Left to Right are presented in Figure 2.13. There is a difference of the detected obstacle's position between the readings made from Left to Right and Right to Left. The results obtained, when the sensor is moving from Left to Right, present a symmetrical width around the central angle ($0°$).

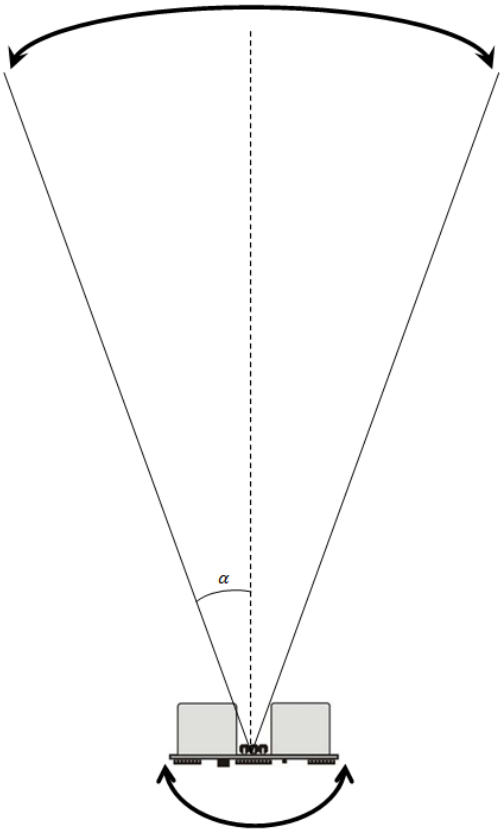

Figure 2.7 – Scheme of the Sensor's Roratory axis. Its total rotating angle is 2 α.

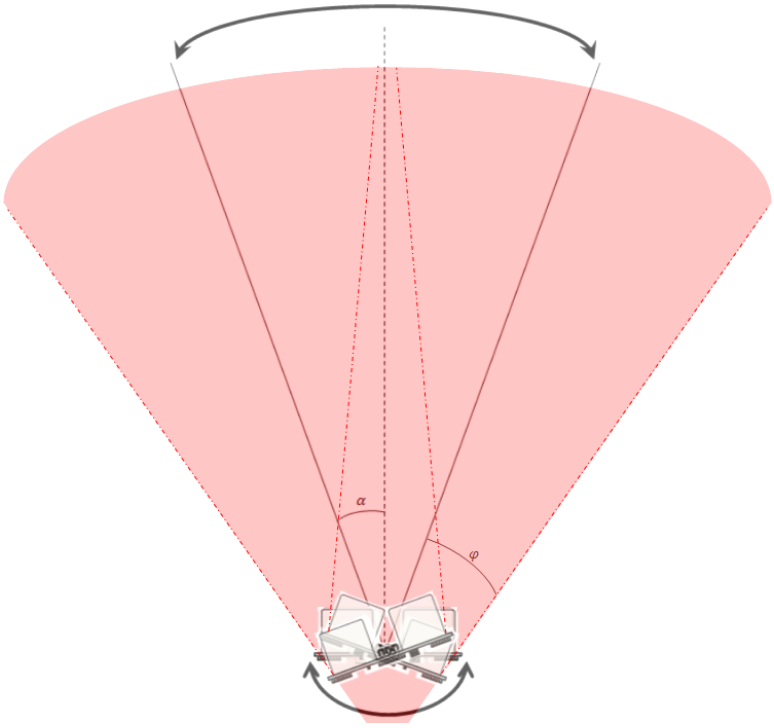

Figure 2.8 – Detecting area of the sensor, when using a Rotating Axis.
This is a result of a combination of Figure 2.3 and Figure 2.7.

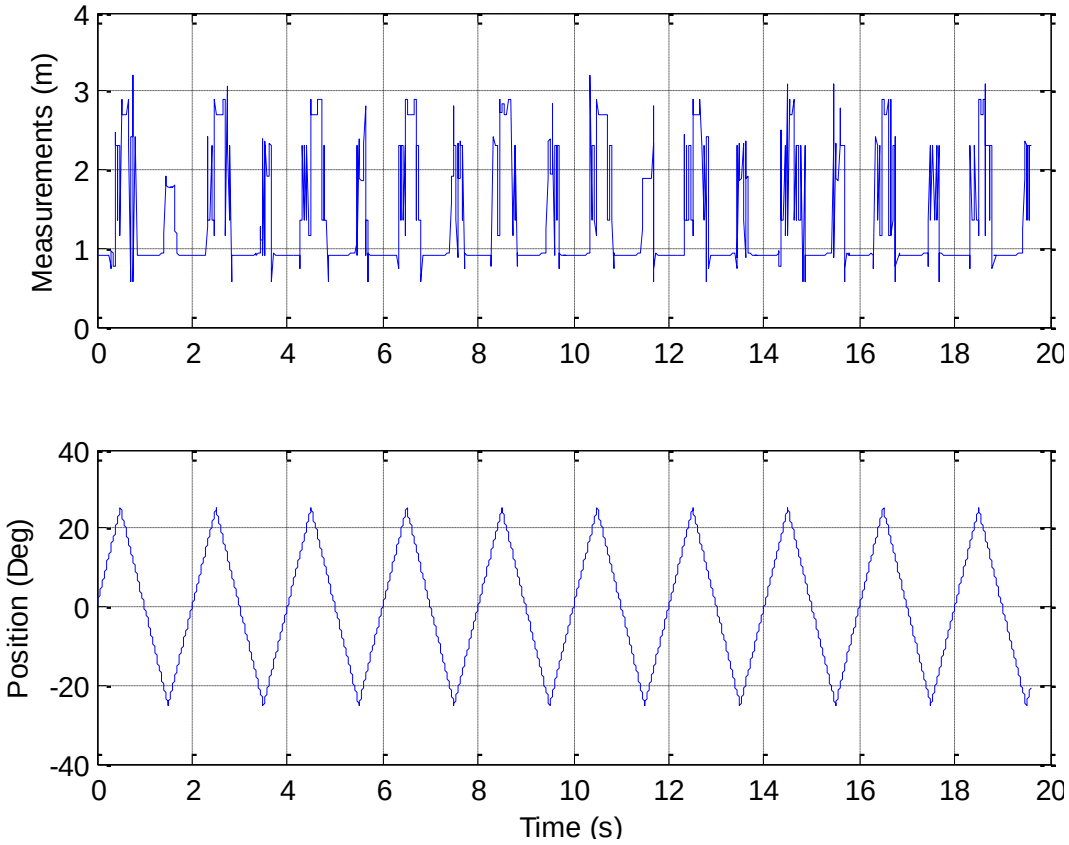

Figure 2.9 – Results obtained from the Rotating US Sensor.
The top graph presents the measured value.
The second graph shows the Sensor's orientation over Time.

Table 2.1 – Obstacle limit detection.

Right to Left				Left to Right			
Right side of the obstacle		Left side of the obstacle		Left side of the obstacle		Right side of the obstacle	
Time (*s*)	Angle (°)	Time (*s*)	Angle (°)	Time (*s*)	Angle (°)	Time (*s*)	Angle (°)
0.83	8.5	1.40	−20	1.67	−16.5	2.30	15
2.82	9	3.42	−21	3.68	−16	4.25	12.5
4.84	8	5.43	−21.5	5.69	−15.5	6.32	16
6.82	9	7.41	−20.5	7.70	−15	8.28	14
8.81	9.5	9.41	−20.5	9.69	−15.5	10.33	16.5
10.83	8.5	11.42	−21	11.69	−15.5	12.31	15.5
12.85	7.5	13.42	−21	13.68	−16	14.33	16.5
14.85	7.5	15.43	−21.5	15.70	−15	16.29	14.5
16.77	11.5	17.43	−21.5	17.68	−16	18.31	15.5
18.80	10	19.42	−21				
	Mean		Mean		Mean		Mean
	8.9		−20.95		−15.67		15.11
	Obstacle's detection Angle width (°)				Obstacle's detection Angle width (°)		
	29.85				30.78		

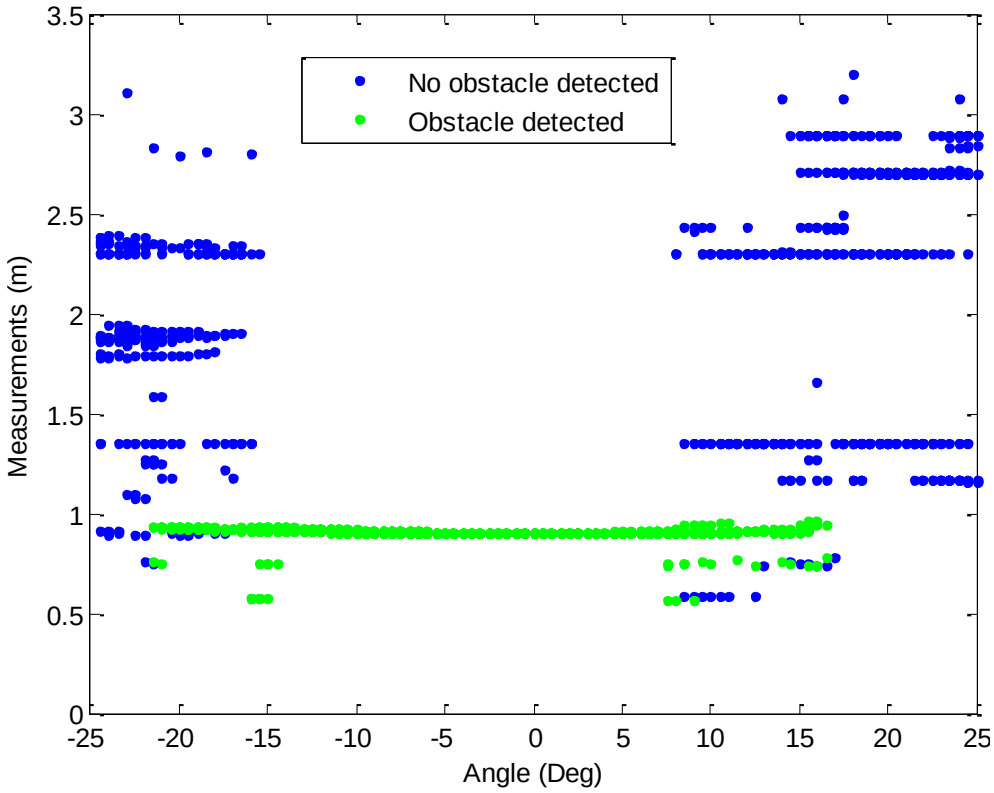

Figure 2.10 – Readings obtained by the sensor, highlighting the detected obstacle. Measurements between the Obstacle and the Sensor over Time. This data is the same as in the Figure 2.9.

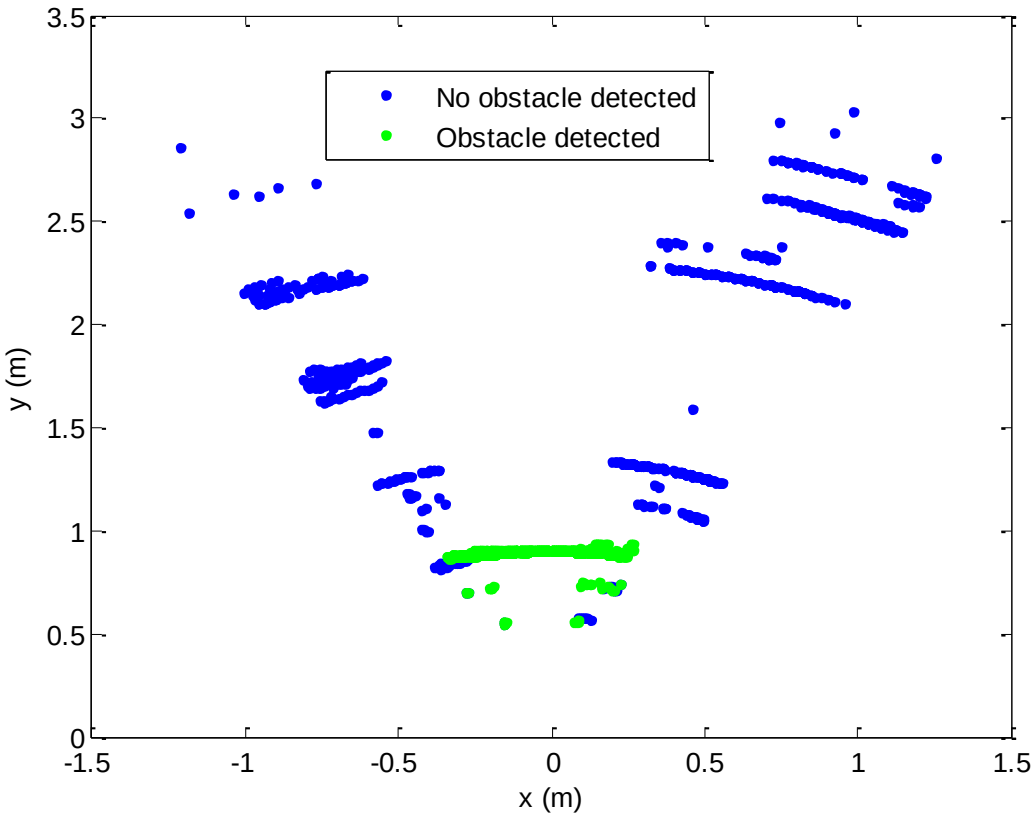

Figure 2.11 – Position of the detected obstacle on a $< x, y >$ plane.Same information as the one used in Figure 2.10.

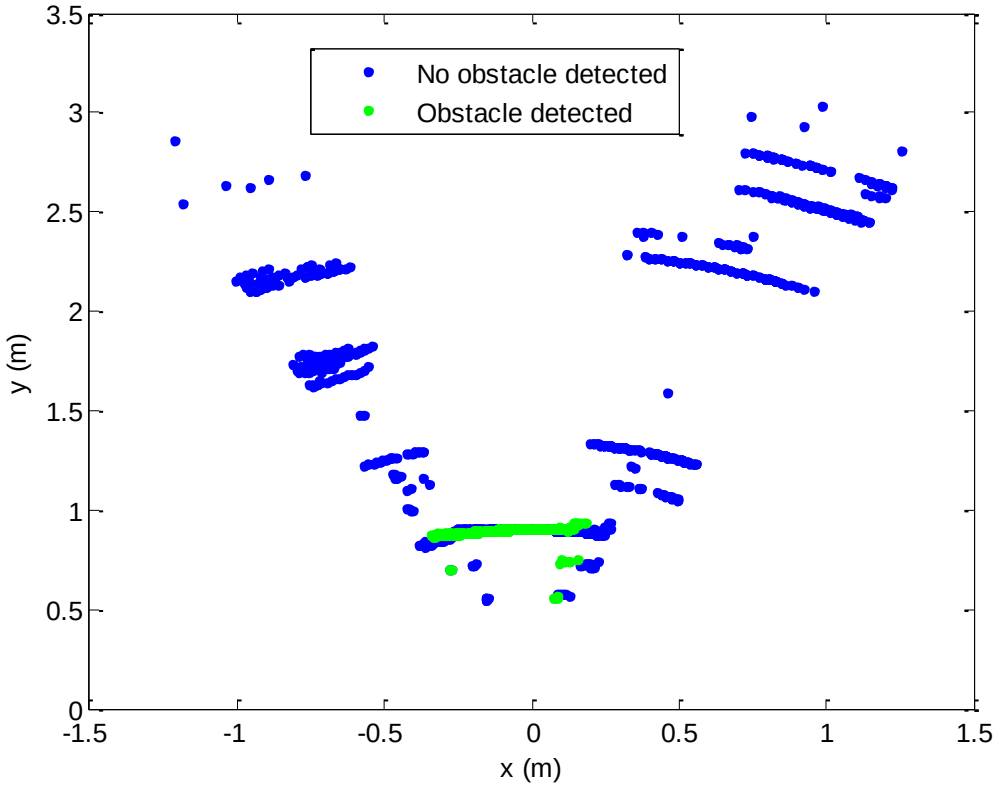

Figure 2.12 – Position of the detected obstacle on a $< x, y >$ plane, identifying with only readings from Right to Left. Same information as the one used in Figure 2.11.

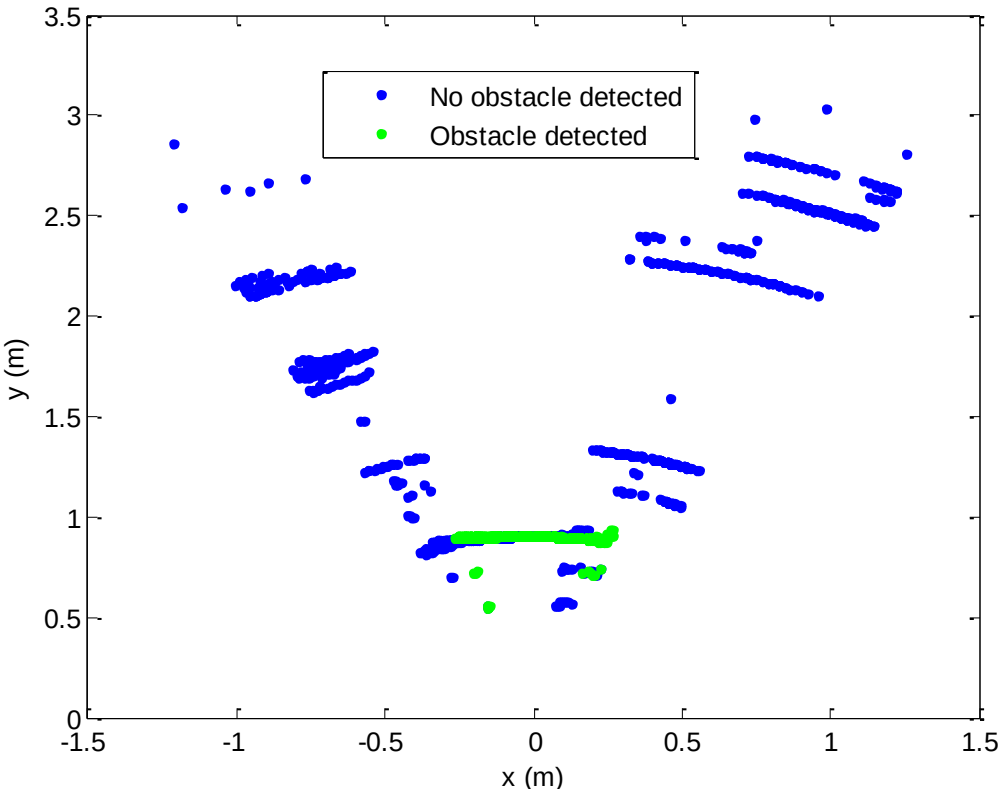

Figure 2.13 – Position of the detected obstacle on a $< x, y >$ plane, identifying with only readings from Left to Right. Same information as the one used in Figure 2.11.

2.3.2 Other ways to detect a Flat Surfaced Target

On the previous experiments, for the sensor to provide correct measurements, the obstacle had to have a sound absorbing surface, to avoid echoes. The next results are obtained with an uncovered flat obstacle (no sound absorbing surface). However, to avoid the echo the obstacle cannot be perfectly parallel to the US sensor. Echo effect is avoided by rotating the obstacle by a few degrees. This way, secondary sound reflections are avoided.

It is verified that the closer the obstacle is to the sensor, the larger the angle of the obstacles needs to be. This can be verified from the data from Table 2.2, which presents the distances the obstacle was positioned and the angle it was adjusted in relation with the sensor. The angle indicated was the minimal angle found so the sensor could detect the obstacle.

Table 2.2 – Distances from the obstacle and related angles.

Obstacle's Distance (m)	Obstacle's Angle (°)
1.64	4
1.33	7
1.04	8
0.83	10
0.58	13
0.31	24
0.22	31

2.4 Summary

This chapter addressed the study of the Ultrasonic Sensor installed on the *National Instruments* mobile robot, its characterization and the evaluation of the problems on detection of different obstacles.

A comparison between the sensor's reading and the actual distance was made, showing that it possessed an average offset of $2.5cm$.

An experiment was made to verify the sensor's sensing width, which demonstrated the sensor had a sensing angle of about $24°$, close to what the datasheet indicated ($20°$).

Problems were encountered, related with the measurements within a close environment, such as the occurrence of echoes, which confuses the sensor's readings. This problem has several solutions, one of the first in mind is to reduce the intensity of the emitted sound. That is not possible to implement with the sensor under analysis. This approach shows as a drawback a

decrease of the sensor's detection range. With a different sensor it would be interesting to study the development of an adaptive scheme to avoid echo, but still keep the detection range.

Under controlled conditions it is possible to cover all the obstacles with sound absorbing material, as it was done for the experiments. The use of an autonomous MR within an unknown environment where unpredictable obstacles may be encountered, such as in an office, makes this solution unfeasible.

Another option is the development of a supervising system, using information previously collected, as well as information from other sensors, to depict the obstacle, its position, profile, among others. To do so, the sensor needs to be extensively studied.

3 Mapping and Trajectory Planning

This chapter addresses the problem of map construction and its use for trajectory planning. It describes the use of the algorithm EKF-SLAM to associate the robot with the various detected landmarks on the environment. An algorithm, based on reinforcement decision making, is developed for the planning of the mobile robot's trajectory.

3.1 Environment Mapping

Maps are an important element for planning and control of AMR Movement. They allow to plan the sequence of actions to be performed by the robot, to verify the robot's actual position and to make necessary corrections to the robot's performance.

The construction of a map requires the environment's geometry and the obstacle positions within the environment to be perfectly known [Thrun05]. However, this is not the kind of environment in which humans live:

- Generally speaking, there is no complete or exact information about the surrounding environment;
- The surrounding environment is subject to changes.

These motives make the robot mapping an important subject in association with AMR, as well as a pointer for research. The basic goal is to build a map (mostly 2D/3D) of the surrounding environment, detecting walls, furniture and other obstacles, and it may also be used to determine the observer's (robot) position and orientation to plan trajectories to take [Thrun05; Durant06; Valenc13].

An UGV may locate its position through one of two types of mechanisms:

- Idiothetic – uses a combination of models and sensors, like IMUs, to estimate the Mobile Unit's position by dead reckoning. This means it tends to accumulate error.
- Allothetic – uses external information to locate itself, such as landmarks, by triangulating its position. This approach tends to present constant error, depending on the used sensors' and the landmarks' reliability.

To build a map of the environment through the robot's sensorial information requires the knowledge of robot's position and the estimation of the obstacles position. This means that the mapped obstacle positions depend on the robot's own position, giving rise to a correlation between the obstacles' and the robot's position errors [Thrun05, Riisga05, Namins13].

On the other hand, to locate the robot's position using the map, it is necessary to know which obstacles and landmarks are within the robot's range. The detected landmarks need to be already registered on the map, so that the mobile platform may be able to locate itself. In this case the robot's position depends on the position of the obstacles around it [Thrun05].

The acronym SLAM stands for Simultaneous Localization And Mapping. It is a set of techniques for mapping within the autonomous mobile platforms framework. It combines localization techniques with mapping, both computations being performed simultaneously. SLAM is a "chicken-or-egg" type of problem – a map is needed for localization, and localization is required for mapping.

3.1.1 Mapping and Localization of the Robot

When the robot moves, that is, it obtains a new position, and it updates its estimated position through odometry computation. On this new position, the robot extracts the landmarks and attempts to associate these to the landmarks that it has previously seen. The re-observed landmarks are used to update the robot's position, while the new ones are added to the map.

The basic algorithm is composed by the following steps:

1. Update the current state estimate using the odometry data.
2. Update the estimated state from re-observing landmarks.
3. Add new landmarks to the current state.

These steps are considered on the Figure 3.1. The first step involves the odometry information, and the robot's state. This state usually contains the robot's position, which includes its orientation, (x, y, R) and the mobile robot's control signals. This information allows the estimation of the MR's next position.

The second step uses the robot's position estimation to identify the landmarks and their configuration relative to the robot. The actual position (or an approximation) is determined and updated from these landmark configurations.

This step also includes the landmark's position uncertainty update, for every known landmark.

The third step adds the newly detected landmarks to the map using the robot's corrected position.

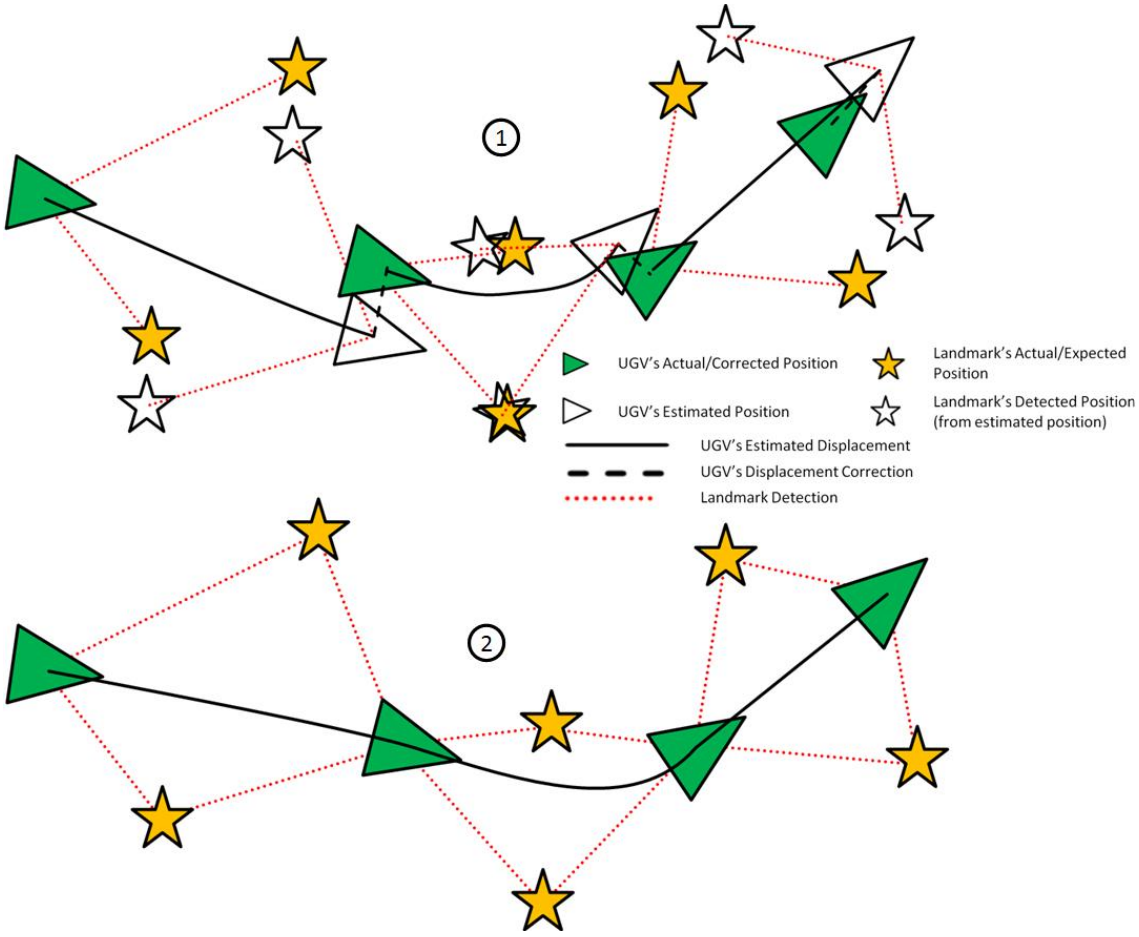

Figure 3.1 – Example of the UGV's SLAM process.
1 – UGV's estimated performance;
2 – UGV's Actual/Resulting Performance.

3.1.2 Some Considerations about Landmarks

Landmarks should be re-observable by allowing the detection from different positions and angles. They should also be unique enough so they can be easily identified or distinguished from each other, so there might not be a mix-up when re-observing them.

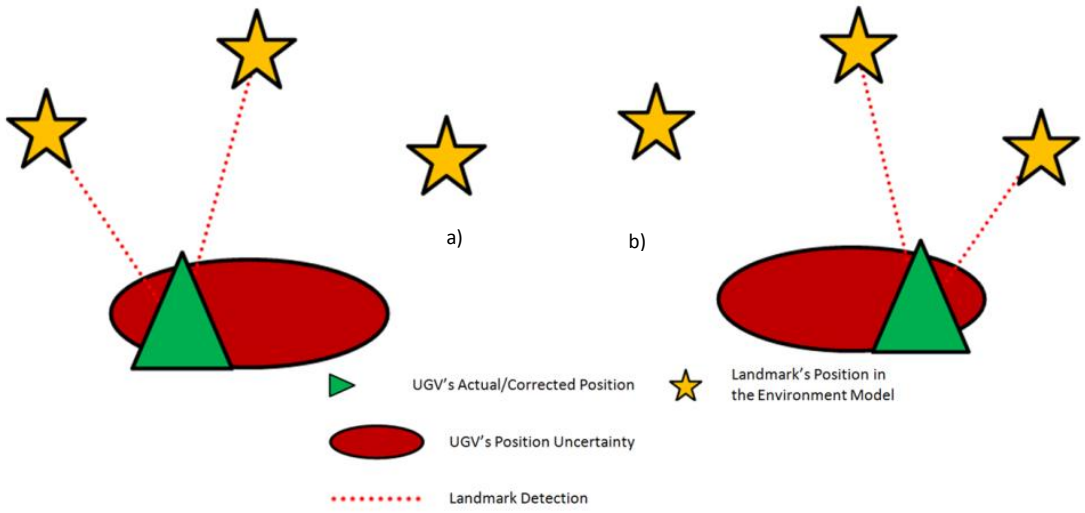

Figure 3.2 – Example of wrong Data Association Problem.

This problem is related with Data Association. Figure 3.2 represents an occurrence of wrong Data Association. The UGV has three landmarks in its Environment Model and it only detects two of them. The robot can either be found in the situation *a)* or in the situation *b)*. This kind of problem can cause the mapping system to diverge, with catastrophic consequences [Durant06].

Another important point about mapping is that landmarks should not be too far apart, because the MR will accumulate error from the odometry by itself, leading to larger error on the position of the new landmarks detected. In other words, the mapped environment needs to have, at least, a trail of landmarks so the robot will have a good performance, and also be capable to avoid getting lost.

The robot also needs to be able to distinguish stationary obstacles, such as walls and furniture, from moving obstacles, such as people or other mobile robots. When (almost) stationary obstacles are moved or altered, this must be detected and the map adjusted accordingly.

3.1.3 EKF-SLAM Algorithm

The EKF-SLAM is one of the most commonly used Mapping algorithms. As the name implies, it uses an Extended Kalman Filter (EKF) to associate the robot with the various detected landmarks on the environment. This algorithm tends to be used on the detection of features of the environment, through the robot's sensorial perception. These features are used to identify landmarks, so they will also be used for localization purposes. The EKF-SLAM is divided into two parts: the motion update, where an estimate of the AMR's position is computed; and the correction update, where the robot's position estimate, as well as the landmarks', are corrected [Durant06; Skrzyp09; Namins13; Riisga05].

On the EKF-SLAM method, the vehicle's state transition (or motion) is described by the expression

$$X_k = f(X_{k-1}, U_k) + W_k \tag{3.1}$$

where $f(\cdot)$ is the function describing the vehicle's kinematics, U_k is the control signals applied and W_k is the state's zero mean uncorrelated white noise. The description of the observation model is done in the form

$$Z_k = h(X_k, L) + V_k \tag{3.2}$$

where $h(\cdot)$ is the function that estimates the expected readings from the environment, L is the detected landmarks states and V_k is the observations' zero mean uncorrelated white observation noise. This noise is also assumed to be uncorrelated with W_k.

The estimation update is described in the following form

$$\hat{X}_{k|k-1} = f\left(\hat{X}_{k-1}, U_k\right) \tag{3.3}$$

$$P_{XX_{k|k-1}} = \nabla f \, P_{XX_{k-1}} \nabla f^T + Q_k \tag{3.4}$$

$$P_{XL_{k|k-1}} = \nabla f \, P_{XL_k} \tag{3.5}$$

$$P_{LL_{k|k-1}} = P_{LL_k} \tag{3.6}$$

where ∇f is the Jacobian of f evaluated at the estimate $\hat{X}_{k-1}$. P_{XX_k} and P_{LL_k} are, respectively, the covariance matrix on the vehicle's state and on the map. P_{XL_k} is the covariance matrix between the vehicle's state, X_k, and all landmarks, m. Q_k is the covariance of W_k.

At this point, the main concern is the estimation of the robot's position. Its "uncertainty", $P_{XX_{k|k-1}}$, affects the cross term $P_{XL_{k|k-1}}$. The map's covariance, $P_{LL_{k|k-1}}$, remains unchanged. The correction-update step is performed according to

$$\begin{bmatrix} \hat{X}_k \\ \hat{L}_k \end{bmatrix} = \begin{bmatrix} \hat{X}_{k-1} \\ \hat{L}_{k-1} \end{bmatrix} + K_k \left[Z_k - h\left(\hat{X}_{k|k-1}, \hat{L}_{k-1}\right) \right] \tag{3.7}$$

$$P_k = P_{k|k-1} - K_k \, S_k \, K_k^T \tag{3.8}$$

where

$$S_k = \nabla h \, P_{k|k-1} \, \nabla h^T + R_k \tag{3.9}$$

$$K_k = P_{k|k-1} \, \nabla h^T \, S_k^{-1} \tag{3.10}$$

$$P_k = \begin{bmatrix} P_{XX_k} & P_{XL_k} \\ P_{XL_k}^T & P_{LL_k} \end{bmatrix} \tag{3.11}$$

being ∇h the Jacobian of h evaluated at the estimate $\hat{X}_{k|k-1}$ and $\hat{L}_{k-1}$. S_k is the innovation covariance, used to compute the Kalman gain matrix, K_k, and R_k, is the covariance of V_k.

At this point the robot's poses (position and orientation) are corrected through the use of the observations made and their expected values, $\left[Z_k - h\left(\hat{X}_{k|k-1}, \hat{L}_{k-1}\right)\right]$. Also, the landmarks' positions are corrected (3.7). All of the covariance terms are updated by the equation (3.8), including the covariance between all stored landmarks, P_{LL_k}, (3.11). At this step the landmark's accuracy is improved through the use of observation data (3.9) and(3.10).

The noise covariance R_k is usually obtained through the features of the used sensors, and the noise covariance Q_k depends on the disturbances from the actuation motors, as well as on the

environment's disturbances. Due to the uncorrelated nature of the noise, the respective noise covariance matrices have a diagonal structure.

3.1.4 An alternative Algorithm

Another commonly used mapping algorithm is FastSLAM [Durant06; Bailey06; Kurt12; Calond06]. It uses a Rao-Blackwellised particle filter to perform the landmark association. It has several advantages over the EKF-SLAM, such as its robustness against ambiguous data association. It deals with this problem by replacing the wrong association with a re-sampled one. It also provides a lower estimation error, compared with the EKF-SLAM, although it presents less consistency. This inconsistency can cause trouble by leading to the filter's divergence.

3.2 Trajectory Planning

For an AMR to move around it requires a controller and a supervising system. It also needs to know its location within the surrounding environment, for which it requires a map. To know how to get from one location to another it is necessary to define a set of positions to be reached and actions to be performed. This leads to the question: How can a robot decide on the actions to perform to complete the mission? [Aranib04]

One of the required tools is Path Planning, or motion planning, which addresses the problem of setting up the robot's movement in a 2D or 3D world, which contains obstacles. The objective of this kind of planning is to determine what actions are appropriate for the robot to perform, so it can reach the envisaged state, while avoiding collision with the obstacles, among other mechanical limitations.

The essence of making such decisions can have both immediate and long term effects. The best choice to make depends on future situations and how they will be faced.

Classical path planning approaches include Potential Fields [Bruce02], Cell Decomposition [Stentz94; Aranib04], and Roadmaps [LaValle98; Dijkst59; Barraq91].

Various path decisions making algorithms exist, such as RRTs (Rapidly-Exploring Random Trees) [Bruce02; LaValle98], A* (A Star) [Hart68], and Dijkstra's algorithm [Dijkst59].

In this work, such decisions are made through a Reinforcement Learning based algorithm, similar to what the author of [Aranib04] proposes. The idea is to allow the MR to learn the path it should take to reach the desired goal.

Assuming the MR already knows its surrounding environment, by simulating the movement throughout the map, a number of alternative paths are tested. To determine which path is the best, it counts the number of actions taken. Once the simulated MR reaches the goal, it attributes a "suggestive" value to each position which was walked through, related to the viability of the path.

3.2.1 Reinforcement Learning

An important question is "What is the best set of decisions to take in order to reach the desired goal?" The RL's solution for this problem is through a set of successive experiments. From the performed experiments, the best one is chosen. The choices made are measured through rewards. An example of such way of learning is applied to animal training. If the animal performs well then it is rewarded, if it performs badly it is punished. This is also done on Reinforcement Learning through rewards and costs.

The distinguishing factor of the Reinforcement Learning is not the learning methods used, but the learning problem in hand [Sutton98].

3.2.1.1 Main Structure of Reinforcement Learning

The Reinforcement Learning system is divided into 4 elements: Policy, Reward, Value and the model of the problem. But before defining these elements, it is necessary to formalize the path planning problem. To formalize the presented problem, there are two important concepts that need to be distinguished: State and Action.

State (S_i), in RL, is a representation of a set of values or conditions which the process or its model encounters. In this case, it is the robot's position inside the mapped environment. Using a grid-based map, this division is easily made.

For a transition between states to be performed, actions or events (a_i) need to take place. These actions may have been decided by the system, like the MR's decision of turning on the present position, or by an external agent, like a foreign presence pushing the robot to a different position. At this initial stage, no external presence is added to the problem.

The Reward Function, R_w, defines the goal of the reinforcement learning problem. Generally speaking, it attributes a value for each state or state/action pair. That value is named a reward. It defines if the state/action taken is good or bad, for the aim of the RL system is to collect the maximum reward possible.

The Reward Function indicates how good a decision is in the short run, while the Value Function, V_f, indicates how good a decision is on the long run. In other words, the Reward Function indicates how much reward the system will gain if it chooses that action. The Value Function indicates the total reward the system might gain in the end if it chooses the action. Returning to the animal training example, if the animal knows it will gain a large reward, even though it will make a few misbehaviors, it will take that set of actions.

The Policy Function defines the overall behavior of the system. In other words, depending on the system's state, it chooses which action it should take, as a response to the stimulus. This may be as simple as a function or lookup table or so complex that it requires extensive computations, like in a search process. This is the function that defines the system's personality. It can be stubborn, by following the currently known best path, or curious and open minded, by walking around, looking for a better solution.

These three elements, when combined, can lead to the selection of the best sequence of actions to reach the desired goal, that is, the one yielding the largest reward.

For the System to learn how to act, it doesn't necessarily require for the actual action to be performed. This may be done through simulations using models, which is faster and safer. The model summarizes the knowledge on the actual plant. It mimics the plant behavior and the interaction with the environment. It is through the model that states and available actions are defined. The model exploitation avoids possible damage to the hardware, corruption of data and waste of time and resources.

However, the model may differ from reality, like the existence of unexpected obstacle along a planned path or some unexpected robot behavior. On these occasions, the combined use of both learning methods (simulation and practice) is certainly an advantage.

One of the challenges, when using RL, is the trade-off between exploration (search through every state and action available so it can pick the best one) and exploitation (uses what it knows to collect the best reward). It is the difference between searching the whole environment after all the existing rewards in it, wasting a lot of time, or just selecting among the already experienced state-action pair. This challenge usually reflexes on Policy Function [Sutton98].

3.2.1.2 Path planner with RL

The intention is to plan the best trajectory for the MR to take to reach the desired location. This study uses a simplistic 2D Grid Based map (Figure 3.3). Each cell is identified as a State of the system. Associated to each state there is a boolean indicating if that space is free of obstacle or not. The MR's trajectory has a starting point and a finishing point.

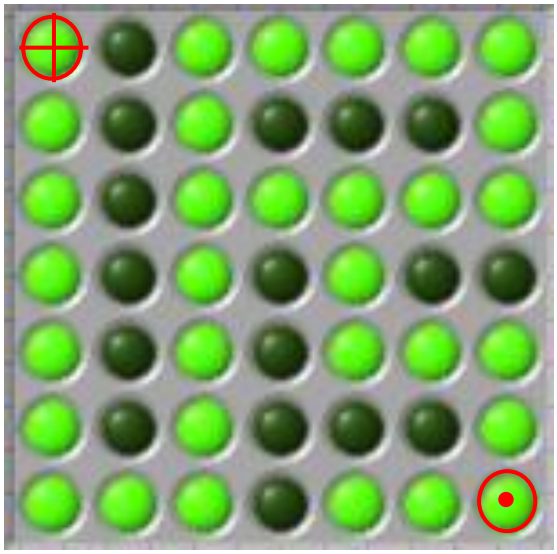

Figure 3.3 – Grid-Based Map representation.
The cross marks the starting point while the dot marks the goal.

For a MR to move from one state to another, it requires actions. In this problem model there are 4 available actions from which to take: Up, Down, Left and Right. Their availability depends if the adjacent state is occupied or not.

Reward Function

There is no difference between all states on the map, except for the finishing state, as it is the desired goal. As so, the transition to this goal state, S_g, carries the highest (positive) reward. In the present situation the value 100 is used. The remaining states are used to reach this position. If they carry a 0 reward value, that means there is no loss by walking around. In reality, for an UGV, there is always loss of time, energy and physical wear, among others. So it requires the attribution of a negative reward. In this situation it is attributed the reward value (-1) to the transition to any of the remaining states. The resulting Reward Function is

Reward Function
$R_w(S_i)$ // S_i *is the State in question.* *if(* $S_i == S_g$ *)* // *Verifies if* S_i *is the State Goal,* S_g. *return* 100*;* *else* *return* (-1)*;* *end;* *end;*

Value Function

The Value Function's array (V) indicates the best states for the MR to move to. These suggestions are based on the values each state posses on the array. The values, calculated by the Value Function, V_f, depend on the MR's present state, where it is positioned and the next state, to where it will transition to.

This function has many variations, depending on the methods used for estimating the Value Function, such as Dynamic Programming (DP), Monte Carlo (MC) and Temporal-Difference (TD) [Sutton98].

In this case a variation of the Temporal Difference Method was used, the On-line Tabular (TD(λ)) [Sutton98]. The TD Method is chosen over the other two methods mainly due to two features:

- Using values of the successive states to estimate the value of the current one. This is called bootstrapping.
- The capability of being used with a simulator (to simulate sample episodes).

The other two methods, DP and MC, only possess one of these features.

The Value Function applied is the following:

Value Function
$V_f(S_c, S_c', V, e)$ // S_c *is the Current State, while* S_c' *is the chosen next State* $\delta = R(S_c') + \gamma\, V(S_c') - V(S_c);$ // *Computation of the Error State Value Prediction* $e(S_c) = e(S_c) + 1;$ // e *is responsible for the decadence of all states* *for each State =* S_i $V(S_i) = V(S_i) + \beta\, \delta\, e(S_i);$ // *All previous visited States are* $e(S_i) = \gamma\, \lambda\, e(S_i);$ // *Degradation of the remaining States* *end;* $S_c = S_c';$ *end;*

The parameter β is the step-size parameter, γ the discount parameter and λ the decay-rate parameter. The function $e(\cdot)$ is the eligibility trace, in a way, it is the temporary record of the occurrence of an event. The values used on each of the coefficients are:

$$\gamma = 0.8; \quad \beta = 0.1; \quad \lambda = 0.1;$$

At the beginning of each episode, that is, after a series of samples beginning at the starting point and ending at the finishing point, the eligibility trace, $e(\cdot)$, is always reset to 0.

The reason for choosing this variation of the TD Method On-line Tabular TD(λ), is due to the eligibility trace array. This array keeps track of the states which are used, during the current episode, and favors them if it reaches the desired destination. This trace is used to recursively update the value assigned to each state. If it takes a long time to reach the desired goal, then the eligibility trace will favor less the first states.

Policy Function

The Policy Function is the element responsible to decide which available action to take at each state. This decision is always influenced by current and neighboring states, taking into

consideration the rewards to gain and the difference between values, which are stored within the Value Function's array (V).

A Greedy Policy has the characteristic of always choosing the best known option. This won't allow the search of new solutions, which may be better than any of the known ones. The ϵ-Greedy Policy is a compromising alternative between exploitation and exploration.

The path planner implementation uses a ϵ-Greedy Policy, π_ϵ, where the ϵ is a parameter which tunes the probabilistic selection between searching among all alternatives, and choosing the best solution.

Policy Function

```
π_ε(S_c, V, e)                          // In this situation was required to know the V and
                                        // e to make the best pick
    δ_π=[];

    for each available a of S_c
        S_c' = T(S_c, a);               // Compute the next State, knowing the action
        δ_π(a) = R(S_c') + γ V(S_c') − V(S_c);   // available, a, and the transition model T.
    end;


    if(random > ε)                      // Randomly selects either Hard or Soft Policy

        M_δ_π=MAX(δ_π);                 // Chooses the Highest Value

        if(length(M_δ_π)==1)            // There is a possibility of existing more than
        one
            return a_c ⟵ M_δ_π;
        else
            a_c is chosen at random, out of M_δ_π   // Randomly selects the highest
            return a_c
        end;

    else
        a_c is chosen at random         // Randomly selects the action among all
        available
        return a_c
    end;

end;
```

where a_c is the chosen action. The ϵ used was initialized with 0.1 and it is degraded 10^{-3} every 20 episodes. This choice for policy was made to give the possibility of exploring the surrounding states, not just being stuck with the states found by the first successful episode.

The Policy Function may either return the action chosen or the next state. Normally it returns the action so it can be applied on to the simulator.

3.2.2 Implementation of the Trajectory Planner

A program was developed to test the trajectory planning algorithm. It allows the user to define the environment map, after which it plans a trajectory to be followed by the MR. The program is implemented on National Instruments LabVIEW™. The map used on this program is a simple Grid Based map, with each cell containing the information of being occupied or not (Figure 3.3).

In RL, each cell of the map corresponds to a state. At each state 4 actions are available – move up, down, left or right. These actions will symbolize the representation of the MR's moves to the adjacent cell or state. Not all states are available, for the adjacent state might be occupied or be limited by the map border.

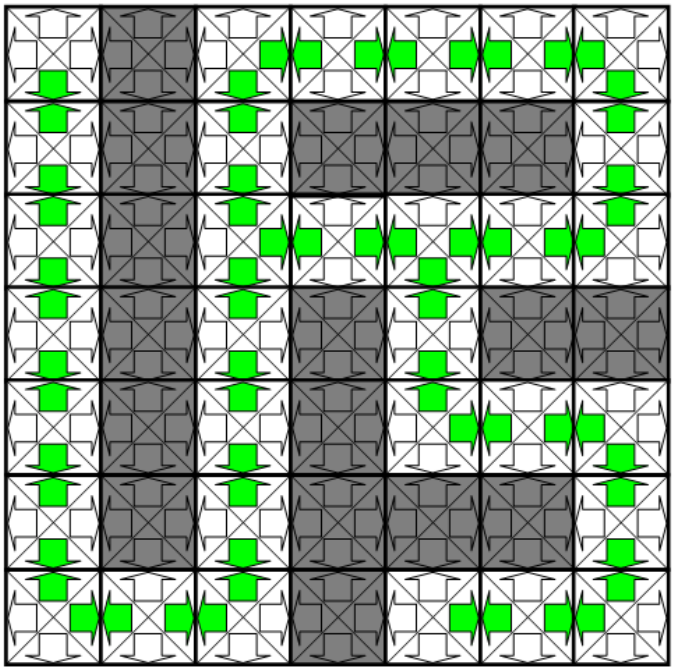

Figure 3.4 – Diagram representation of the map in Figure 3.3, containing the actions available at each cell.

For example, if the MR is in the top left state, of Figure 3.3, it can't go up or left, because there are no states (being on the border of the map) and can't go right because there is an

obstacle. This forces on taking the single action available, which is down. The available actions from the situation in Figure 3.3, results on the diagram in Figure 3.4.

On the situation proposed on the map of the Figure 3.3, the MR starts on the top left cell and has to reach the bottom right one.

3.2.3 Trajectory Smoother

The path planner chooses which states, or cells, the MR passes through, but does not provide the actual trajectory which the MR will follow.

The initial idea to depict the MR's trajectory is:

- The MR moves to reach the next state's central position.
- Once it gets there, it must rotate, facing the next state's position.
- The procedure is repeated at every state.

This method would produce a time "waste" during the MR's direction changes. The proposed procedure intends to smooth down the trajectory, by producing a faster and easy to follow trajectory. The procedure is based on the following reasoning:

A driver needs to look further ahead instead of looking directly into the front of his vehicle. This allows the driver to see the signs with more time and prepare, in advance, to follow the signs.

This idea was applied in the manner illustrated in Figure 3.5. There are 2 parameters – the observing distance, ρ, and moving distance, k_m. The observing distance has the objective to give out the direction that the MR must be heading. This direction is defined by a vector having one end on the MR's position, and the other over the line segment connecting the center of two cells. This vector has the length ρ. Knowing the orientation, provided by the observing distance, the next MR's position is computed using the moving distance k_m. This procedure was implemented over the path planner simulator.

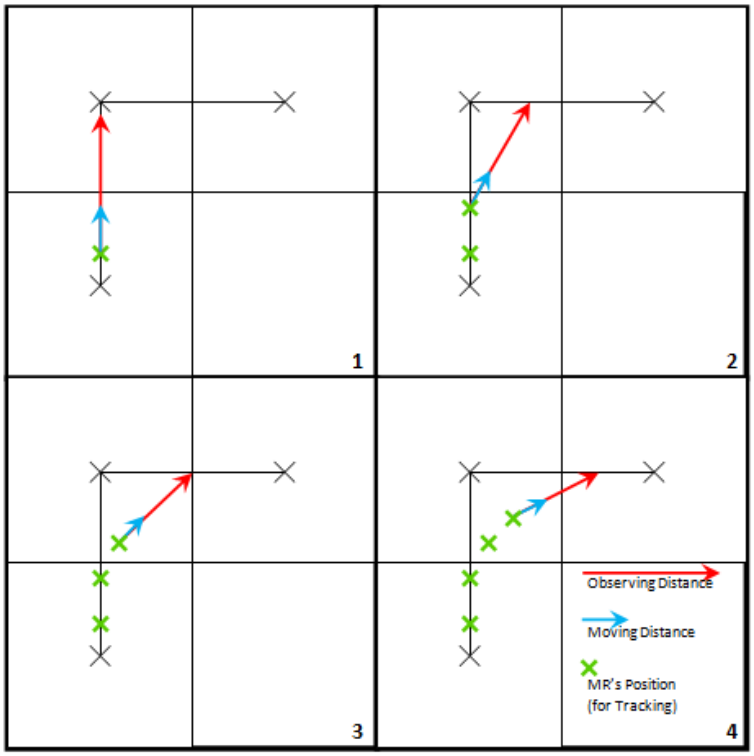

Figure 3.5 – Diagram demonstrating how the Trajectory Smoother works.

The Trajectory Smoother function is implemented as:

Trajectory Smoother
if($S_1.y == S_2.y$ *)* $Obs\_Pos.x = S_2.x;$ $Obs\_Pos.y = \sqrt{\rho^2 - (Obs\_Pos.x - Cur\_Pos.x)^2} + Cur\_Pos.y;$ *if(* $(S_2.y - S_1.y) < 0$ *)* $Obs\_Pos.y = 2\ Cur\_Pos.y - Obs\_Pos.y;$ *end;* *else* $Obs\_Pos.y = S_2.y;$ $Obs\_Pos.x = \sqrt{\rho^2 - (Obs\_Pos.y - Cur\_Pos.y)^2} + Cur\_Pos.x;$ *if(* $(S_2.x - S_1.x) < 0$ *)* $Obs\_Pos.x = 2\ Cur\_Pos.x - Obs\_Pos.x;$ *end;* *end;* $Nxt\_Pos.x = Obs\_Pos.x \quad k_m/\rho;$ $Nxt\_Pos.y = Obs\_Pos.y \quad k_m/\rho;$

\

The picture in Figure 3.6 gives a graphical representation of the result combination of both Path Planner and Trajectory Smoother.

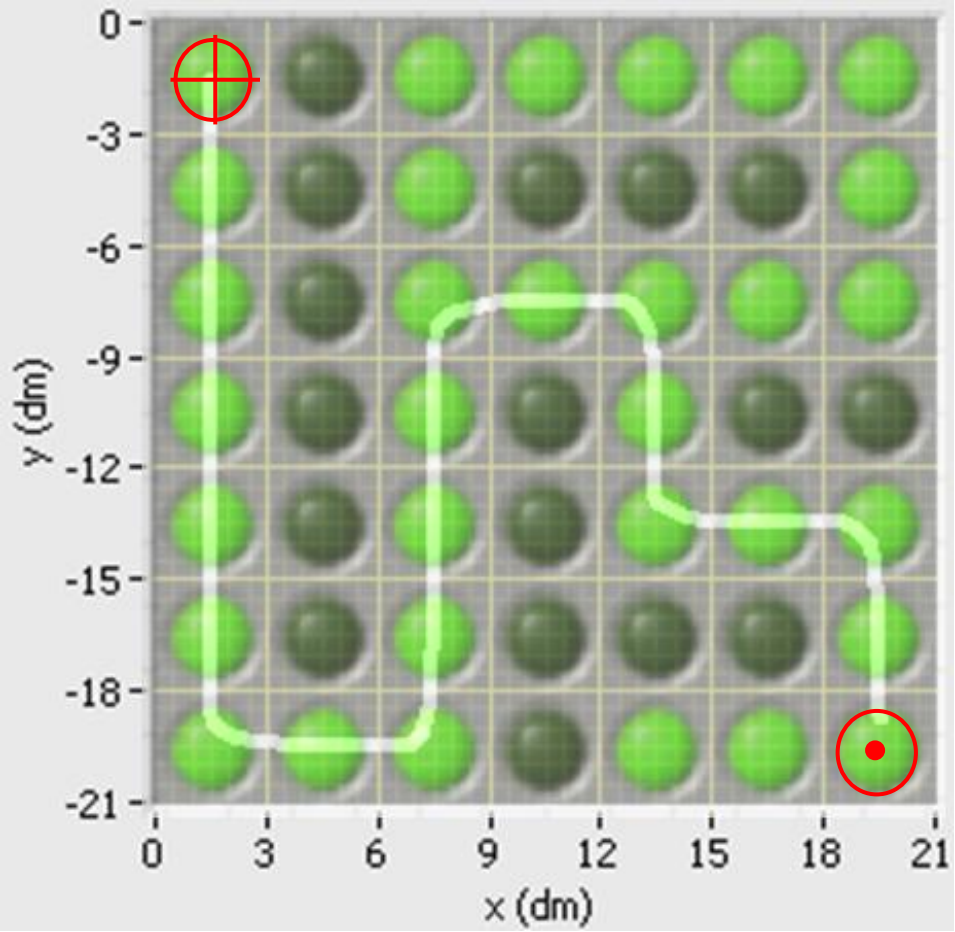

Figure 3.6 – Result of the combination of the Path Planner and Trajectory Smoother, from the situation in Figure 3.3.

Here the size of each cell (distance between two cells) is $3dm$, the observing distance, ρ, is $1dm$ and moving distance, k_m, is $0.4dm$.

A second map was used to test this trajectory planner, using the same parameterization for both the Path Planner and the Trajectory Smoother. The result obtained is presented on Figure 3.7, which is not the only path available, but is one of the best (according to the Path Planner parameters).

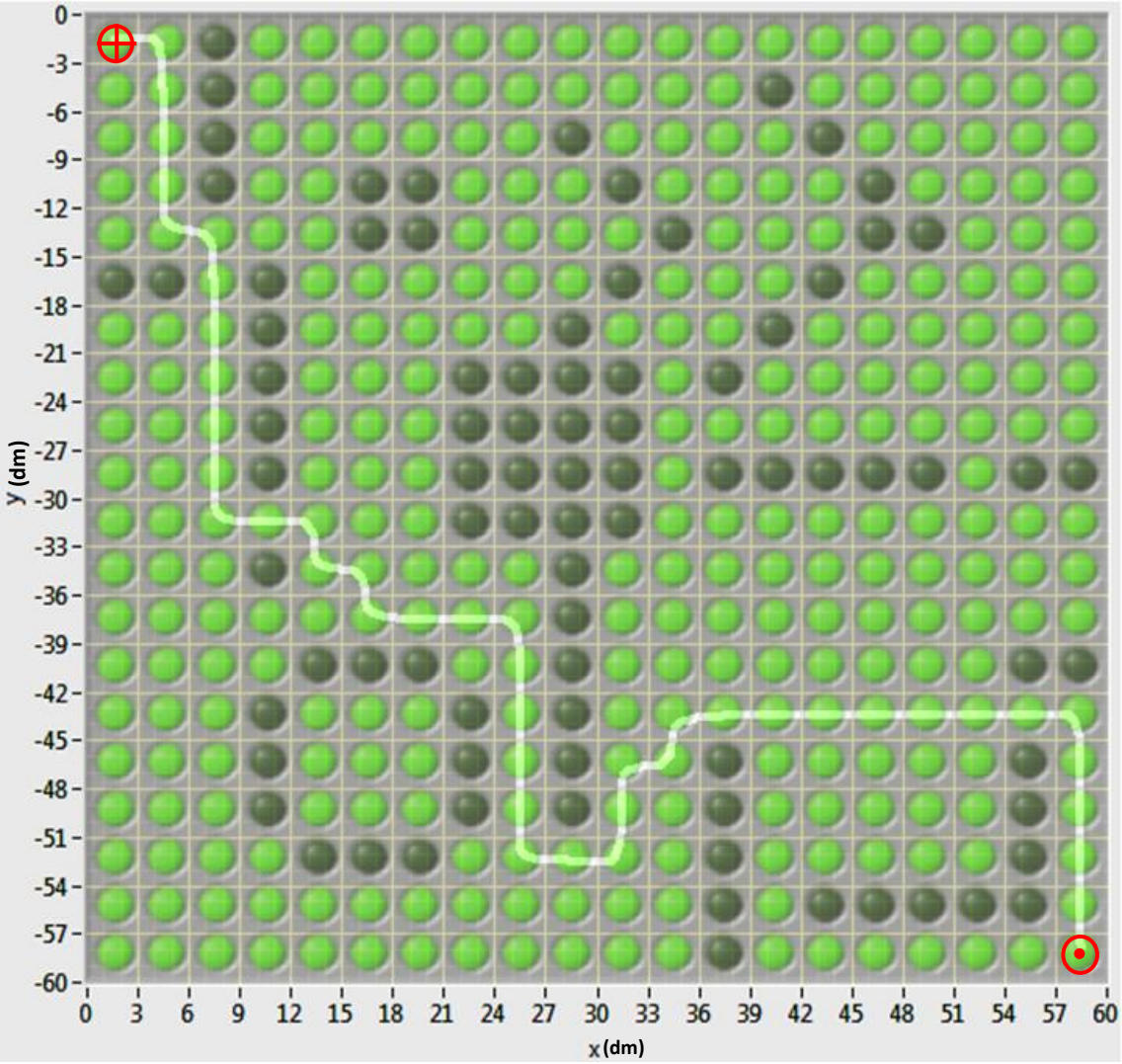

Figure 3.7 – Result of the combination of the Path Planner and Trajectory Smoother, using a more complex map.

3.3 Summary

This chapter addresses two connected problems, the mapping of the robot's surrounding environment, and the planning of the best route for robot to take on the developed map.

Mapping the working environment, in which the robot actuates on, is fundamental for the planning and execution of the movement, as well as for the determination of the MR's position. This document wouldn't be complete without the reference to this subject.

The first part of this chapter it's presented the Map's construction problem and utilization. It describes the use of the algorithm EKF-SLAM to associate the robot with the various detected landmarks on the environment.

Although no implementation of the SLAM has been done, the EKF algorithm is used for different applications on the remaining of this work.

The second part of this chapter addresses the problem of setting up the robot's path in a 2D environment which contains obstacles, and the planning of appropriate actions so that the robot can reach the goal state while avoiding collisions.

For this effect, an algorithm was developed based on reinforcement decision making. The algorithm is implemented as a simulation program, implemented in LabVIEW. The simulator was used to perform a set validation tests. Some significant results were presented.

4 Multi-rate Sensor Fusion Localization

This chapter studies the mobile robot's localization problem using on-board sensors. The data provided by different sensors is combined into consistent information on the robot's movement. The problem resulting from the lack of synchronism of the multi sensor data is also addressed.

4.1 Localization through Sensors

The number of industrial mobile robots applications has increased significantly in the last decades and its use in other indoor environments also shows a growing interest. Due to its simple mechanical structure and agility wheeled robots are still the first choices when it comes to industrial or domestic applications. Knowing exactly where a mobile entity is and monitoring its trajectory in real-time is an essential feature. Most applications, however, still rely on a more or less complex infrastructure to fulfill this task and that reduces the robot's autonomy [Kim13; Lee09; Tennina11].

The guidance system for a mobile robot is usually composed of three modules: trajectory generation and supervision system, control system and navigation system. The navigation system aims to determine reliable information on the robot's position, velocity and orientation and provide it to the other modules. Knowing the location of an autonomous mobile equipment and monitoring its trajectory in real-time is still a challenging problem that requires the combined use of a number of different sensors. The most frequently used sensors are inertial measurement units [Kim13; Lee09; Tennina11; Armest04; Bancro11; Hol07; Cho11; Marin13]. With inertial sensors the robot's position and velocity can be computed by integrating the acceleration measurements given by the sensors, but the position and the orientation errors grow over time as a result from the accumulation of the measurements' noise and bias. As computing the position from acceleration measurements requires two integration steps, the position error grows much faster than the error of orientation, which requires only one integration step. Thus, when an IMU is used for position measurements it is usually combined with another position measuring system with smaller drifting error.

To tackle with this task a number of solutions are proposed in the literature. Some authors use multiple IMUs or combine them with electromechanical odometers [Lee09; Cho11]. As all the sensors are prone to drift errors the problem is not eliminated but the precision is improved. The use of GPS as an absolute positioning system is also proposed [Bancro11; Marin13], however this is not an option for indoor operation. A number of solutions for indoor applications use active beacons as a means to determine the absolute position of the robot [Kim13; Lee09; Tennina11]. This type of approach requires the existence of a fixed infrastructure that involves additional cost and reduces the robot's autonomy.

The objective of this work is to track the position of the mobile robot. The MR can obtain position information either from its internal sensors, such as inertial measurement unit and odometer, or from external systems. Each sensor or sensory system isn't always reliable just by itself. The combination (fusion) of data from different sensors can provide more reliable information. Different sensors usually don't have the same sample-rate and may even be asynchronous.

4.2 Multi-rate Sensor Fusion

The information provided by the sensors are the forward and rotation speed (respectively v_T and v_R) of the MR and its position(x, y).

Assuming that the system dynamics can be represented by:

$$\frac{d}{dt}X(t) = \mathrm{A(t)}\,X(t) + w_t \tag{4.1}$$

where X is the system's n-dimensional state. The system dynamics is represented through the matrix A. Noise acting on the state is represented by w_t.

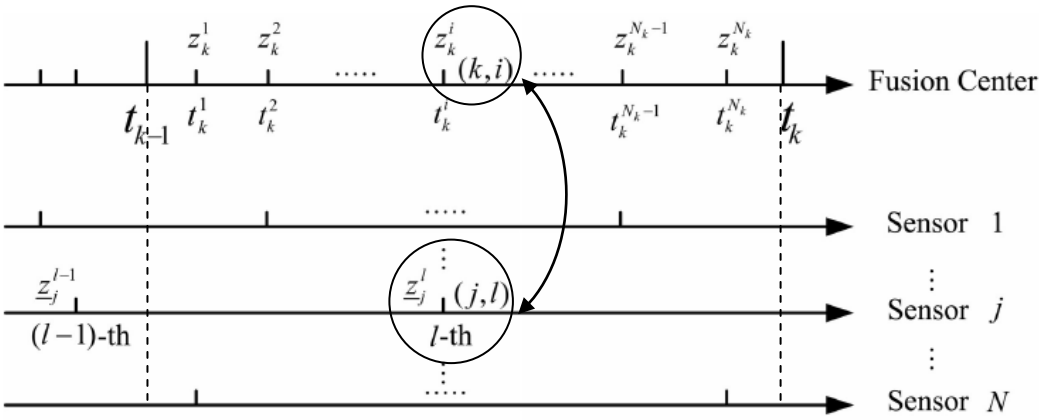

Figure 4.1 – Raw measurements between t_{k-1} and t_k.

The state is observed by a finite number of sensors. The measurements, provided by each sensor, are collected between t_{k-1} and t_k (the moments when the sensor fusion is performed).

In Figure 4.1 assume $\underline{z}_j^l$ to be the l^{th} measurement, provided by the sensor j at time t_k^i. Let z_k^i be the set of all sensor measurements collected at t_k^i, which is represented by:

$$z_k^i = H_k^i\, X\left(t_k^i\right) + v_k^i \tag{4.2}$$

where H_k^i is the output matrix associated to the respective measures and v_k^i is the measurement noise.

4.2.1 Organizing the data

The relationship between the state and the measured value is made through the output matrix H_k^i. As the measurements are not synchronous with the fusion moments, the measurement time delay must be taken into account. The state transition matrix Φ, from system (4.1), is used to adjust the time shift from t_k^i to t_k.

$$\bar{H}_k^i = H_k^i \Phi^{-1}\left(t_k, t_k^i\right) \tag{4.3}$$

$$z_k = \left[\left(z_k^1\right)^T, \left(z_k^2\right)^T, \cdots, \left(z_k^{N_k}\right)^T\right]^T \tag{4.4}$$

$$H_k = \left[\left(\bar{H}_k^1\right)^T, \left(\bar{H}_k^2\right)^T, \cdots, \left(\bar{H}_k^{N_k}\right)^T\right]^T \tag{4.5}$$

The output equation at fusion moments is given by expression (4.6), where z_k is the set of collected measurements during the time interval $(t_{k-1}, t_k]$.

$$z_k = H_k\, X(t_k) + v_k \tag{4.6}$$

The Noise Covariance Matrices R_k and Q are computed through (4.7) and (4.8).

$$R_k = \begin{bmatrix} R_k^1 + \bar{H}_k^1\, Q\left(t_k, t_k^1\right) \left(\bar{H}_k^1\right)^T & \cdots & \bar{H}_k^1\, Q\left(t_k, t_k^{N_k}\right) \left(\bar{H}_k^{N_k}\right)^T \\ \vdots & \ddots & \vdots \\ \bar{H}_k^{N_k}\, Q\left(t_k, t_k^{N_k}\right) \left(\bar{H}_k^1\right)^T & \cdots & R_k^{N_k} + \bar{H}_k^{N_k}\, Q\left(t_k, t_k^{N_k}\right) \left(\bar{H}_k^{N_k}\right)^T \end{bmatrix} \tag{4.7}$$

$$Q\left(t_k, t_k^i\right) = \int_{t_k^i}^{t_k} \Phi(t_k, \tau)\, q(\tau)\, \left(\Phi\left(t_k, t_k^i\right)\right)^T\, d\tau \tag{4.8}$$

being R_k^i the Noise Covariance Matrix of the corresponding measurement (4.2), and q the Noise Covariance Matrix of the continuous system (4.1).

4.2.2 State Estimation

The discretization of the linear continuous-time system (4.1), leads to the following representation.

$$X(t_k) = \Phi(t_k, t_{k-1}) X(t_{k-1}) + w(t_k, t_{k-1}) \tag{4.9}$$

where

$$w(t_k, t_{k-1}) = \int_{t_{k-1}}^{t_k} \Phi(t_k, \tau)\, w(\tau)\, d\tau \tag{4.10}$$

4.2.2.1 Data Fusion using Extended Kalman Filter

The Extended Kalman Filter (EKF) is used as an estimator for the MR's Position and orientation, [Armest04]. It uses the MR's dynamics, measurements, provided by the sensors, and noise properties to estimate the MR's state.

The KF is divided in 2 steps, the estimation and filtering. The estimation is where the MR's state is estimated using its dynamics, (4.11) and (4.12). While the filtering is where the state is corrected, by means of the measured values, (4.13) and (4.14).

$$\hat{X}_{k|k-1} = \Phi(t_k, t_{k-1})\, \hat{X}_{k-1} \tag{4.11}$$

$$P_{k|k-1} = \Phi(t_k, t_{k-1})\, P_{k-1} \Phi(t_k, t_{k-1}) + Q(t_k, t_{k-1}) \tag{4.12}$$

$$\begin{aligned} &(P_k)^{-1} = \left(P_{k|k-1}\right)^{-1} + \\ &+ \left[H_k + \Lambda_k^T \left(P_{k|k-1}\right)^{-1}\right]^T \left[R_k - \Lambda_k^T \left(P_{k|k-1}\right)^{-1} S_k\right]^{-1} \left[H_k + \Lambda_k^T \left(P_{k|k-1}\right)^{-1}\right] \end{aligned} \tag{4.13}$$

$$\begin{aligned} &\hat{X}_k = \hat{X}_{k|k-1} + \\ &+ P_k^f \left[H_k + \Lambda_k^T \left(P_{k|k-1}\right)^{-1}\right]^T \left[R_k - \Lambda_k^T \left(P_{k|k-1}\right)^{-1} \Lambda_k\right]^{-1} \left(z_k - H_k\, \hat{X}_{k|k-1}\right) \end{aligned} \tag{4.14}$$

where $\hat{X}_{k|k-1}$ and $\hat{X}_k$ are the estimated state and the filtered state at t_k, respectively, and $P_{k|k-1}$ and P_k are the estimated and filtered error covariance matrix at t_k.The Noise correlation Matrix, Λ_k, is given by (4.15).

$$\Lambda_k = \left[-Q\left(t_k, t_k^1\right)\left(\overline{H}_k^1\right)^T, \ldots, -Q\left(t_k, t_k^{N_k}\right)\left(\overline{H}_k^{N_k}\right)^T\right] \tag{4.15}$$

4.2.2.2 Fusing Data

There are two distinct types of time instants, the time when a certain sensor is read and its measure is recorded, and the fusion time when all the measurements collected are "fused".

Each sensor has its own set of sample moments, t_k^i, and its output matrix, H_k^i. Between the fusion times, t_k and t_{k-1}, each sensor stores the measures read, z_k^i, the time when each was

taken, t_k^i, and the respective output matrix, H_k^i. As each sensor has its own sample rate, it is also necessary to record the output matrix H_k^i. The different sample rate also means that the augmented measurement vector, z_k, and the augmented output matrix, H_k, may change size. So does the rest of the matrices that are related, such as Λ_k and R_k.

As the sensor fusion may vary the number of measured values, one may wonder if it is better to apply a fusion time interval as large as the slowest sensor (capturing all measurements), or as short as the fastest sensor (capturing at most a single measure). If it is the slowest it may obtain a more accurate state of the MR, but it may take longer than needed, and it won't trace the whole trajectory, but only the position of the last measurement read. If it is the fastest, then it may trace the whole path, but due to single sensor inaccuracy, it may go off-track most of the times. The key is to perform the sensor fusion when it is required to know the MR's state.

This is assuming that the fusion time interval is large enough to have at least one measurement stored. When the control of the MR requires a faster "supposition" of its position (so fast that it doesn't allow any of the sensors to store information), the only way to perform is to use the last information that was provided, z_{k-1}. This estimation gives a little more relevance to the dynamics of the model. Such a solution was mentioned on [Armest04].

On the next section three cases will be compared:

- With fusion time equal to the fastest sensor;
- With fusion time faster than the fastest sensor;
- With fusion time slower than the fastest sensor.

4.3 Off-Line Analysis

The algorithm was implemented into a program. This program aims to test the algorithm using the data from experiments.

The program is divided in two parts, the Data Collection and the Data Fusion. The Data Collection consists of collecting the data provided by each sensor, with their respective sample rates. The Data Fusion combines all the information collected by the sensors and estimates the

MR's actual position and direction. The information used in these experiments is the travel and rotation speed (v_T and v_r), obtained from the MR's odometer and the gyro sensor from the IMU. The MR's "actual" positions are measured at the beginning and end of each trajectory segment (x and y) as well as the direction the MR is facing (R). The MR's "actual position will be used on the data fusion, but the orientation will not, though it could have been included.

In this project, two distinct MRs were used on the experiments. The first (MR1) is a *Pitsco* 4-wheeled robot, equipped with a single-board RIO (*NI Robotics Starter Kit*). The second (MR2) is a *LEGO NXT Mindstorm* 2-wheeled robot. On both MRs is mounted the same IMU Sensor and both performed different trajectories.

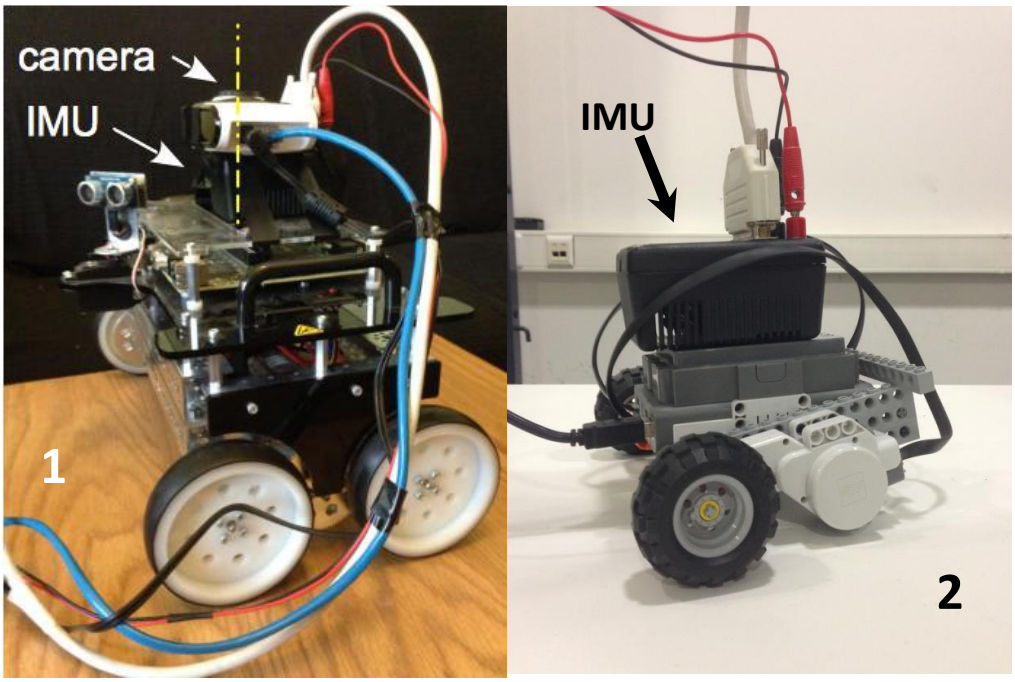

Figure 4.2 – Photos of the used MRs. 1 – MR1, Pitsco 4-wheeled robot;
2 – MR2, Lego NXT Mindstorm, 2-wheeled robot.

Due to software license problems, at this moment the robot MR1 cannot acquire the IMU's data. So the control experiments need to be performed with another robot (MR2), able to extract and use the information directly from the IMU.

The data used on the following experiments, were provided by de sensors installed on the MRs, which were controlled manually.

4.3.1 MR's Dynamics

The System model considers the following variables: forward speed, v_T, rotation speed, v_R, Orientation, R and position, x and y. The non-linearized dynamics used for the MR (4.1) on a certain moment are:

$$\frac{d}{dt}x(t) = v_T(t)\ \cos\big(R(t)\big) \tag{4.16}$$

$$\frac{d}{dt}y(t) = v_T(t)\ \sin\big(R(t)\big) \tag{4.17}$$

The linearization and discretization of this model corresponds to the following equations

$$\begin{aligned} &x(t_k) = x(t_{k-1}) + v_T(t_{k-1})\cos\big(R(t_{k-1})\big)\ (t_k - t_{k-1}) - \\ &-v_R(t_{k-1})\ v_T(t_{k-1})\sin\big(\mathrm{R}(\mathrm{t_{k-1}})\big)\ (\mathrm{t_k} - \mathrm{t_{k-1}})^2 \end{aligned} \tag{4.18}$$

$$\begin{aligned} &y(t_k) = y(t_{k-1}) + v_T(t_{k-1})\sin\big(R(t_{k-1})\big)\ (t_k - t_{k-1}) + \\ &+v_R(t_{k-1})\ v_T(t_{k-1})\cos\big(\mathrm{R}(\mathrm{t_{k-1}})\big)\ (\mathrm{t_k} - \mathrm{t_{k-1}})^2 \end{aligned} \tag{4.19}$$

The model also uses as additional state variable v_T, v_R and R, leading to a state vector: $X = [v_T, v_R, R, x, y]^T$. The resulting state transition matrix, between two consecutive state samples, is

$$\begin{aligned} &\Phi(\mathrm{t_k}, \mathrm{t_{k-1}}) = \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & (t_k - t_{k-1}) & 1 & 0 & 0 \\ \cos\big(R(t_{k-1})\big)\ (t_k - t_{k-1}) & -v_T(t_{k-1})\sin\big(R(t_{k-1})\big)\ (t_k - t_{k-1})^2 & 0 & 1 & 0 \\ \sin\big(R(t_{k-1})\big)\ (t_k - t_{k-1}) & v_T(t_{k-1})\cos\big(R(t_{k-1})\big)\ (t_k - t_{k-1})^2 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{4.20}$$

The 4 measured variables are v_T, v_R, x and y, leading to the output expression:

$$\begin{bmatrix} v_T(t_k) \\ v_R(t_k) \\ x(t_k) \\ y(t_k) \end{bmatrix} = \begin{bmatrix} H_k^{v_T} \\ H_k^{v_R} \\ H_k^{x} \\ H_k^{y} \end{bmatrix} \hat{X}_k = \begin{bmatrix} [1 & 0 & 0 & 0 & 0] \\ [0 & 1 & 0 & 0 & 0] \\ [0 & 0 & 0 & 1 & 0] \\ [0 & 0 & 0 & 0 & 1] \end{bmatrix} \hat{X}_k \quad (4.21)$$

4.3.2 MR1's main trajectory

On the experiments with MR1, an external computer was used to record the information from the odometry and IMU. During this experiment the robot's trajectory was marked. At the end of the experiment, the marked trajectory was measured.

From the off-line position measurements and the commands given to MR1, Figure 4.3 and Figure 4.4 are the expected tracking results. The MR's initial position is always $x = 0, y = 0$ and its orientation is $0°$ (parallel to the horizontal axis, x).

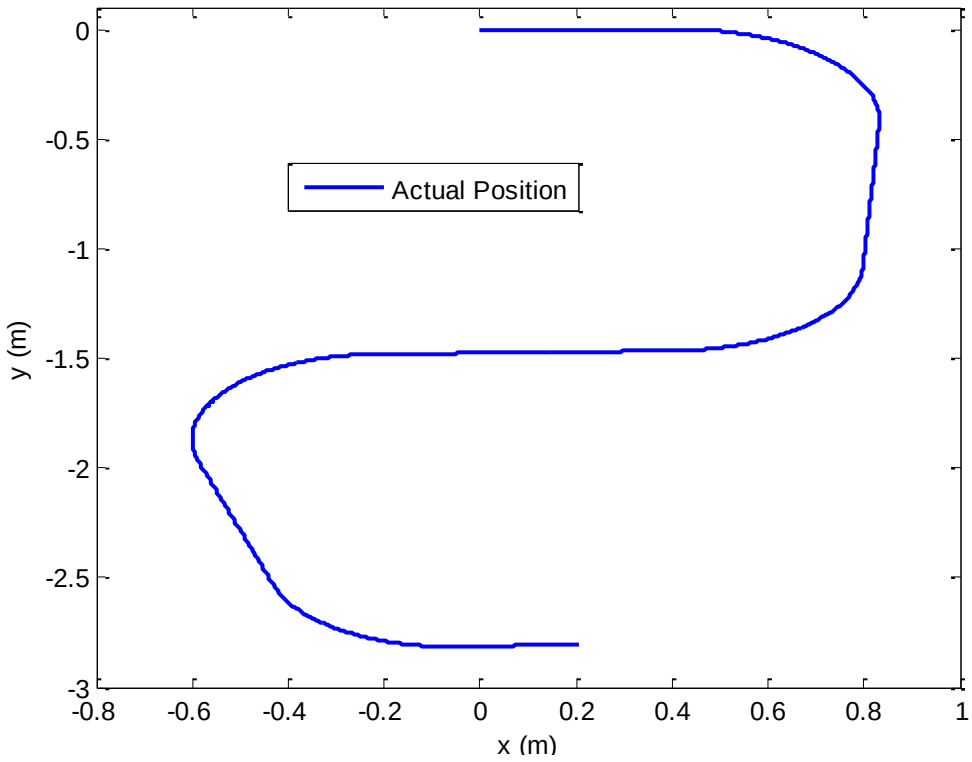

Figure 4.3 – MR1's Actual Trajectory.

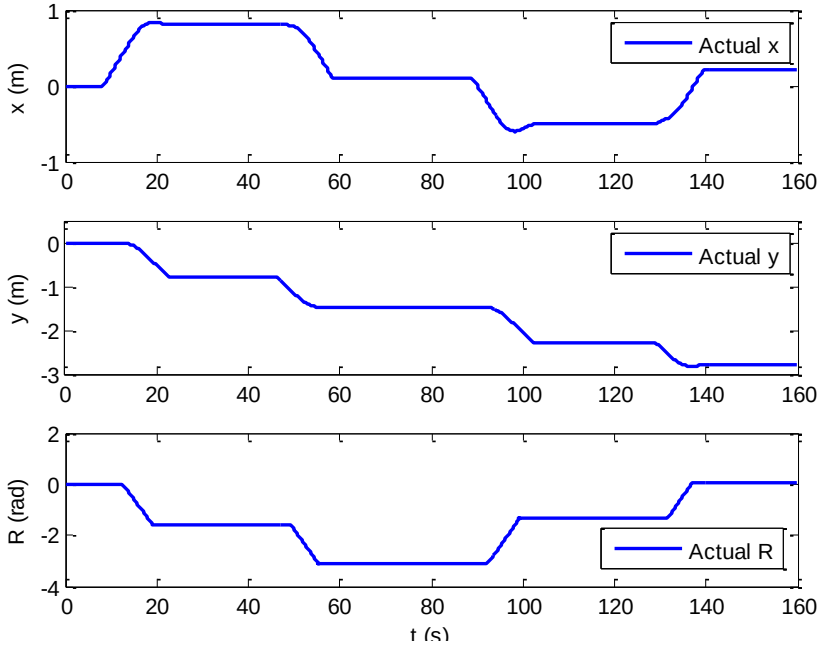

Figure 4.4 – MR1's Position and Direction.

4.3.2.1 MR1's experiment 1

The experimental data collected with MR1 is used off-line to evaluate the data fusion algorithm. The values of v_T and v_R, as well as the Data Fusion timing share the same sample rate. The MR1's real position is sampled with a large interval (when the MR is still its position was measured). Data Collection and Data Fusion were preformed separately on these experiments.

The time samples used were:

- for the v_T and v_R, 0.04s;
- for the Data Fusion, 0.04s;
- for the position (x, y), 40.00s.

The true position has a sampling rate of 40s, coinciding with the instants when the MR is immobile. The 0.04s sampling period used for v_T and v_R, is the shortest for which the used equipment produces a consistent sampling rate.

The output matrix used for the speeds (v_T and v_R, respectively) is:

$$H_k^i = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

The output matrix used for the position (x and y, respectively) is:

$$H_k^i = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

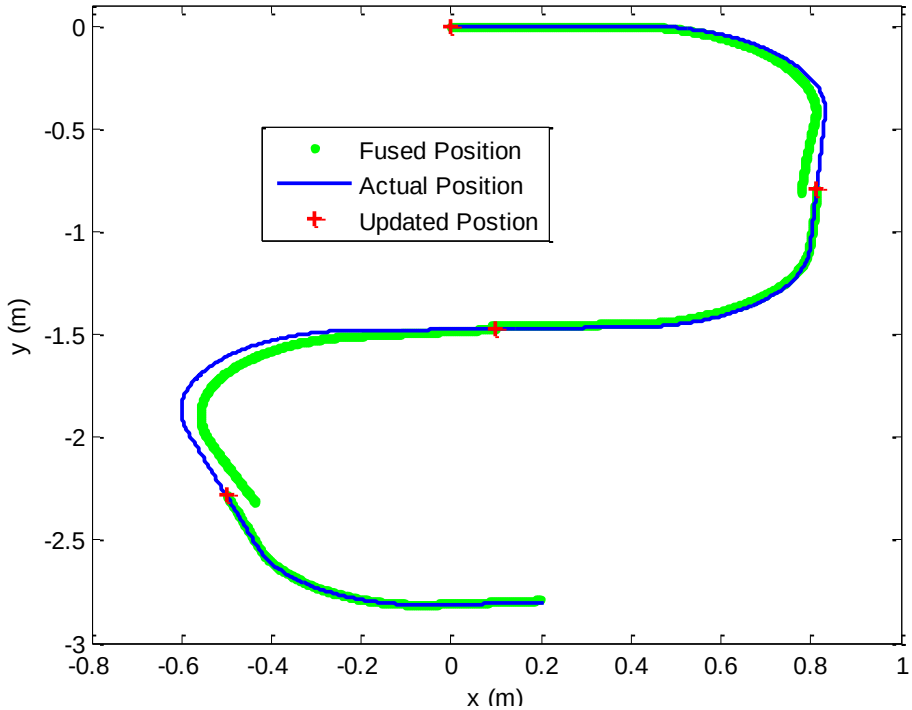

Figure 4.5 – Comparison between the "actual" position and the estimated position.

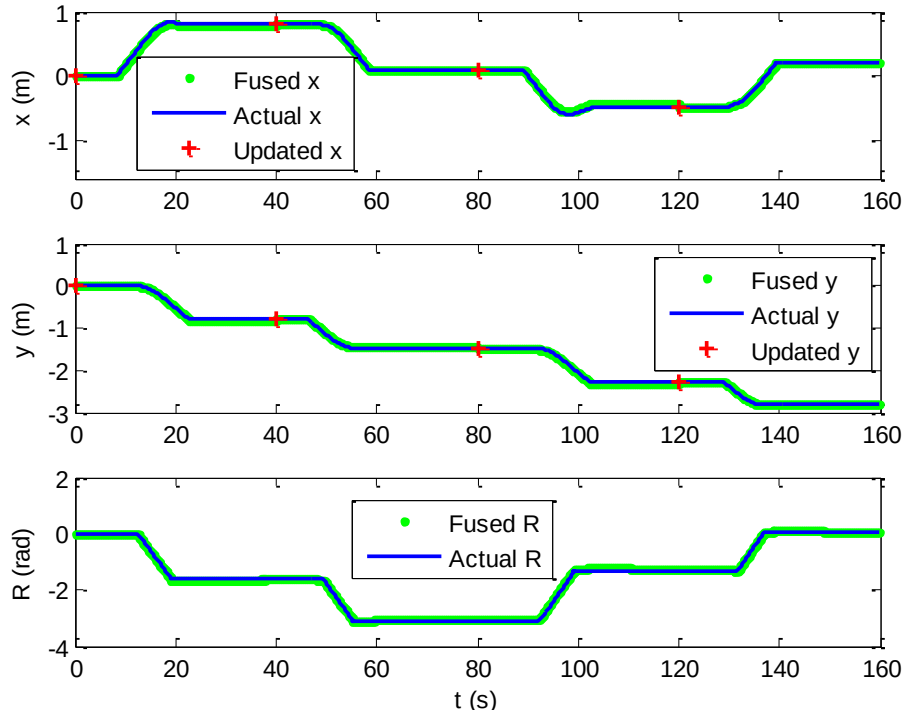

Figure 4.6 – Comparison between the "actual" position and direction and the estimated position and direction.

The position measurement provided by the sensor fusion system gives good results over both straight line trajectories and sections with larger rotation radius. For sections with shorter rotation radius the error increases. This leads us to suspect that the gyro calibration may present some small deviation.

Taking into account that during the test the sensor fusion system receives only three measurements from the actual robot position, the overall result is to be considered good.

In this case, to reduce the error of the estimated position, a larger number of updated positions should be used, mainly on the tighter curves (between 0s and 40s and between 80s and 120s).

4.3.2.2 MR1's experiment 2

Enlarging the Data Fusion interval allows the use of more information from sensors at each fusion step. The downside is the reduction of the sample rate from the MR's state estimate. This experiment was performed using the same sensor data that was collected for the previous experiment (Experiment 1).The difference stems from the use of Data Fusion interval that is five times larger than the previous.

The sampling rates used are:

- for the v_T and v_R, $0.04s$;
- for the Data Fusion, $0.20s$;
- for the position (x, y), $40.00s$.

At each fusion moment the temporal update of the data is done according to expressions (4.3) - (**4.6**). The prediction (4.11) uses the transition matrix (4.20) which depends on the orientation R. The test shows that the use of a larger fusion time step decreases the accuracy, especially on the curves. This results from the fact that (4.20) depends of the true value of R.

The position update does not include the robot orientation R. This causes a loss of accuracy, as it is seen in the last trajectory segment (from $120s$ up to the end). If R was adequately updated this last segment would suffer a rotation and the estimated trajectory would be closer to the true one.

The comparison between the first two tests shows that a reduction on the Data Fusion interval can improve the position estimate.

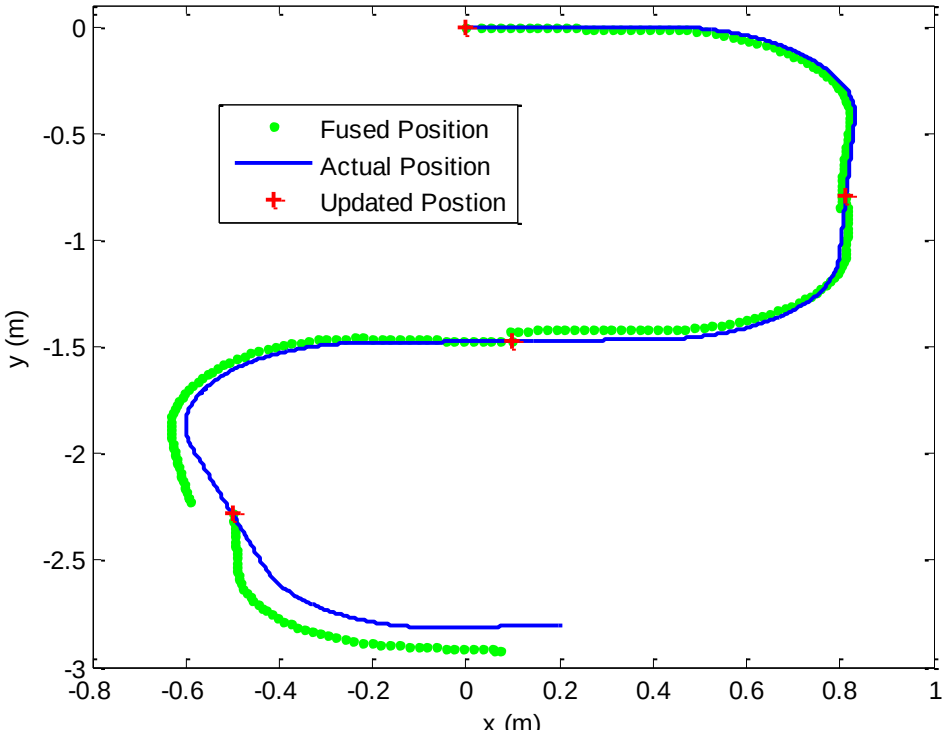

Figure 4.7 – Comparison between the "actual" position and the estimated position.

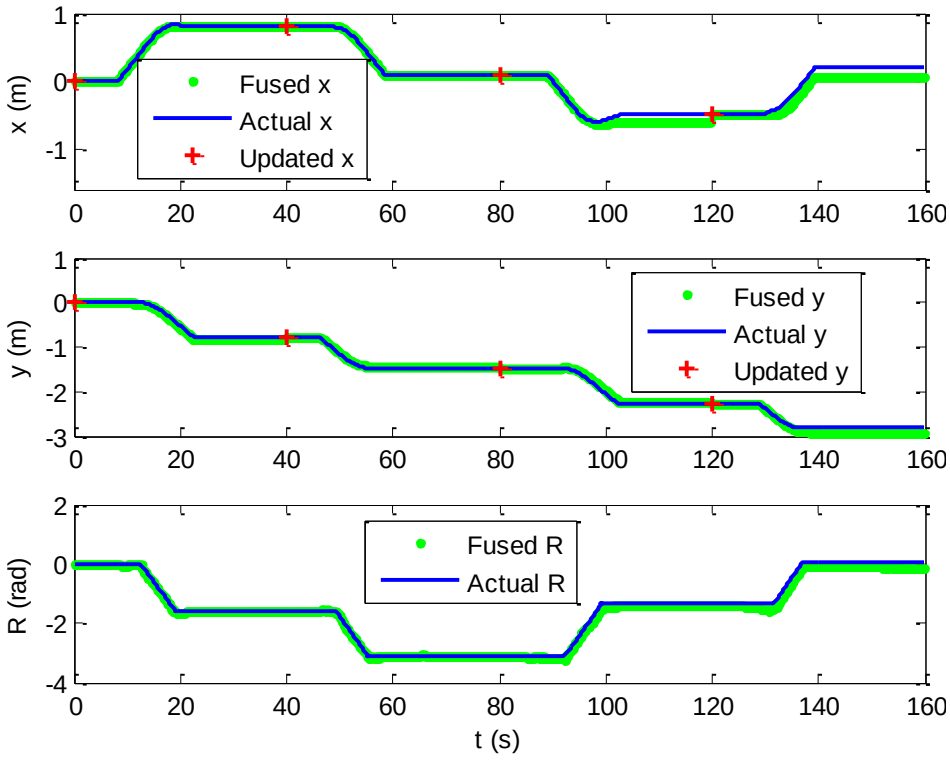

Figure 4.8 – Comparison between the "actual" position and direction and the estimated position and direction.

4.3.2.3 MR1's experiment 3

A Data Fusion step may also occur without any new information available. In this case this case the state estimate includes only a prediction step (4.11). This experiment uses the same data and fusion time step from Experiment 1. However a larger sample time is assumed for v_T and v_R. For these signals the time between samples is five times larger. This is done through decimation of the experimental data.

The sampling rates used are:

- for the v_T and v_R, $0.20s$;
- for the Data Fusion, $0.04s$;
- for the position (x, y), $40.00s$.

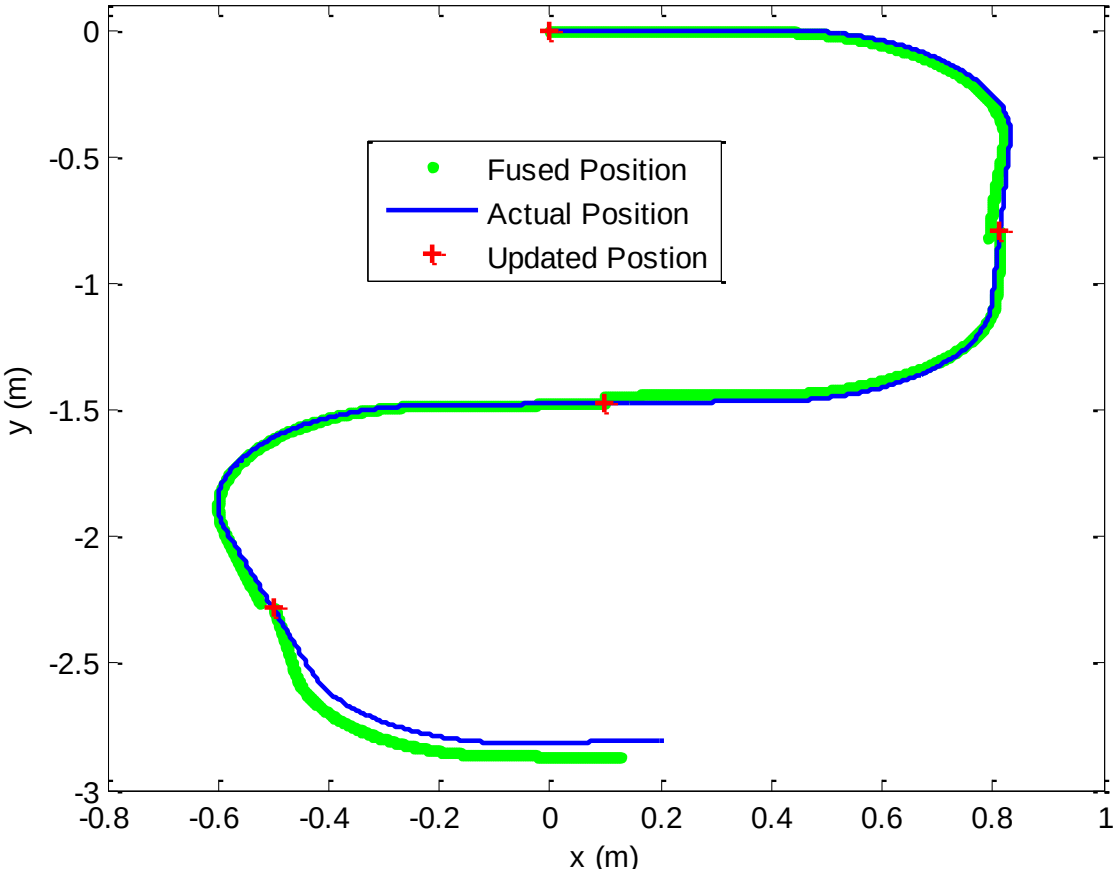

Figure 4.9 – Comparison between the "actual" position and the estimated position.

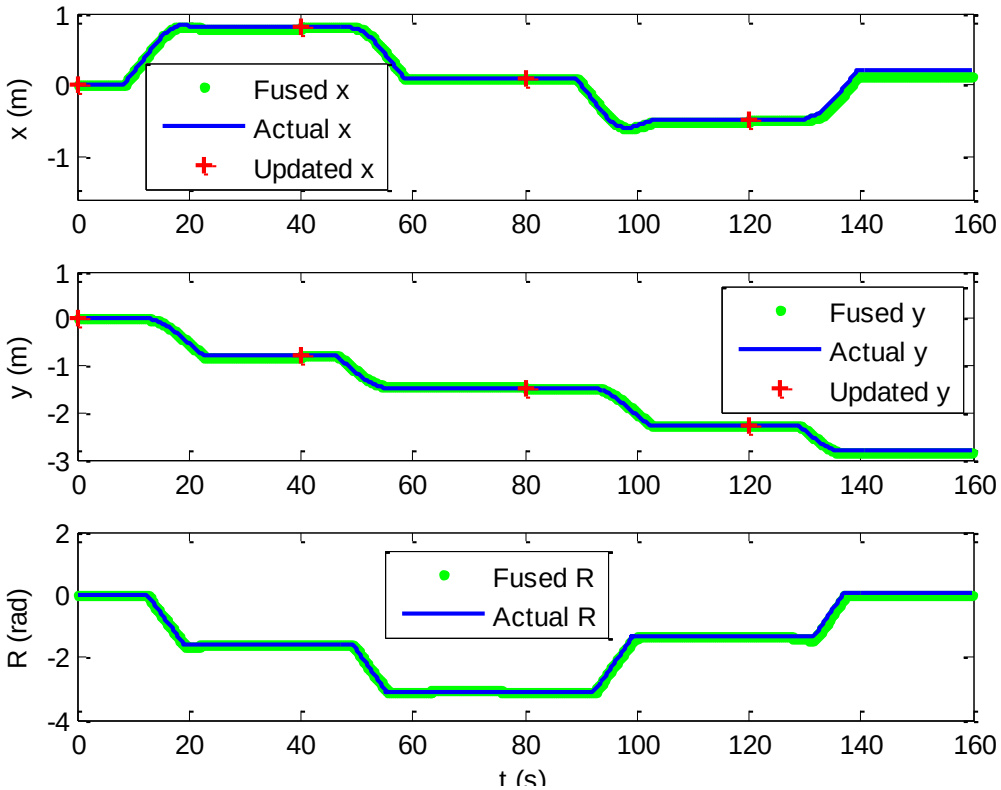

Figure 4.10 – Comparison between the "actual" position and direction and the estimated position and direction.

As the Data Fusion is faster than any sensor, several consecutive fusion moments take place without new measures being available. In this case only the prediction step of the EKF is computed.

Comparing this experiment with the previous experiment (experiment 2), the use of a shorter fusion time step, allows for a faster update of the transition matrix (4.20). As a result the tracking of the actual trajectory is better than in experiment 2. As in experiment 2 it may also be noticed the error becomes larger during the last trajectory segment. This stems from the fact that the robot orientation *R* is not updated from an external source during the test.

4.3.2.4 Comparing Experiment 1, 2 and 3

All the experiments show the importance of external sensors to correct the MR's position. Without them, the error of the MR's position would keep increasing, although, in these three experiments, it wouldn't be too far off.

The result of the difference between the Fused Position and the Actual Position of each experiment is shown in Figure 4.11. It is verified that the experiments 1 and 3 show better results than the second experiment. This occurrence might be due to the error increase in MR's orientation (R), that is visible from the instant 40s up to the end.

Both experiments 1 and 3 showed close results, except for the last trajectory segment (from the instant 120s up to the end), possibly due to the error on the MR's Orientation.

The average value of Distance error in *experiment 1* is $0.0234m$, in *experiment 2* is $0.654m$ and in *experiment 3* is $0.0344m$. While the rotation in *experiment 1* is $0.0286rad$, in *experiment 2* is -0.593rad and in *experiment 3* is -0.018rad.

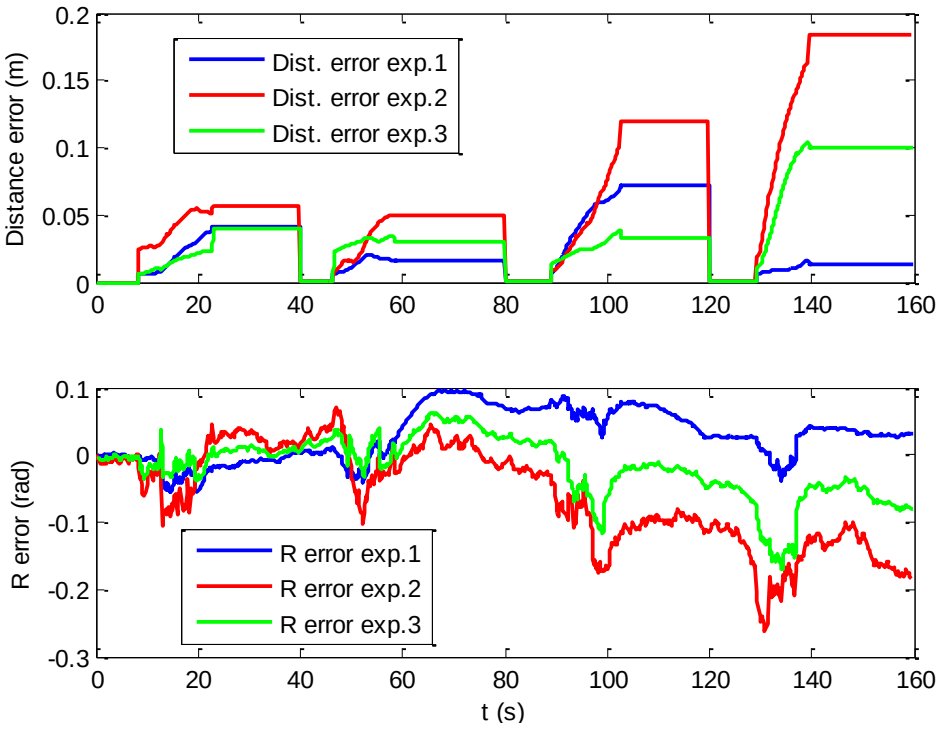

Figure 4.11 – Comparison between the Data Fusion based on Velocity and on Estimation.

4.3.3 MR2's performed trajectories

Three distinct trajectories were performed for the MR2. The robot's position was measured when the actuation set point changed. This measured position is used on the sensor fusion experiments. On these trajectories fewer points were used specially while performing curves.

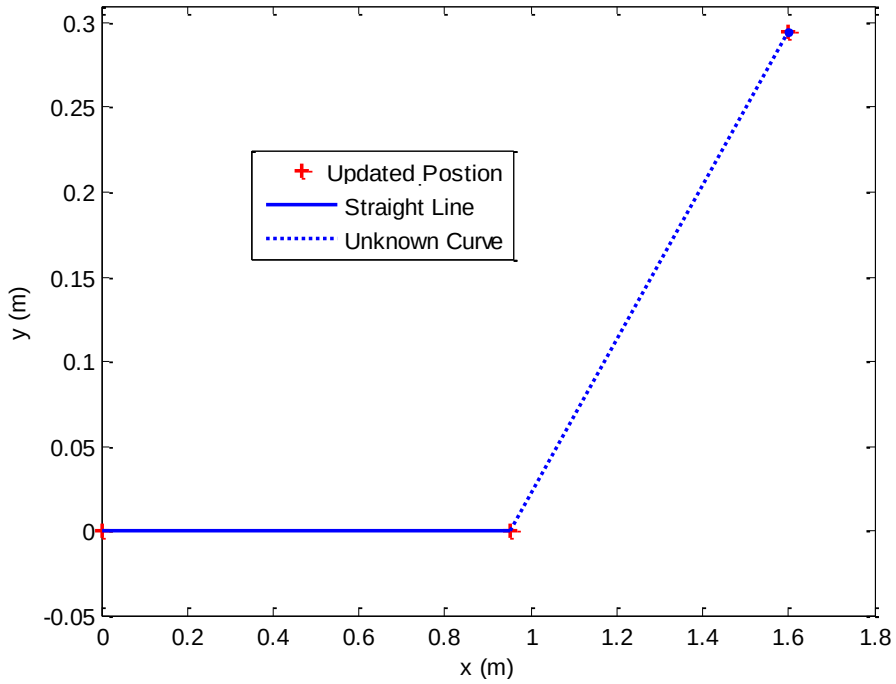

Figure 4.12 – MR2's expected Trajectory 1, with the actual positions.

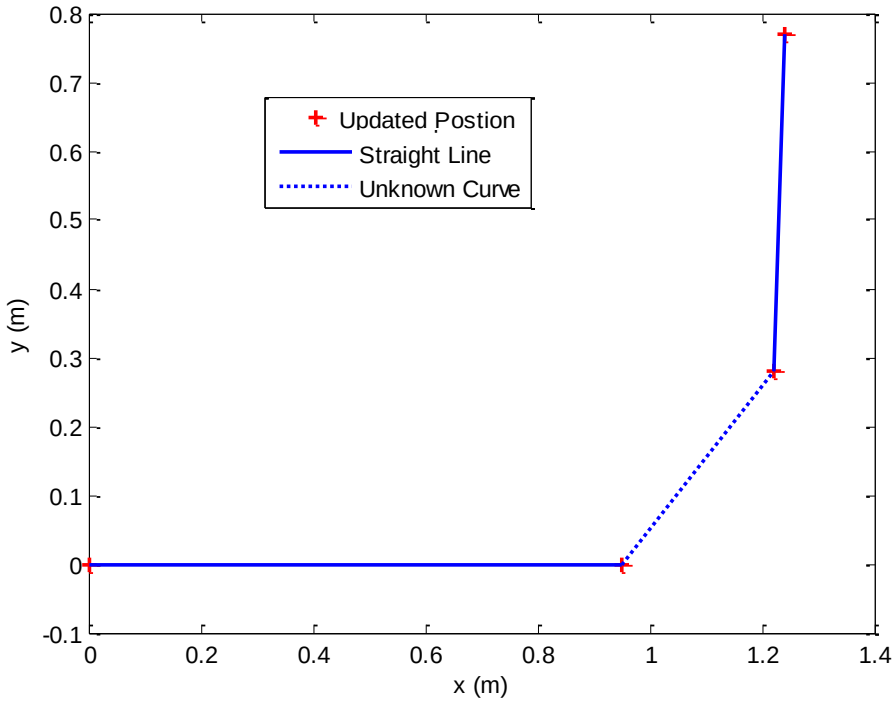

Figure 4.13 – MR2's Expected Trajectory 2, with the actual positions.

4.3.3.1 MR2's experiment 1

This experiment was performed using the data acquired on the MR2's first trajectory (Figure 4.12). The results obtained are presented in Figure 4.14 and Figure 4.15.

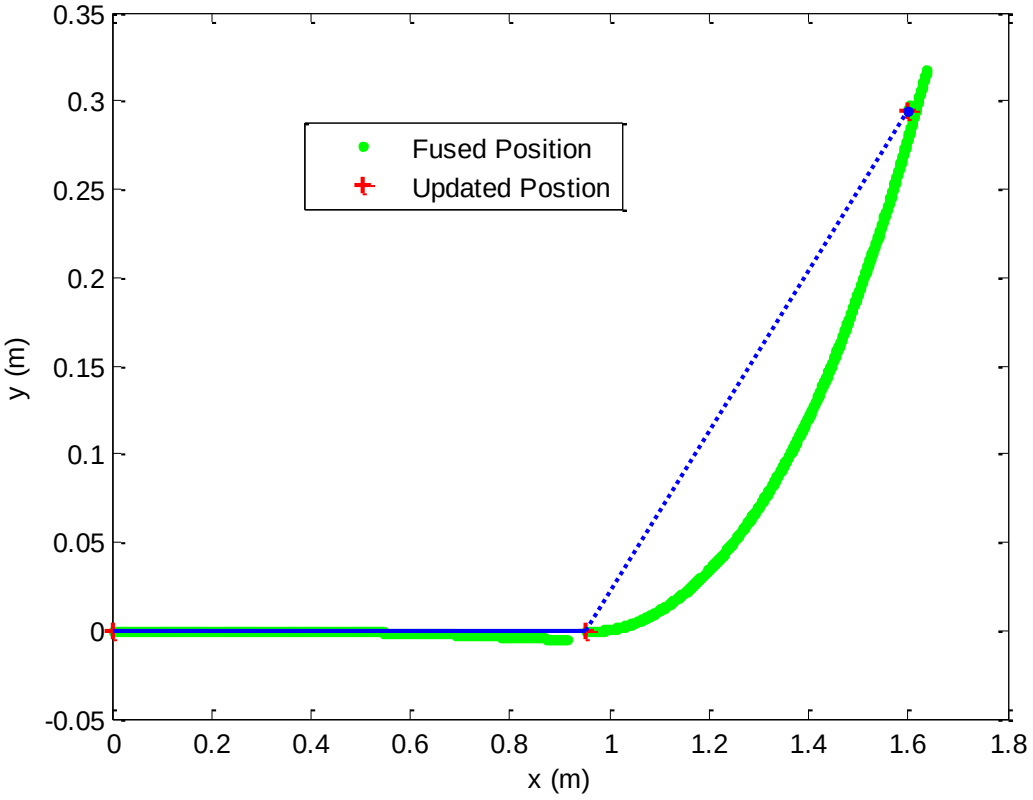

Figure 4.14 – Comparison between the "actual" position and the estimated position.

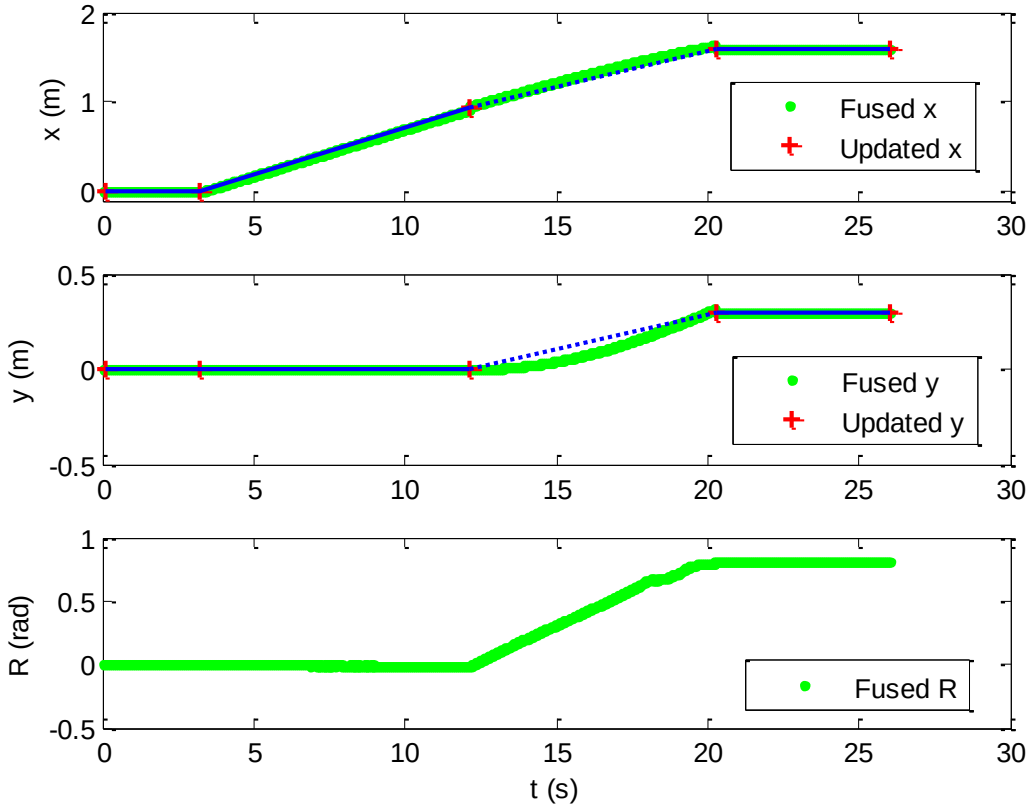

Figure 4.15 – Comparison between the "actual" position and direction and the estimated position and direction.

This second MR is lighter than MR1, which leads to the possibility of lack of friction force between the wheels and the ground, causing the MR to estimate its position further than it actually is. The cables connected to the MR, and to the IMU, may also have some effect on the robot displacement. Both these problems were minimized on the reported experiments.

4.3.3.2 MR2's experiment 2

This experiment uses the data acquired with MR2 performing trajectory 2 (Figure 4.13). The results obtained are presented in Figure 4.16 and Figure 4.17.

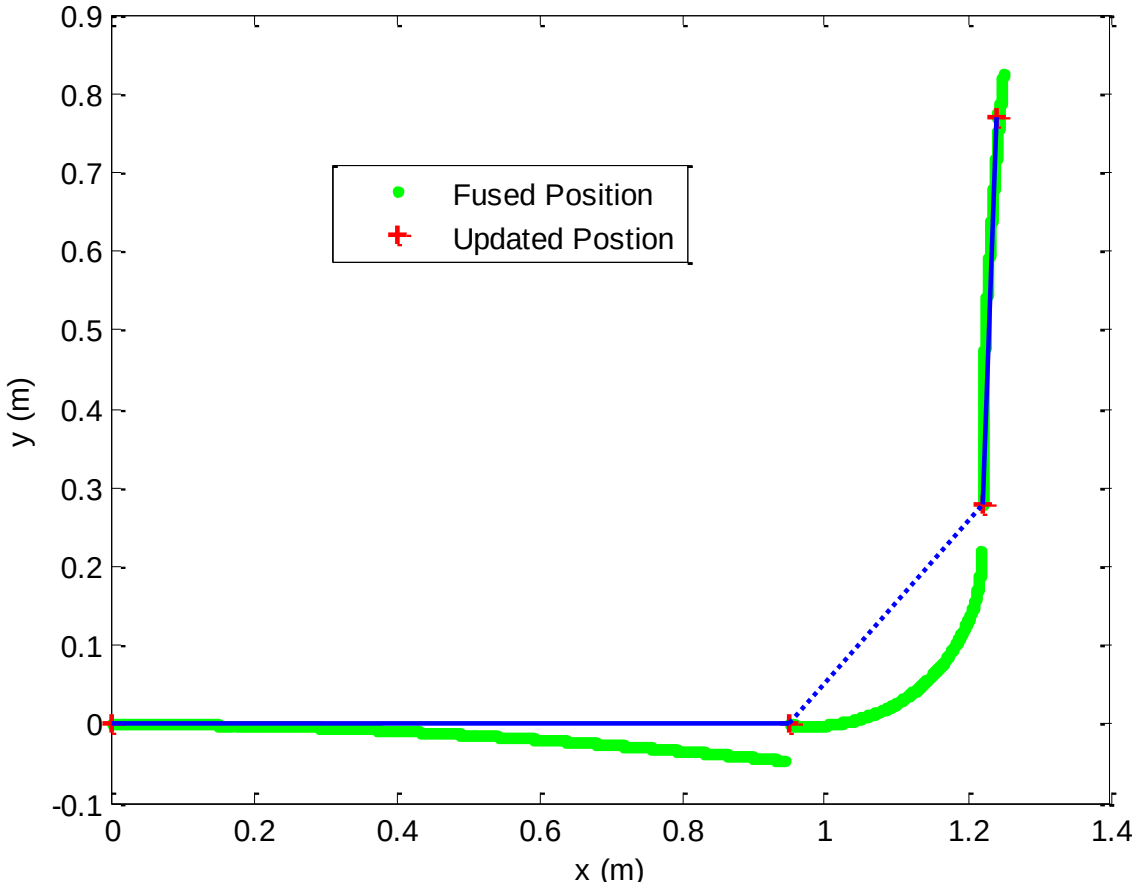

Figure 4.16 – Comparison between the "actual" position and the estimated position.

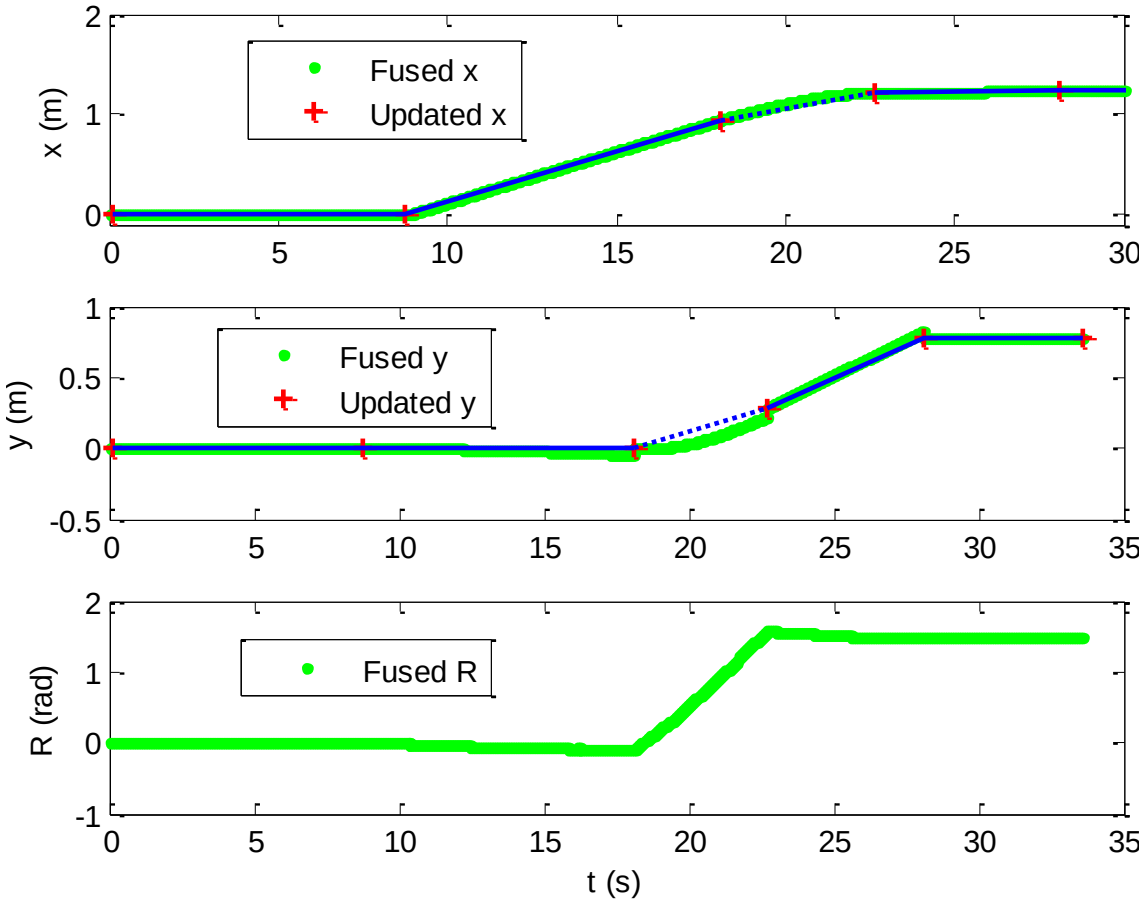

Figure 4.17 – Comparison between the "actual" position and direction and the estimated position and direction.

On the initial straight line, it can be verified that the MR was drifting to the right, which could mean one of two things. Either the MR started with a different direction (in this case $R > 0°$) and one of the wheels was "sliding", or the IMU reading contained error (this last option it is less likely, reviewing the previously obtained data).

4.4 Summary

The proposed sensor fusion method is aimed to keep estimating the MR's position, even when it does not possess all the sensors' information at the same time, making it able to be applied directly on an MR. All the experiments presented were performed off-line but using real data collected from the robots. Applying it in real-time maybe ill-advised if one is fusing too

many measurements, making the Data Fusion to take longer than expected. If the fusion time step is much larger than the fastest sensor sample time, this causes a large computational burden to the system.

The two MRs that were used are significantly different. As MR2 is smaller and lighter and its wheels more prone to slide than of the MR1's, this causes larger errors on the position estimate. Due to the lack of a software license required for a control application on MR1, it was necessary to develop MR2.

In the experiments, for both MRs, it was verified that their position could not be estimated only using internal sensors. The use of external references is required. Also it was shown that knowing the MR's actual position was not enough. Better information on its orientation is also required, in order to reduce the trajectory error. The Sensor Fusion algorithm preformed fairly on both MRs trajectories, counting on the correction with the external measurement.

5 Variance adaptive LQR Controller

This chapter addresses the development and test of a LQR controller, using a Kalman Filter as state estimator. The speed with which the controller tracks the set-point changes (faster or slower) depending on the state's estimate degree of confidence (noise variance).

The problem rational is similar to a driver that has difficulty to see the road due to rain, fog, or some other difficulty of predicting the near future – he tends to drive at a slower speed. If he has a better insight into the future he can drive at a faster speed.

5.1 System Dynamics

Assuming that the system can be represented by:

$$X_{k+1} = A_p\, X_k + B_p\, U_k + W_k \tag{5.1}$$

$$Y_k = C_p\, X_k + D_p\, U_k + V_k \tag{5.2}$$

where X_k is the system's *n*-dimensional state, Y_k the system's *m*-dimensional output and U_k the system's *l*-dimensional input. The system dynamics is represented through matrices A_p, B_p, C_p and D_p. Noise acting on the state is represented by W_k and the output's or sensor noise by V_k.

The model used for state prediction is the same or close to the plant's dynamic, but with no noise parameter on the state equations,

$$\hat{X}_{\mathrm{k+1}} = A\, \hat{X}_k + B\, U_k \tag{5.3}$$

$$\hat{Y}_k = C\, \hat{X}_k + D\, U_k \tag{5.4}$$

this results on $X_k \approx \hat{X}_{\mathrm{k}}$, $Y_k \approx \hat{Y}_k$, both input, or actuation signals, are the same. The structure of the closed-loop system is presented in Figure 5.1.

5.2 LQR Controller

The controller is designed so it minimizes the Cost Function represented by:

$$J_{total} = J_{state} + J_{input} = \sum_{k=0}^{N} \hat{X}_k^T \, Q_{LQR} \, \hat{X}_k + \sum_{k=0}^{N-1} U_k^T \, R_{LQR} \, U_k \tag{5.5}$$

being Q_{LQR} and R_{LQR} the State's and Input's weight Matrices, respectively.

To minimize this cost function the state feedback control law (5.6) is used.

$$U_k = -K \, \hat{X}_k \tag{5.6}$$

This results on a controller that tries to take the state to 0, hence the designation of regulator. The controller gain K is computed through the iteration of expressions (5.7) - (5.12), starting from $i = N$ down to 0.

$$\Pi(N+1) = Q_{LQR} \tag{5.7}$$

$$S(i) = R_{LQR} + B^T \, \Pi(i+1) \, B \tag{5.8}$$

$$\Delta(i) = A^T \, \Pi(i+1) \, B \;\; S^{-1}(i) \, B^T \, \Pi(i+1) \, A \tag{5.9}$$

$$M_k(i) = A^T \, \Pi(i+1) \, A - \Delta(i) \tag{5.10}$$

$$K(i) = S^{-1}(i) \, B^T \, \Pi(i+1) \, A \tag{5.11}$$

$$\Pi(i) = Q_{LQR} + M_k(i) \tag{5.12}$$

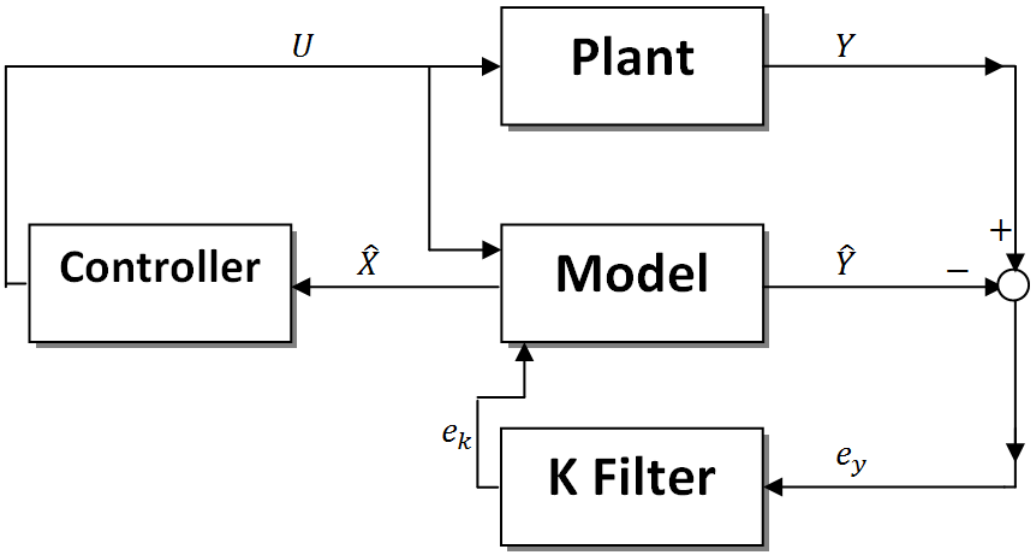

Figure 5.1 – Controlled System.

In order to allow the controller to adapt, either to the plant's dynamic changes or to the uncertainty state changes, a receding horizon policy is applied. At each moment only the first value of the computed $K(i)$ sequence is used. On the next time instant a new sequence is computed, either with a new dynamics or different control weighting matrices (if no change has occurred there is no need to perform the computations as the same controller gain sequence will result).

5.3 Adaptive Controller scheme

One of the goals is to develop an algorithm that adapts to the level of uncertainty of the state estimate (Figure 5.2). The aim is to slow down the reaction speed of the system, when the state variance increases. This is done by changing the weight of the input contribution to the cost function. The rational of this change is to increase the cost when the variance increases. Higher input costs, leads to a slower closed loop dynamics.

$$J_{input} = \sum_{k=0}^{N-1} U_k^T \left(\left(1 + v_f \, \sigma(k) \right) R_{LQR} \right) U_k \tag{5.13}$$

being v_f the variance factor.

This new cost function is obtained by replacing R_{LQR} by

$$R_\sigma(k) = \left(1 + v_f\ \sigma(k)\right)\ R_{LQR} \tag{5.14}$$

on the previous algorithm (5.7) - (5.12). This amounts to replace equation (5.8) by

$$S(i) = R_\sigma(k)\ + B^T\ \Pi(i+1)\ B \tag{5.15}$$

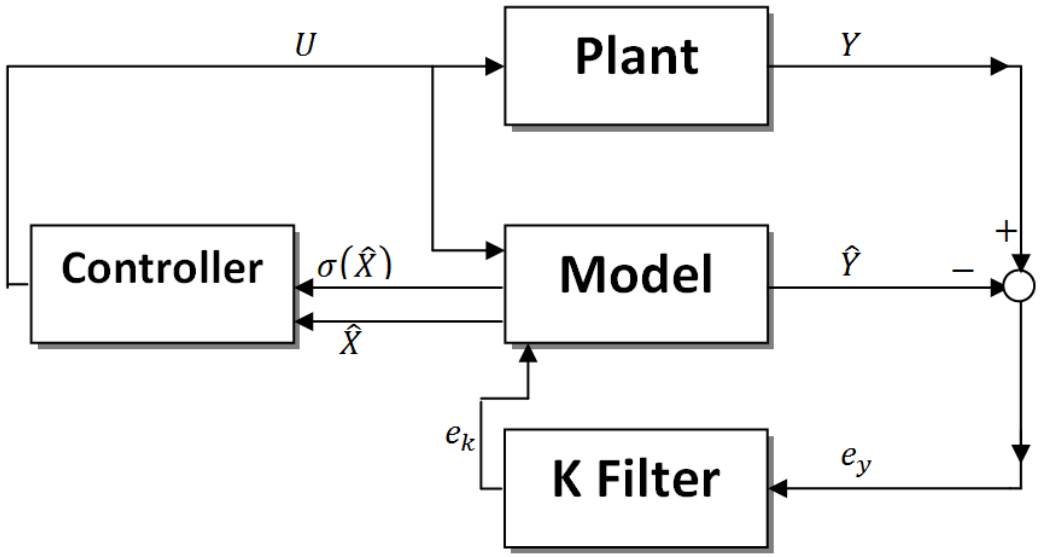

Figure 5.2 – Controlled System with State variance adapter

5.4 Kalman Filter

In most applications the process state is not directly accessible from sensor data. For that, it is required to use a state observer, in this case a Kalman Filter is used for the purpose. Assuming the plant dynamics (5.1) - (5.2), the state observer is implemented by the following set of equations:

- The predicted state error covariance matrix, quantifying the uncertainty of the model's state:

$$P_{k|k-1} = Q_k + A\ P_{k-1}\ A^T \tag{5.16}$$

where P_{k-1} is the previous state estimate error covariance matrix and Q_k is the state's noise covariance matrix. This noise covariance matrix is obtained through previously obtained experiments.

- State and plant output are predicted according to model (5.3) - (5.4).
- The model's state correction is made from output observations using equation (5.17),

$$\hat{X}_k = \hat{X}_k + K_k \left[Y_k - \hat{Y}_k\right] \tag{5.17}$$

where K_f is the gain filter calculated in the equation (5.18), being R_k the output or sensor noise covariance matrix.

$$K_k = P_{k|k-1}\, C^T \left[R_k + C\, P_{k|k-1}\, C^T\right]^{-1} \tag{5.18}$$

and it is also used to compute the updated covariance matrix of the filtered estimates

$$P_k = P_{k|k-1} - K_k\, C\, P_{k|k-1} \tag{5.19}$$

5.5 Reference tracking

One of the objectives for controlled systems is to make (at least) one of the outputs to follow a reference signal. As up to this moment only the regulator was addressed, which tries to take the state to 0, the controller needs to be altered so an output set point is tracked. There are different ways to modify the controller in order to accomplish this feature. This work considers two different approaches: through the use of integral effect or by adding a series gain compensator.

The use of an integral effect requires the addition of an extra state variable, X_n, used to compute the integral of the tracking error (Figure 5.3). As the state is expanded, the computations for the controller gain need to take the expanded dynamics into account, and an expanded controller gain K_d has to be computed.

Let A_d and B_d correspond to the expanded system's dynamics. Q_d and R_d are the expanded controller weight matrices.

$$A_d = \begin{bmatrix} A & 0 \\ -C & 1 \end{bmatrix} \qquad B_d = \begin{bmatrix} B \\ -D \end{bmatrix} \tag{5.20}$$

As the extra state does not change the number of controller signals the expanded matrix R_d is equal to the matrix R_{LQR}. The new state weighting matrix Q_d is larger than Q_{LQR} as the additional state variable X_n has to be taken into account.

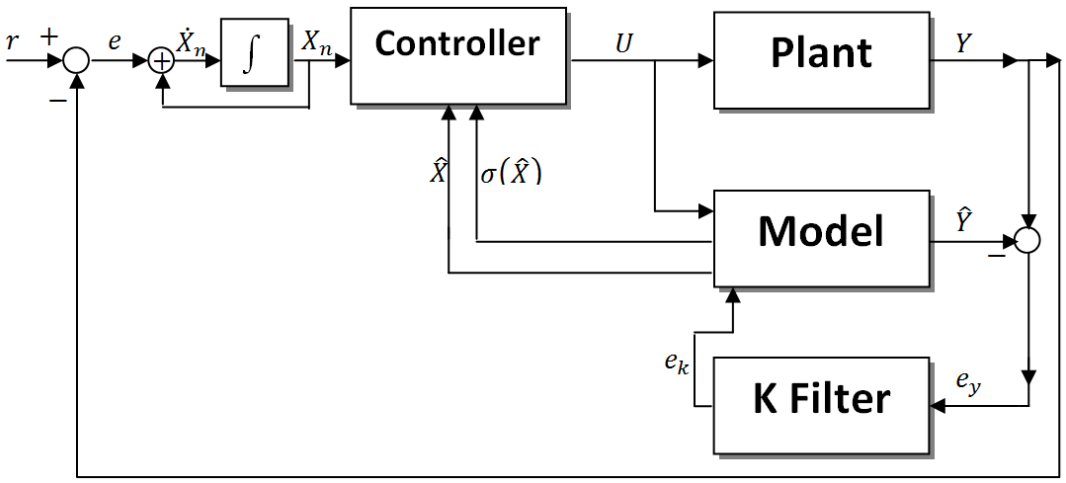

Figure 5.3 – Controlled System with integral effect, capable of following the reference signal.

Another alternative to track a set-point signal is the use of a series gain compensator. The controller is modified according to the structured proposed in Figure 5.4. The objective of the original controller is to take the state to 0, but as it is desired to make the output to track the desired reference, equation (5.6) is changed,

$$U_k = -K\, X_k + K_R\, r(k) \tag{5.21}$$

K_R is computed so that the static gain from set point r to output Y is unity.

This control structure is mainly used when the uncertainty on the close loop static gain results small, as is the case when the plant already has integral effect.

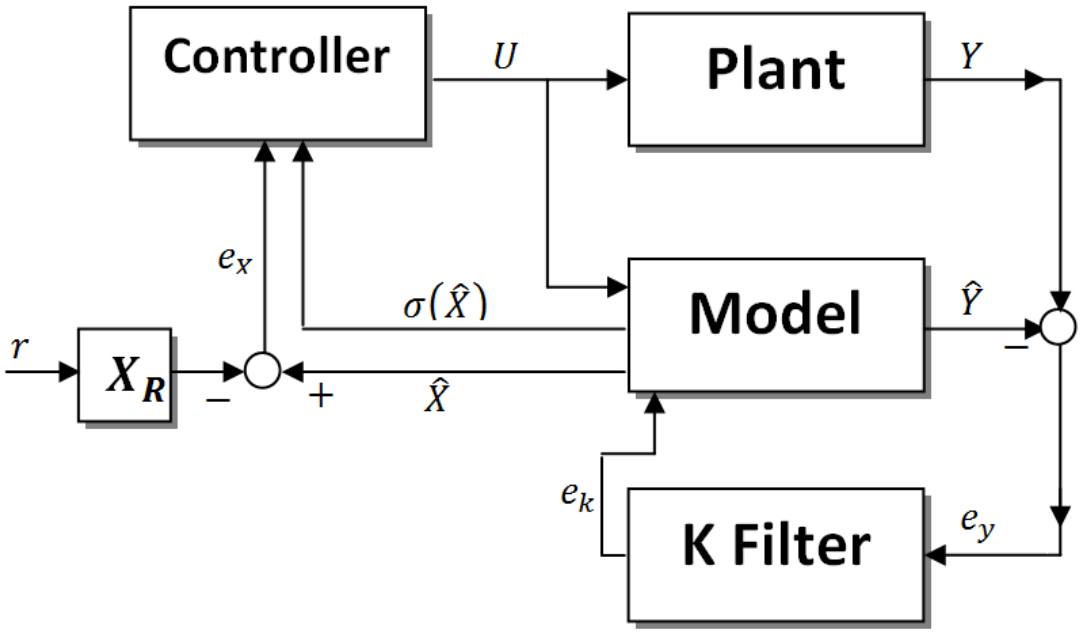

Figure 5.4 – Controlled System with a series gain, capable of following the reference signal.

5.6 Simulation Example 1

The system used in this example is described by state space matrices (5.22).

$$A = \begin{bmatrix} 1 & -0.8 \\ 0.8 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$
$$C = [1 \quad 0] \qquad D = [0] \tag{5.22}$$

The eigenvalues of this discrete time system are $0.5 \pm 0.6245i$, indicating the system is open-loop stable.

The dynamics (5.22) is used both for the plant and the observer model. On the noise free case with zero initial conditions, the state estimate will be identical to the state.

The reference signal is tracked using an integral based controller. Simulation results are presented in Figure 5.5 where a sample rate of $10\,samples/s$ is considered. The other controller parameters, in the described case, are

$$Q_{LQR} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad R_{LQR} = [1]$$

$$Q_k = \begin{bmatrix} 10^{-3} & 0 \\ 0 & 10^{-3} \end{bmatrix} \qquad R_k = [10^{-2}]$$

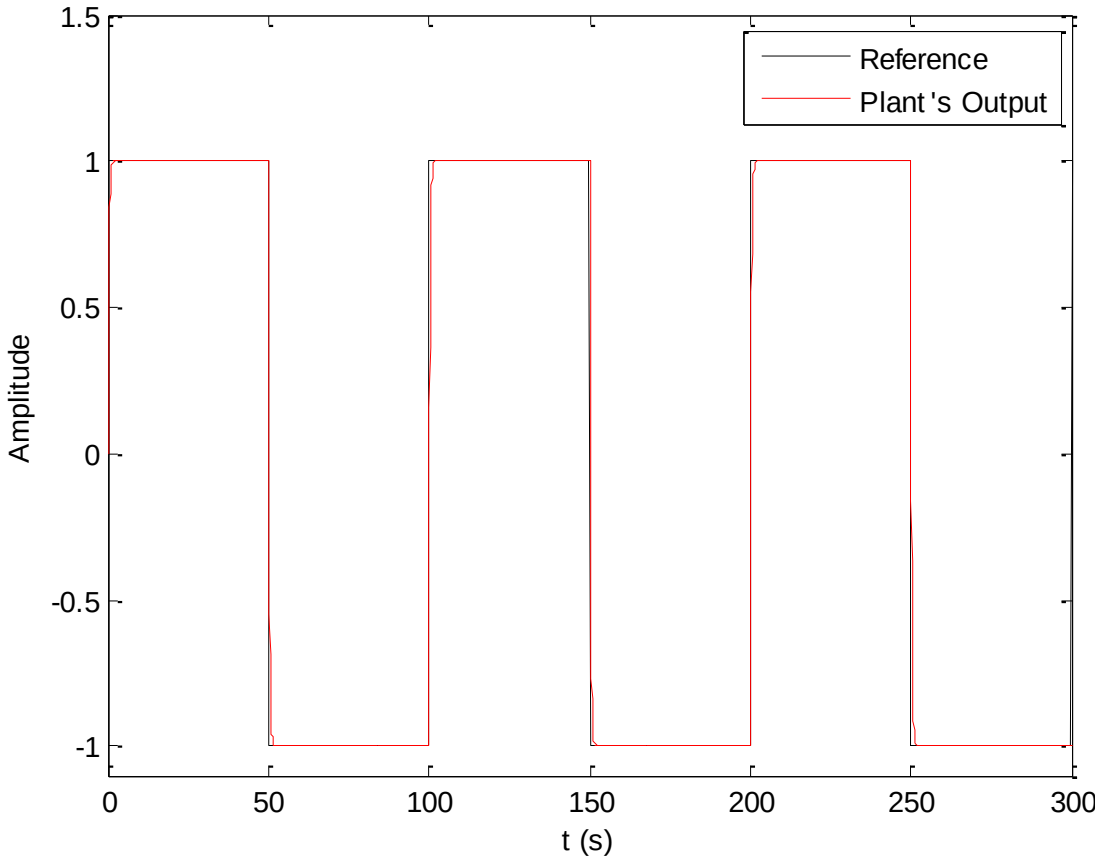

Figure 5.5 – Controlled System, following the reference signal.

As the plant dynamics is equal to the model dynamics, there is no need to correct, in other words, the Kalman Filter is used mainly to accommodate the uncertainty on the initial state value. As a result the variance of the estimated state converges to zero. Therefore adding the variance adapter to the controller makes no difference on the resulting output. Even with a small variance due to the output noise (noise power $5 * 10^{-2}$), and a high variance factor, v_f, (of 10 000), the result shows barely any difference (Figure 5.6 - Figure 5.8).

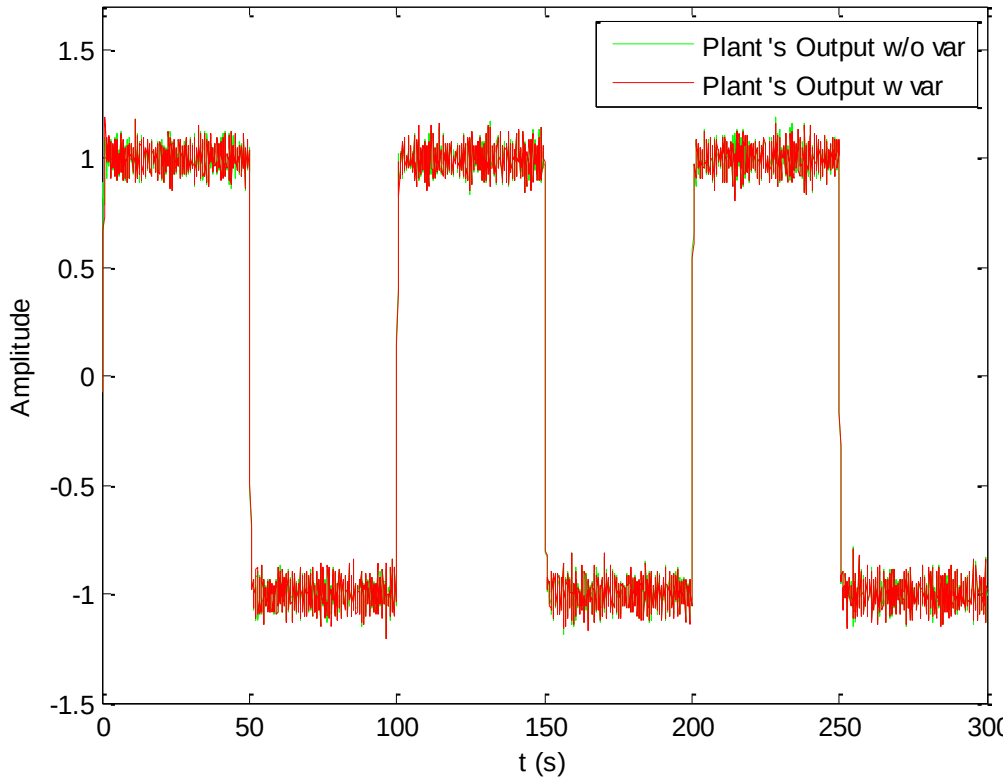

Figure 5.6 – Comparison between controlled systems, with and without variance adaptation.

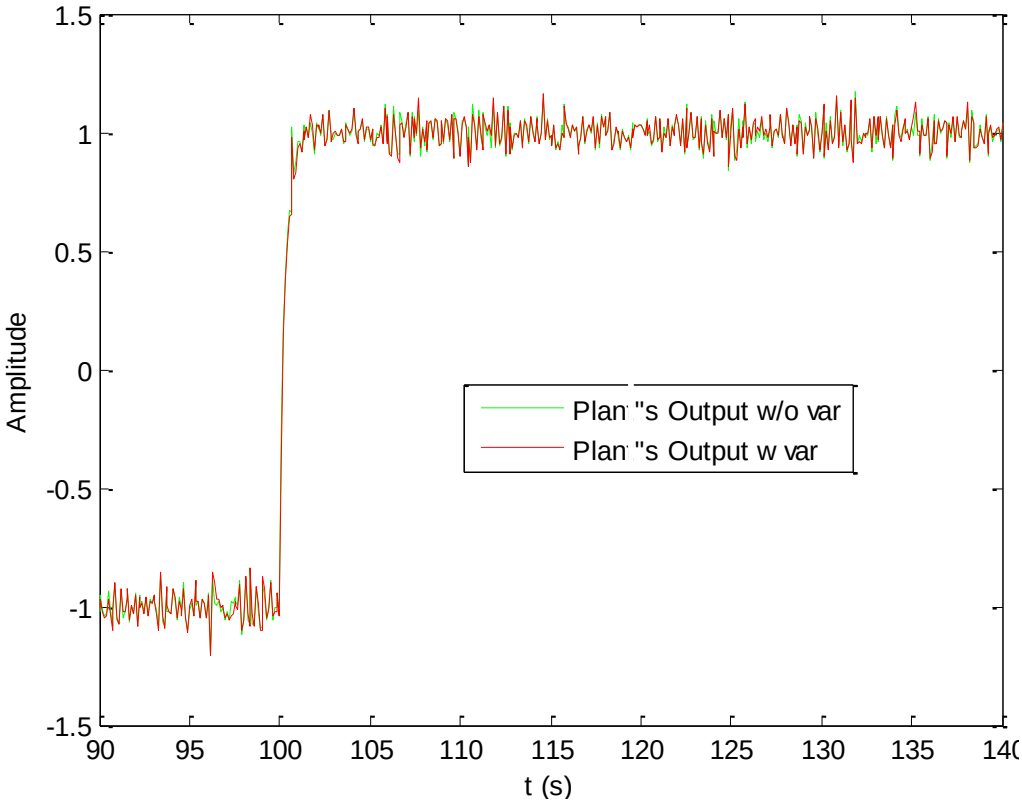

Figure 5.7 – Comparison between controlled systems,
with and without variance adaptation, on the interval 90s – 140s.

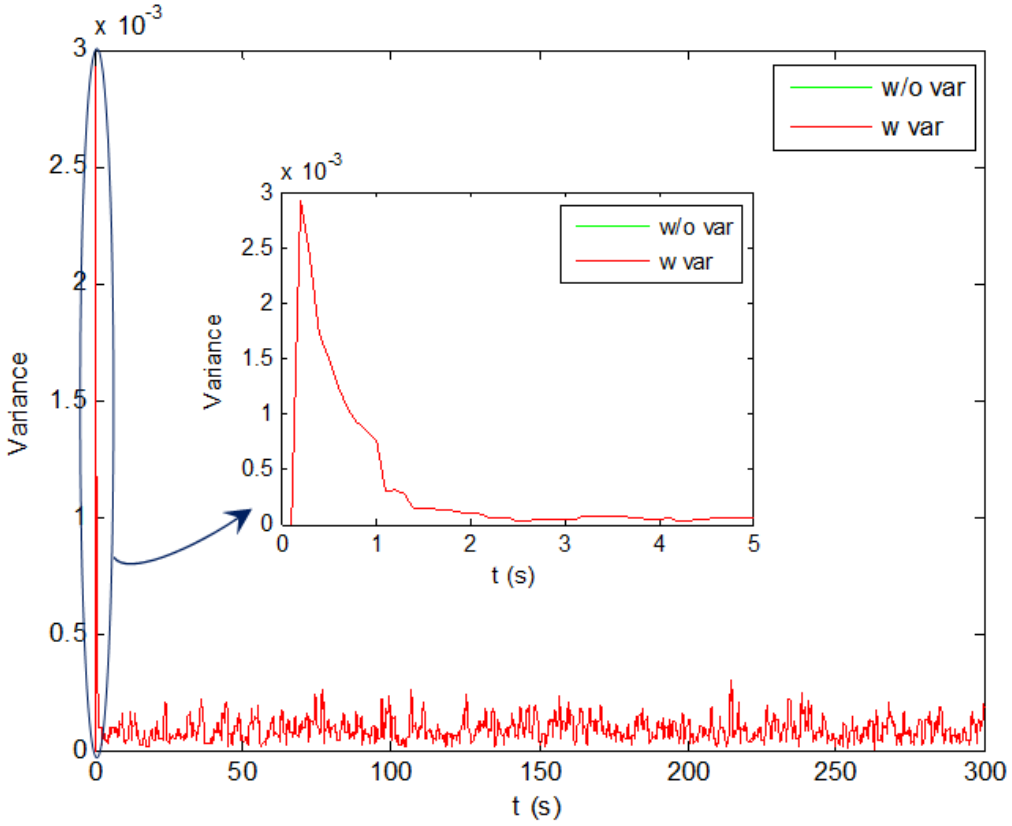

Figure 5.8 – Variance of the estimated state, with and without variance adaptation.

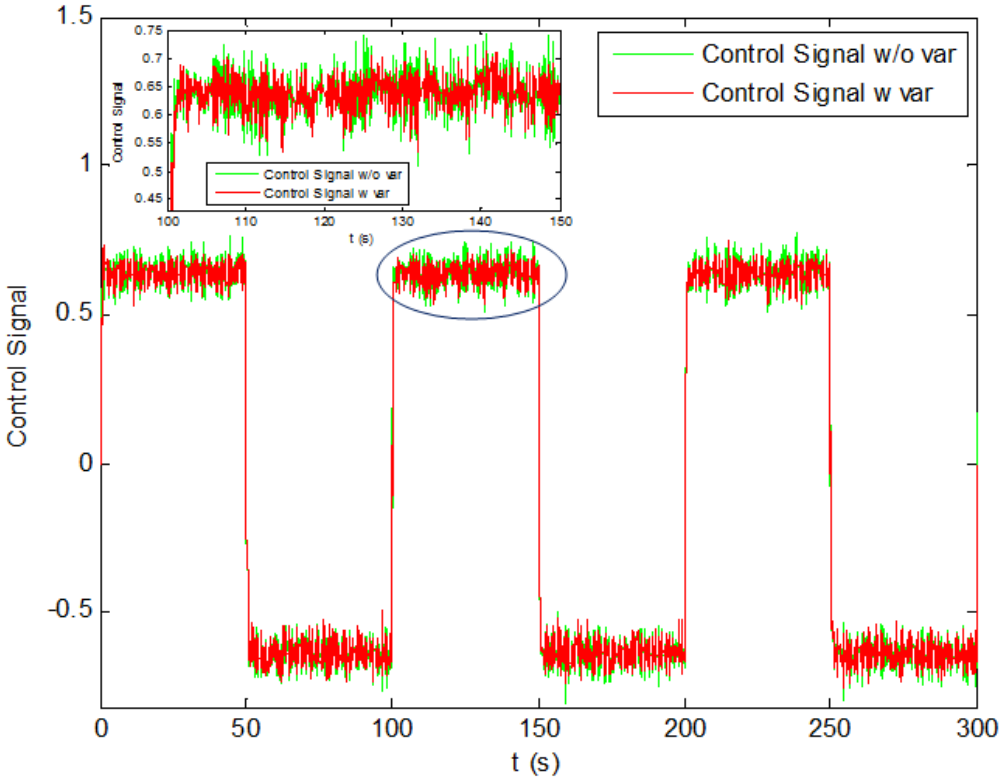

Figure 5.9 – Comparison between controlled signals, with and without variance adaptation.

Figure 5.9 presents the control signal associated with this experiment. Although it shows little difference at the output (Figure 5.7) and at the state variance (Figure 5.8), it presents a smaller oscillating amplitude over the control signal, meaning the possibility of less wear on the equipment.

5.7 Simulation Example 2

On the previous simulation the model for state estimation was the true plant dynamics, resulting on a stable closed-loop system and had making the variance of the estimated state close to 0. Hence it is hard to see the actual effect of the Variance Adapter on the controlled system.

The next step is to consider the use of a plant's dynamics different from the model. The plant state matrix (5.23) is critically stable, with Eigenvalues $0.5 \pm 0.8660i$.

$$A_p = \begin{bmatrix} 1 & -1 \\ 1 & 0 \end{bmatrix} \tag{5.23}$$

On this closed-loop system the reference tracking uses integral effect.

On this simulation (Figure 5.10 - Figure 5.14) the estimated state variance effect is visible on the edges of the reference step, when there is a large difference between the model's state and the plant's estimated state, unlike the previous simulation.

Through Figure 5.11, it is verified that this system has a establishment time of 4 seconds, larger than the previous plant, used on *Simulation Example 1*, which was 2 seconds. This difference is due to the Observer's degradation and the marginal instability.

Although the variance signal (Figure 5.12) is not entirely 0, it is still small, meaning, it still will make little difference when using the Variance Adapter ($v_f = 1000$).

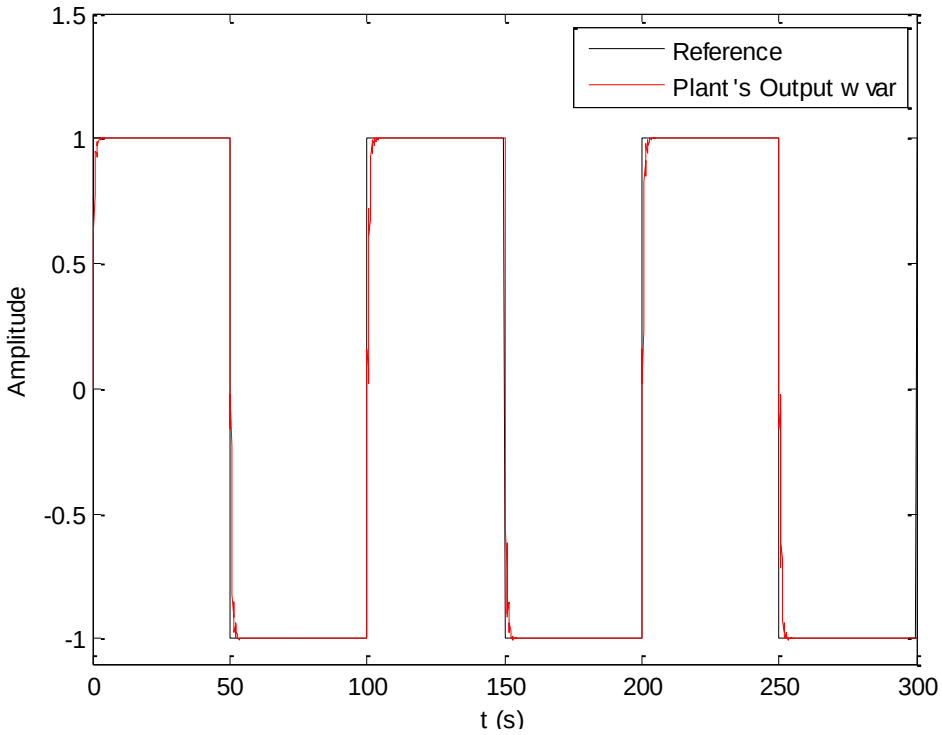

Figure 5.10 – Controlled System, following the reference signal.

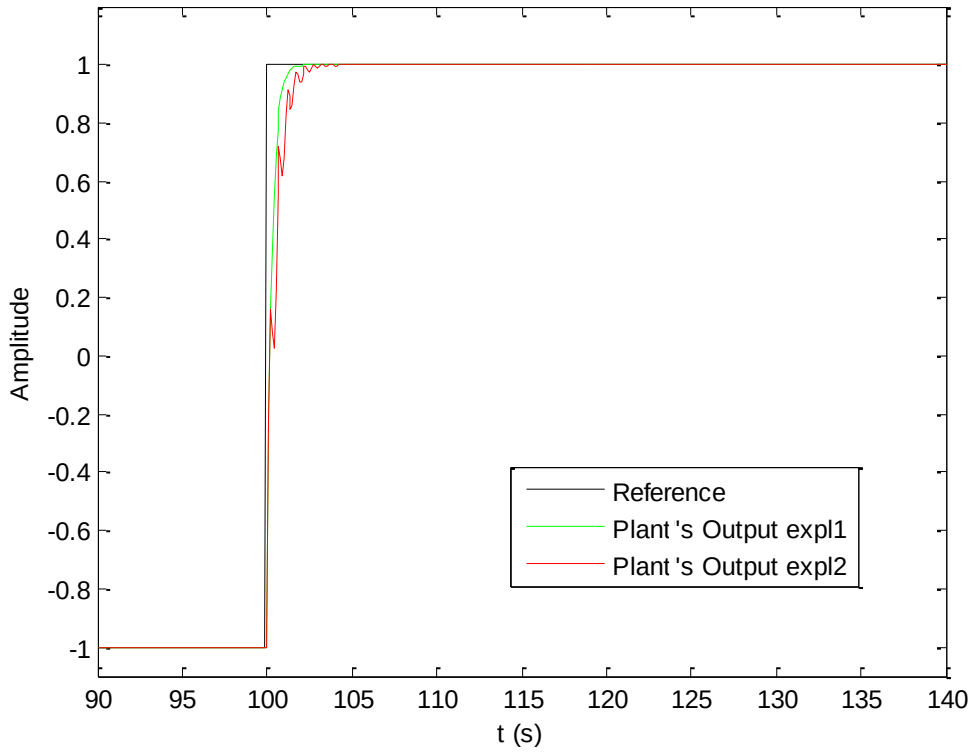

Figure 5.11 – Controlled System used on Simulation Example 1 and 2, following the reference signal, on the interval 90s – 140s.

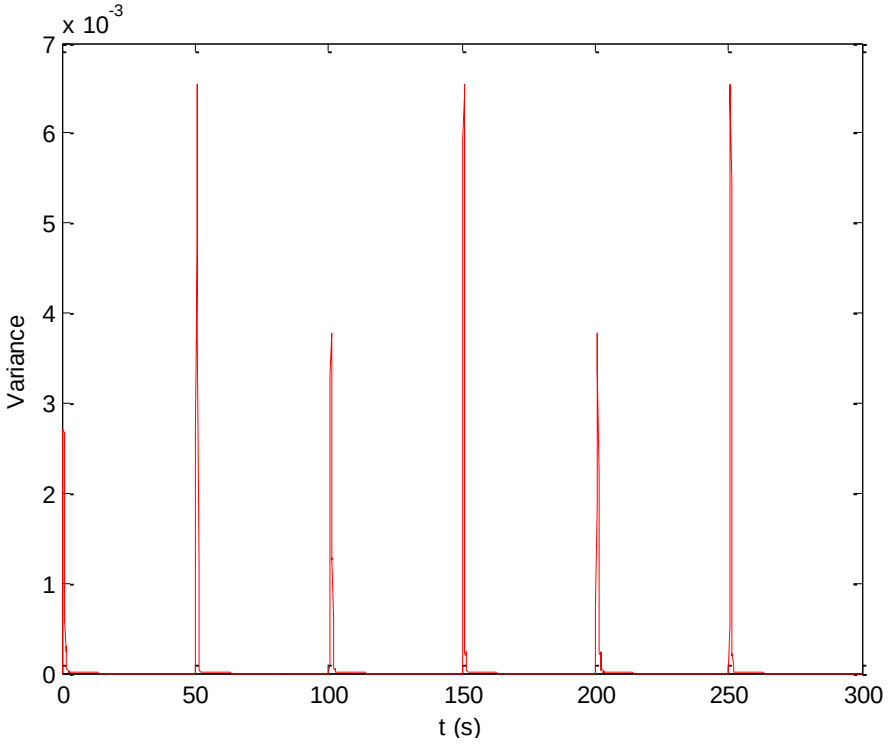

Figure 5.12 – Variance of the estimated state.

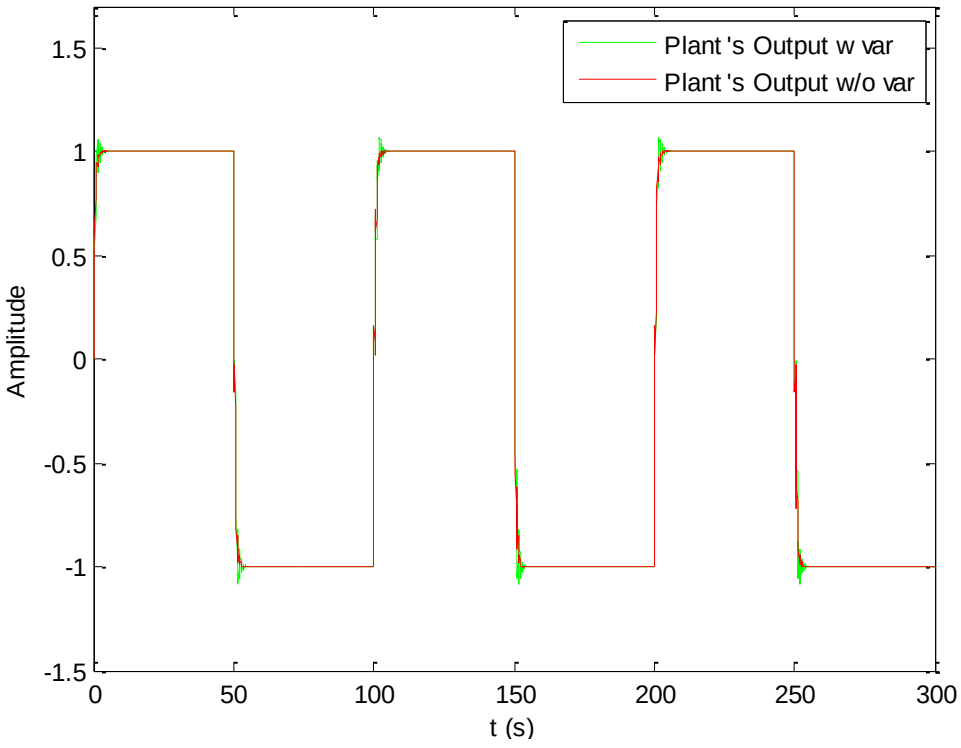

Figure 5.13 – Comparison between controlled systems, with and without variance adaptation.

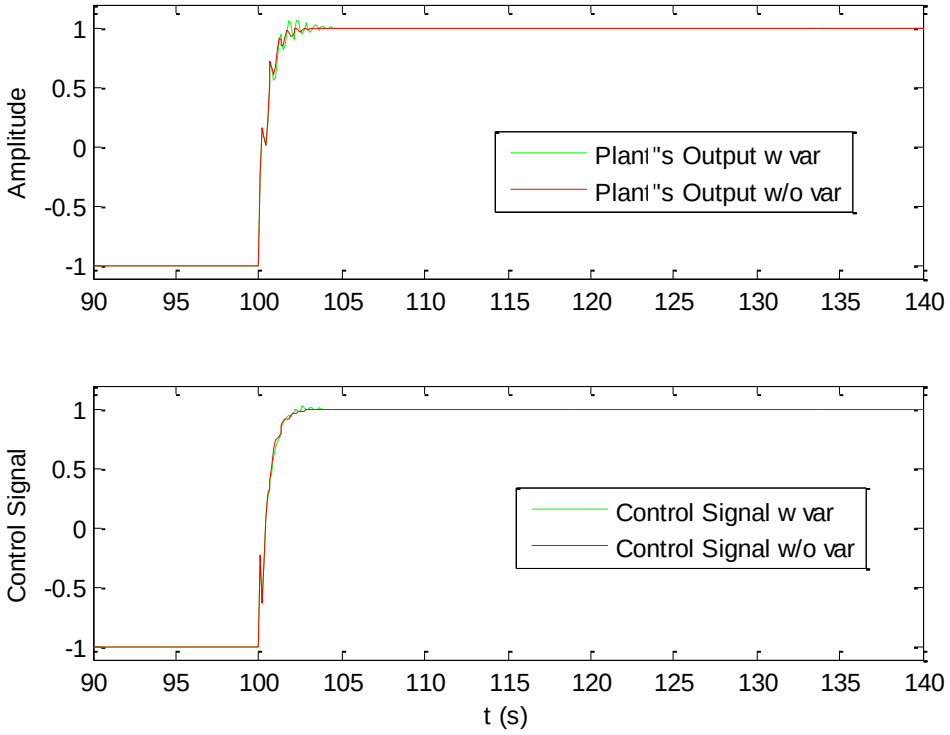

Figure 5.14 – Comparison between controlled systems, with and without variance adaptation, and respective control signal, on the interval 90s – 140s.

Adding noise to the output of the controlled system (noise power $5 * 10^{-2}$) makes the state variance higher, but not enough to produce a significant difference (Figure 5.15 - Figure 5.17).

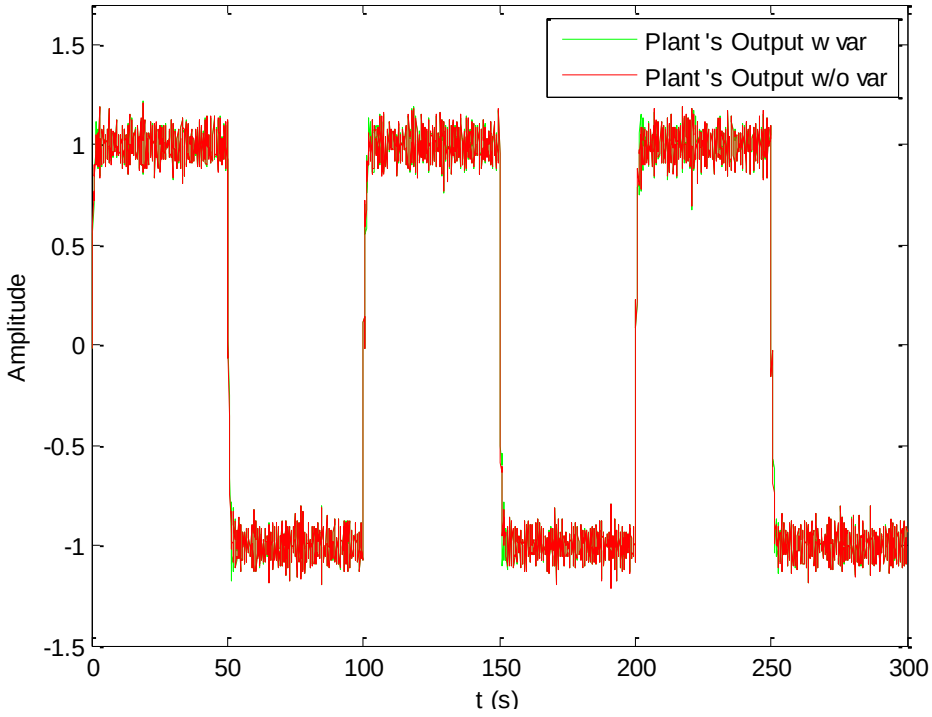

Figure 5.15 – Comparison between controlled systems, with and without variance adaptation.

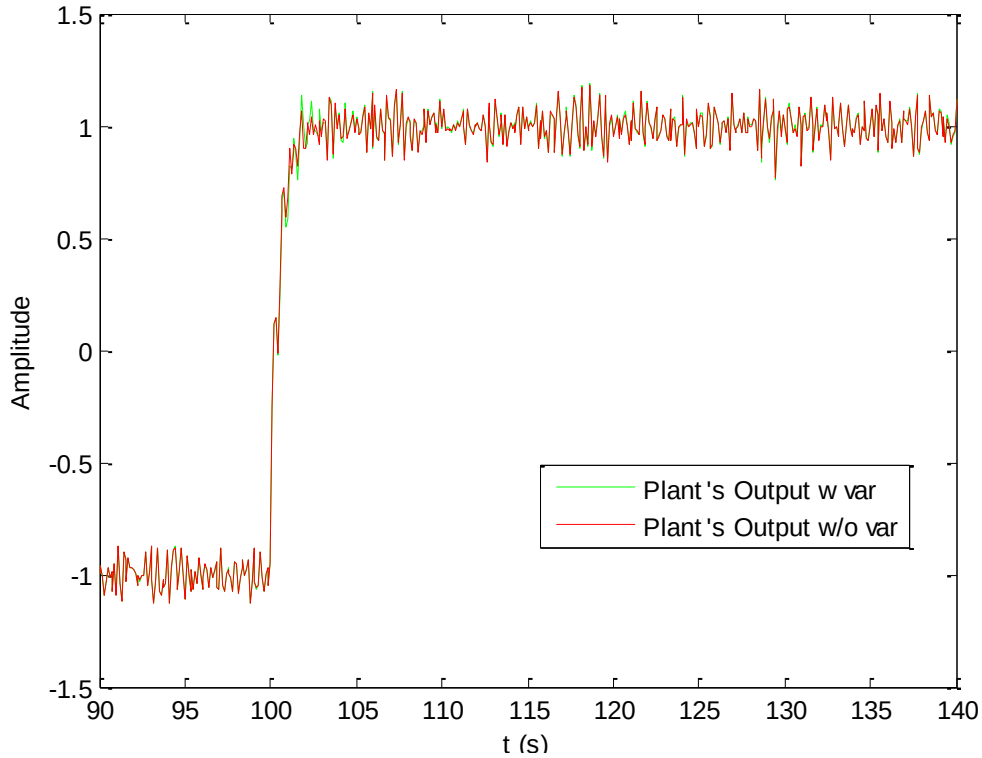

Figure 5.16 – Comparison between controlled systems,
with and without variance adaptation, on the interval 90s – 140s.

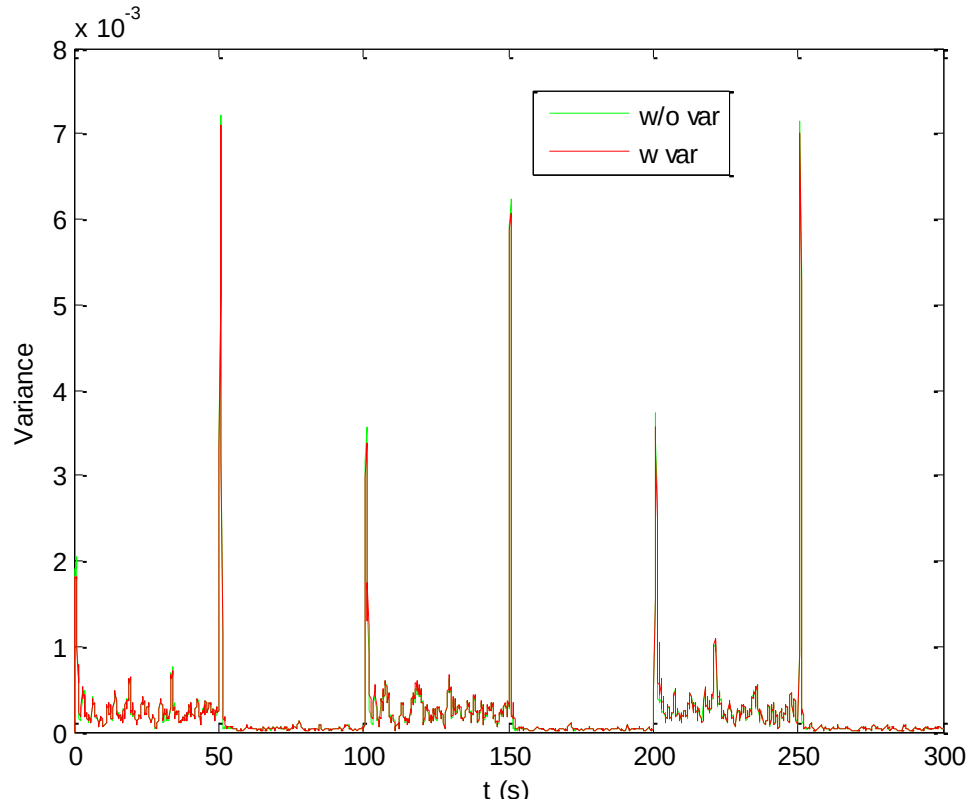

Figure 5.17 – Variance of the estimated state, with and without variance adaptation.

This experiment shows that the variance adapter doesn't always have advantages. In this situation the plant's dynamic is not the same as the model's, resulting on a disassociation between the plant's and model's states. As so, the adaptive variance causes an increase on the output's oscillation and overshoot (Figure 5.14 and Figure **5.16**), even though it slightly reduces the state variance (Figure 5.17).

5.8 Simulation Example 3 – Inverted Pendulum

5.8.1 Plant model

One of the models used to test this controller is the inverted pendulum. It is composed by a cart with a pendulum on top of it. The objective is to keep the pendulum at equilibrium by changing the cart's position. The control action is the pushing force applied to the cart. This system was chosen for being non-linear and open-loop unstable.

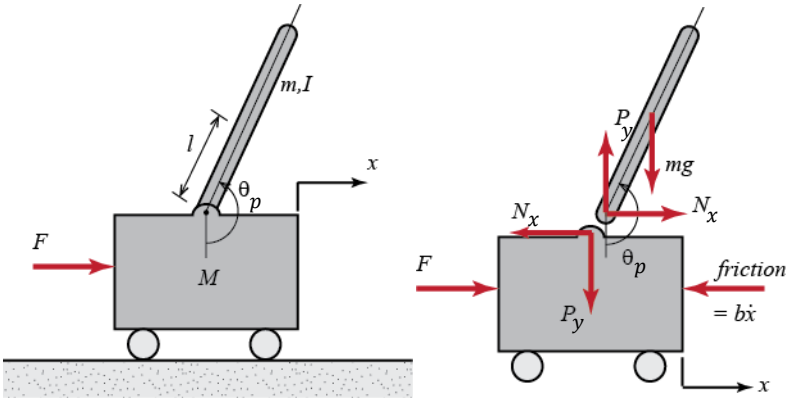

Figure 5.18 – Inverted Pendulum, basic scheme used.

The system may be described by the following set of 4 equations,

$$M\,\ddot{x} + b\,\dot{x} + N_x = F \tag{5.24}$$

$$N_x = m\,\ddot{x} + m\,l\,\ddot{\theta}_p\cos(\theta_p) - m\,l\,\dot{\theta}_p{}^2\sin(\theta_p) \tag{5.25}$$

$$P\sin(\theta_p) + N_x\cos(\theta_p) - m\,g\sin(\theta_p) = m\,l\,\ddot{\theta}_p + m\,\ddot{x}\cos(\theta_p) \tag{5.26}$$

$$-P\ l\sin(\theta_p) - N_x\ l\cos(\theta_p) = I\ \ddot{\theta}_p \tag{5.27}$$

The constants presented on equation (5.24) - (5.27) are:

$M$and m, the cart's and pendulum's Mass, respectively;

b, the friction coefficient of the cart;

N_x, P_y, the horizontal and vertical forces applied by the pendulum to the cart;

F, the horizontal force applied to the cart;

l and I, the pendulum's length and mass inertia, respectively;

g, gravity constant.

The variables used are, x, which is the cart's position, and θ_p, is the pendulum's angle. Combining all the equations (5.24) - (5.27) results on system (5.28),

$$\begin{cases} (M+m)\ddot{x} + b\,\dot{x} + m\,l\,\ddot{\theta}_p\cos(\theta_p) - m\,l\,\dot{\theta}_p{}^2\sin(\theta_p) = F \\ (I + m\,l^2)\,\ddot{\theta}_p + m\,g\,l\sin(\theta_p) = -m\,l\,\ddot{x}\cos(\theta_p) \end{cases} \tag{5.28}$$

The input of the system is the force F that is applied to the car. In the sequel it is used

$$\phi = \pi + \theta_p \tag{5.29}$$

5.8.2 Linearized state space model

The linearization of the continuous state space model of the system close to vertical position of the pendulum yields

$$\frac{\mathrm{d}}{dt}\begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \dfrac{-(I+m\,l^2)\,b}{I\,(M+m)+M\,m\,l^2} & \dfrac{m^2 g\,l^2}{I\,(M+m)+M\,m\,l^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \dfrac{-m\,l\,b}{I\,(M+m)+M\,m\,l^2} & \dfrac{m\,g\,l\,(M+m)}{I\,(M+m)+M\,m\,l^2} & 0 \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \\ + \begin{bmatrix} 0 \\ \dfrac{I+m\,l^2}{I\,(M+m)+M\,m\,l^2} \\ 0 \\ \dfrac{m\,l}{I\,(M+m)+M\,m\,l^2} \end{bmatrix} u \tag{5.30}$$

$$Y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u \tag{5.31}$$

As the controller used is discrete, this system has to be converted into a discrete time model. This is done through the zero order hold method with the Matlab's converter function, *c2d*.

5.8.3 Plant system

To simulate the system using a non-linear system model of the plant, equations (5.32) and (5.33) are used.

$$\ddot{x}(k+1) = \frac{u(k) - \left(b\,\dot{x}(k) + m\,l\,\ddot{\phi}(k)\cos(\phi(k) - \pi) - m\,l\,\dot{\phi}^2(k)\sin(\phi(k) - \pi)\right)}{M + m} \tag{5.32}$$

$$\ddot{\phi}(k+1) = \frac{-m\,g\,l\sin(\phi(k) - \pi) - m\,l\,\ddot{x}(k)\cos(\phi(k) - \pi)}{I + m\,l^2} \tag{5.33}$$

This set of equations is combined with Euler integration approach to obtain the cart's and pendulum's position.

5.8.4 Simulation Example 3 – Inverted Pendulum

For the simulation of the controlled inverted pendulum system, the following characteristics are considered:

The length of the pendulum, l, is 0.3m, the mass of the pendulum, m, and cart, M, are $0.2kg$ and $0.5kg$, respectively. The pendulum's mass moment of inertia, I, is $0.006\ kg\ m^2$ and the cart's friction coefficient, b, is 0.1.

The goal is to control the cart's position to follow the desired set point while "balancing" the pendulum. Without noise, with a time sample 0.1 and a control horizon of 10 samples, the result is presented on the Figure 5.19.

The set point tracking mechanism used with of this simulated system is the series gain. The inverted pendulum's dynamics already has tow eigenvalues at 0, adding the integral effect to follow the reference would make the system hard to control, with no benefits to reference tracking.

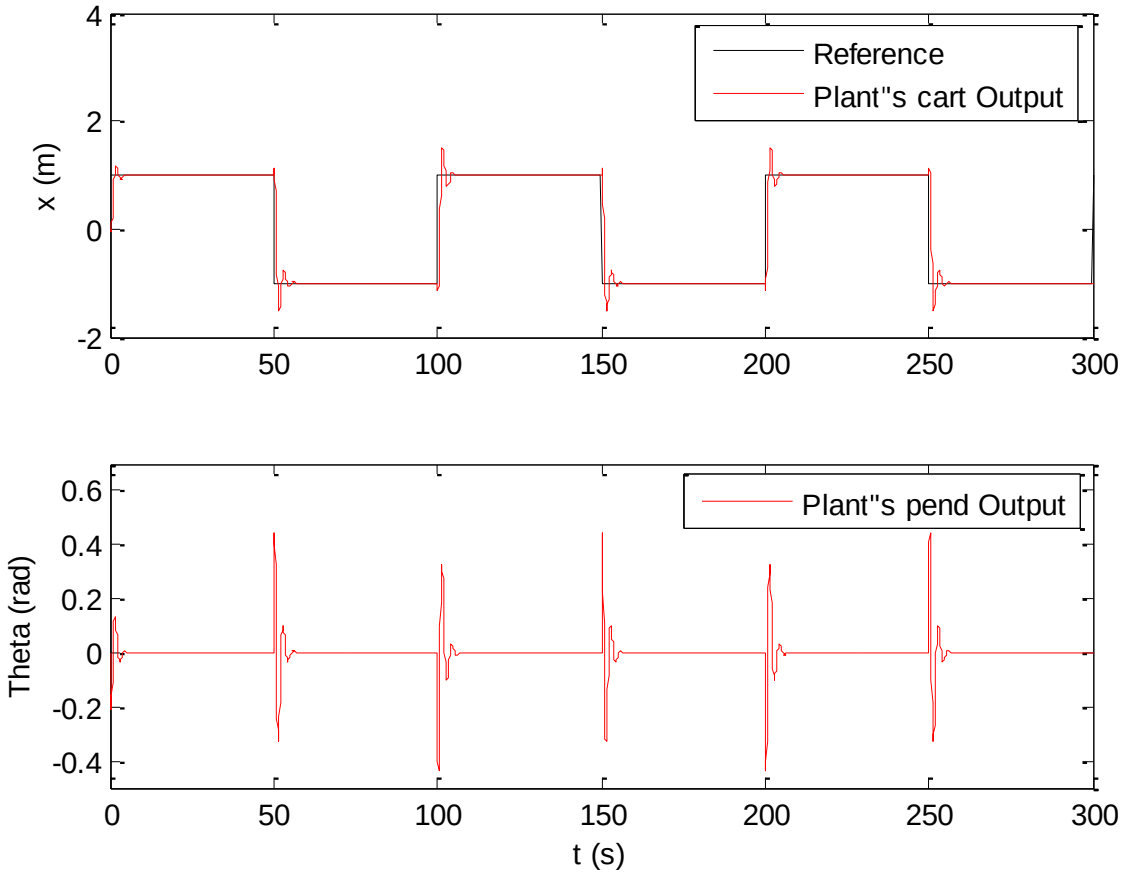

Figure 5.19 – Controlled System following the reference signal.

The variance of the model's state (Figure 5.20) is always close to zero except during the reference's step edges, when there is a strong difference between the model's state and the plant's estimated state.

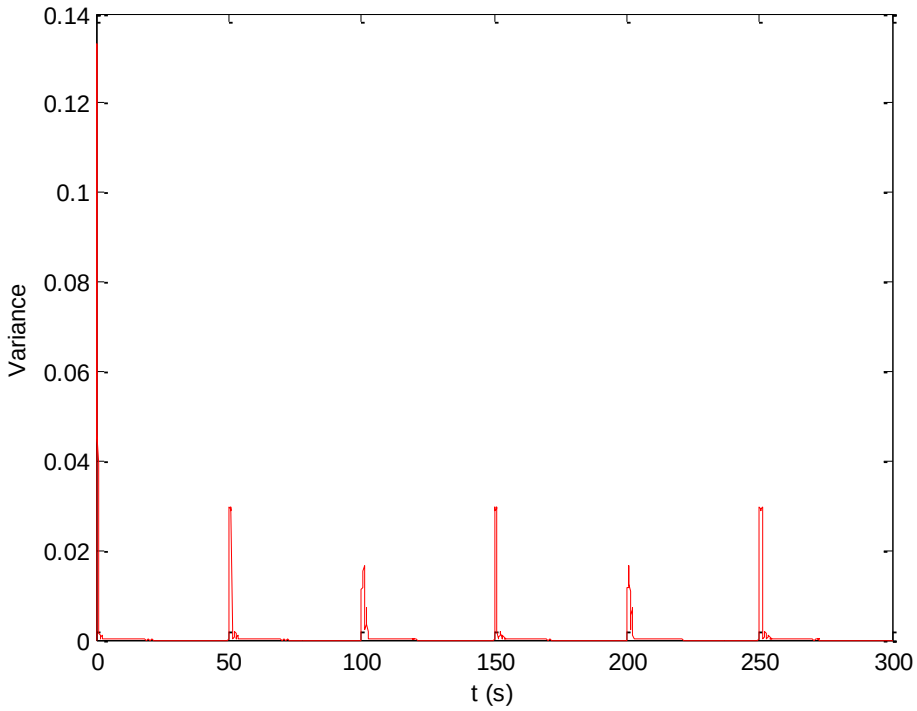

Figure 5.20 – Variance of the estimated state.

Using the variance signal to affect the controlled system will do little difference (Figure 5.21 - Figure 5.22), even when the variance factor is high, due to the signal being almost always close to zero (Variance factor used 100).

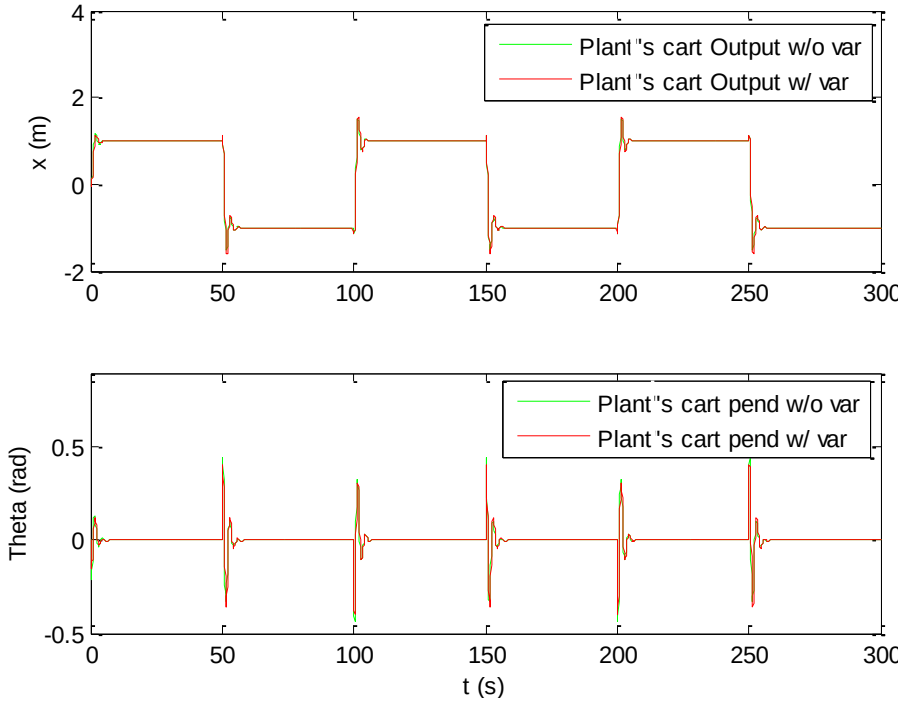

Figure 5.21 – Comparison between controlled systems, with and without variance adaptation.

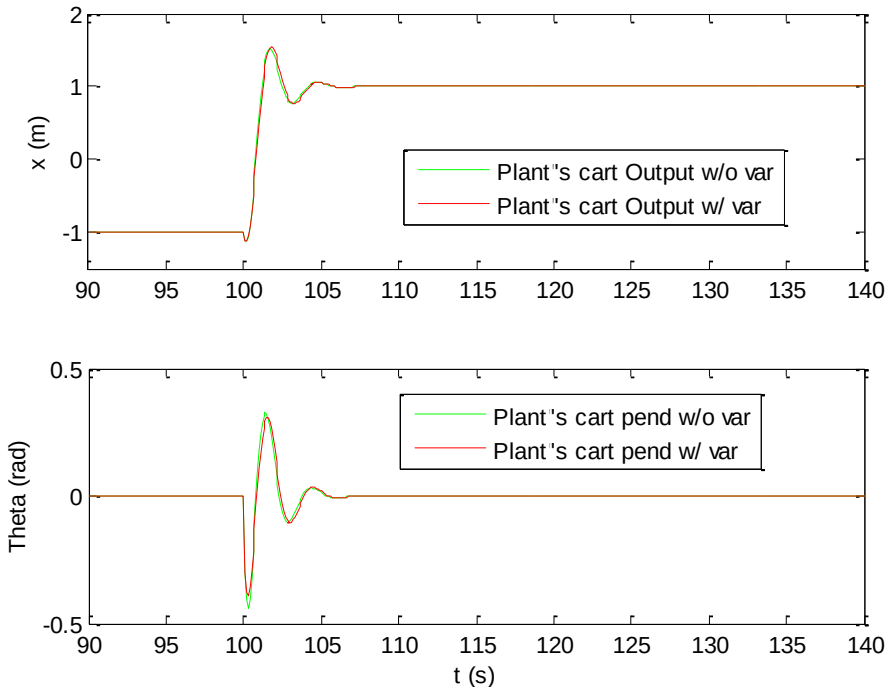

Figure 5.22 – Comparison between controlled systems,
with and without variance adaptation, on the interval 90s – 140s.

Adding a high noise to the output of the controlled system (noise power $5 * 10^{-2}$) makes the state variance signal to be higher, allowing a more visible effect of the variance on the controlled signal.

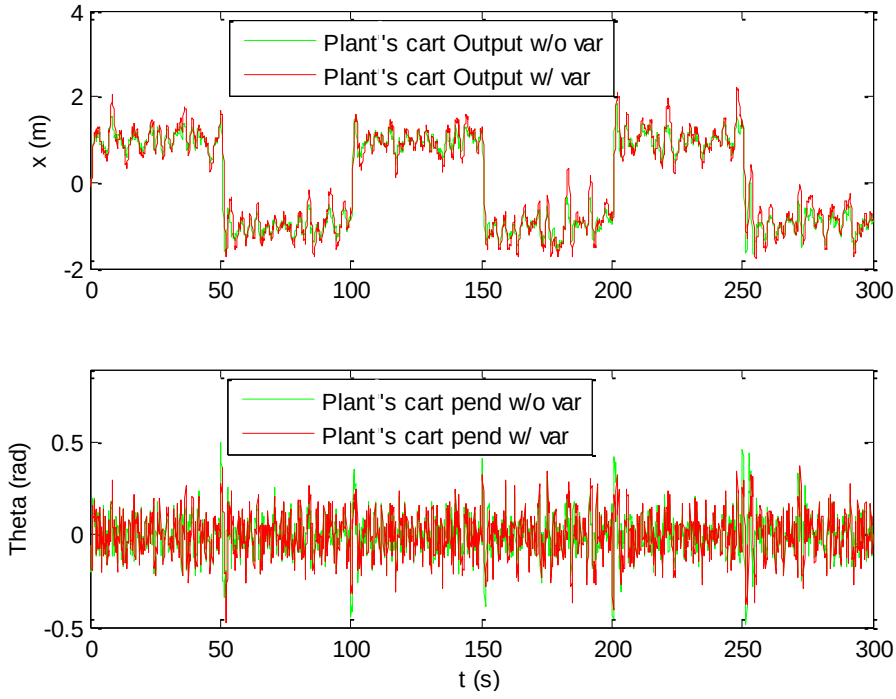

Figure 5.23 – Comparison between controlled systems, with and without variance adaptation.

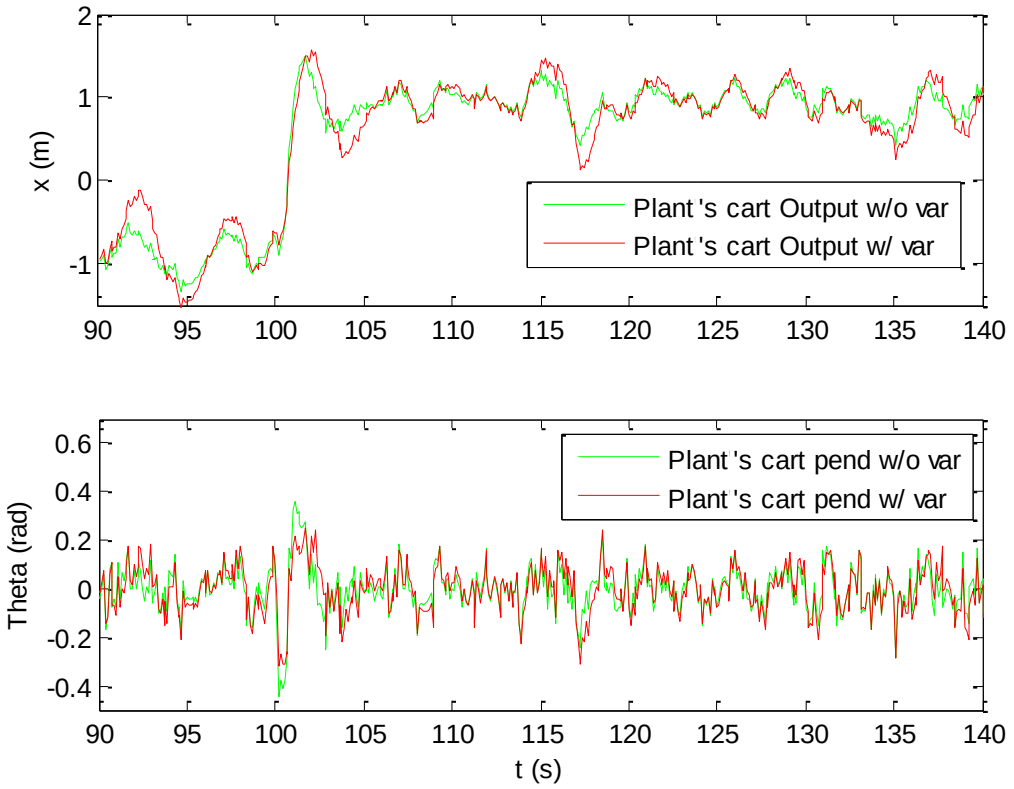

Figure 5.24 – Comparison between controlled systems,
with and without variance adaptation, on the interval 90s – 140s.

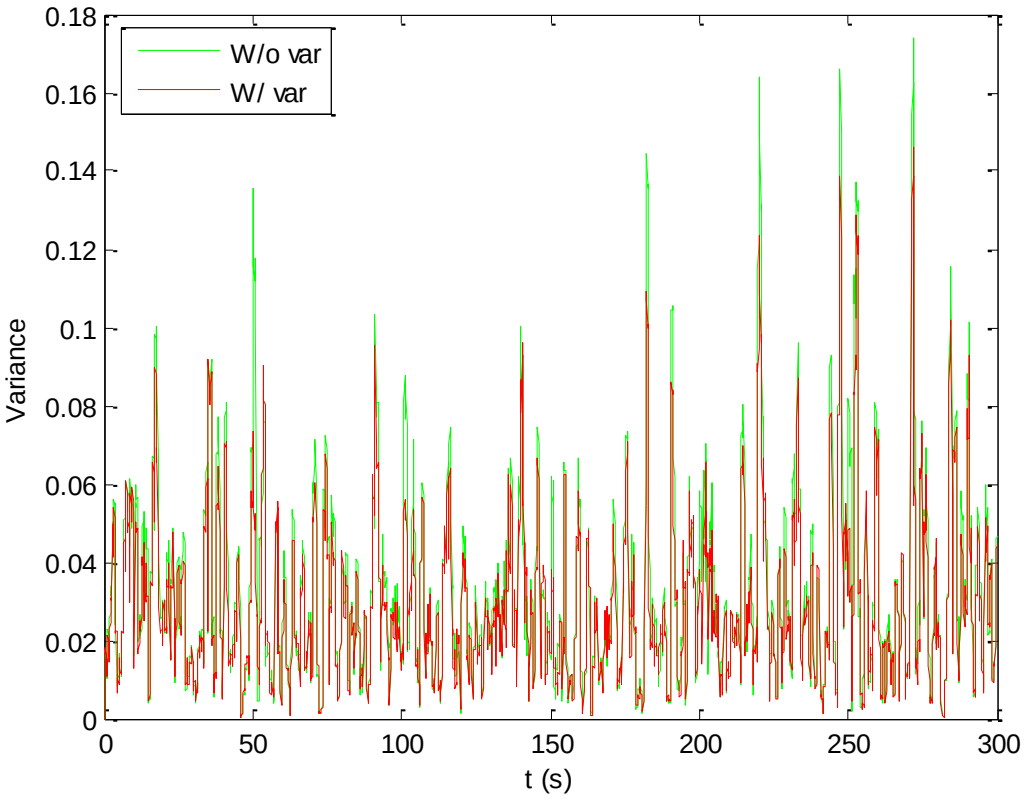

Figure 5.25 – Variance of the estimated state, with and without variance adaptation.

The Variance effect over the controlled system has a stronger presence when noise is added to the system. Although it is verified a slight over shoot on the cart's position, mostly due to the stabilization of the pendulum itself, it shows a decrease on the states variance, meaning a closer approximation between the model's and the estimated state.

The present controlled system, without noise, takes about $5s$ to stabilize (Figure 5.22), too fast to see the difference between the controlled systems, with and without the variance adapter. With a time sample $0.05s$ takes about 10s to stabilize (Figure 5.26).

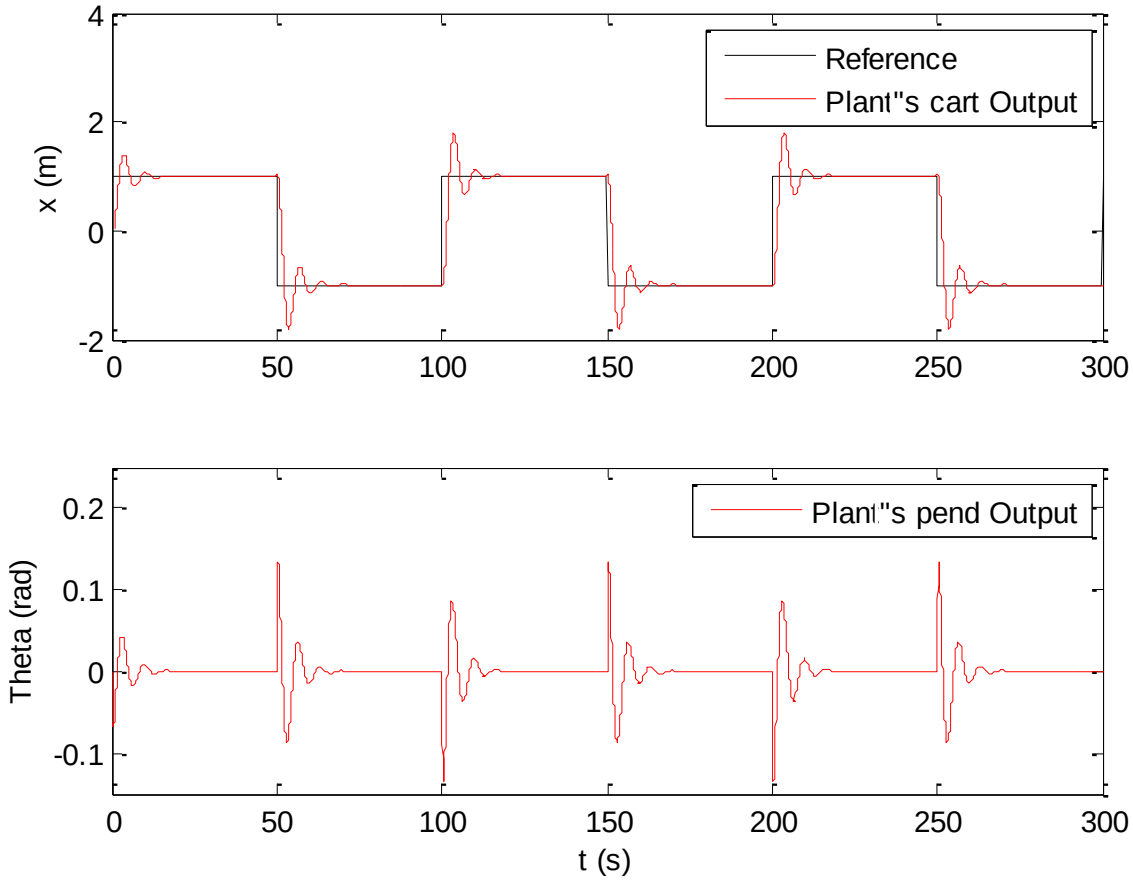

Figure 5.26 – Controlled System following the reference signal.

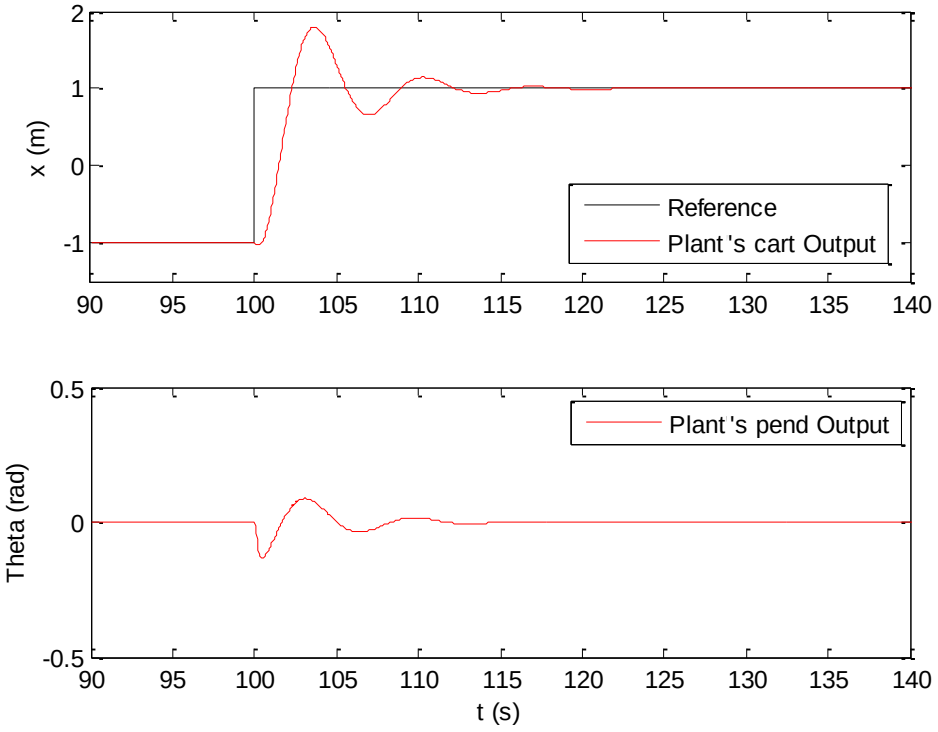

Figure 5.27 – Controlled System following the reference signal, on the interval 90s – 140s.

To see the effect of the variance adapter a high noise is added to the output of the controlled system (noise power $4 * 10^{-2}$) and different variance factors to controlled system.

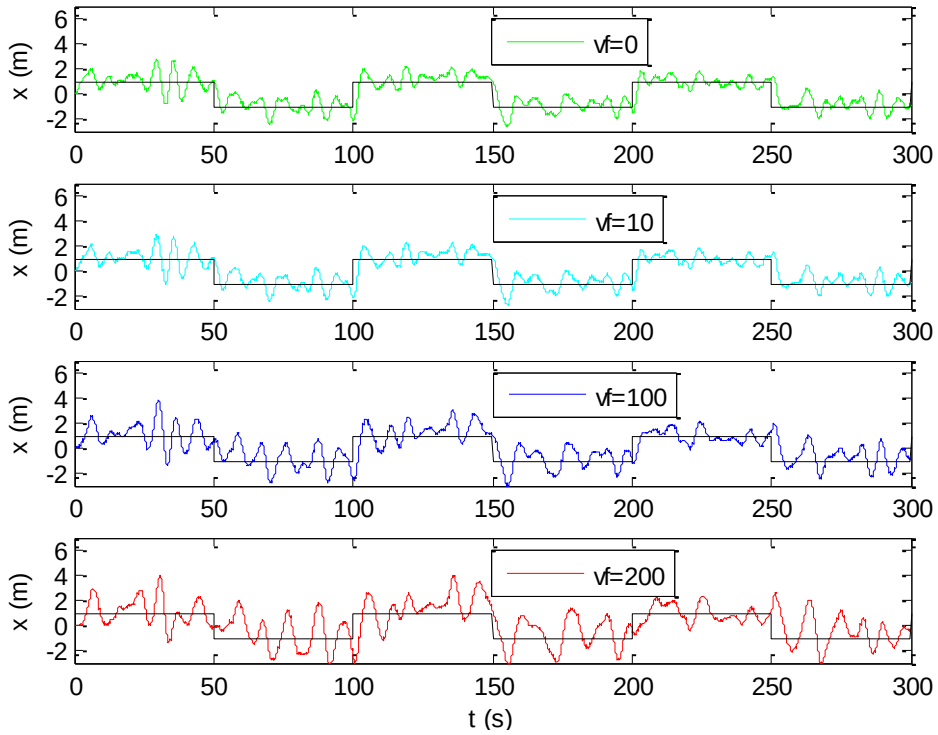

Figure 5.28 – Controlled systems (Cart's position), with different variance factors.

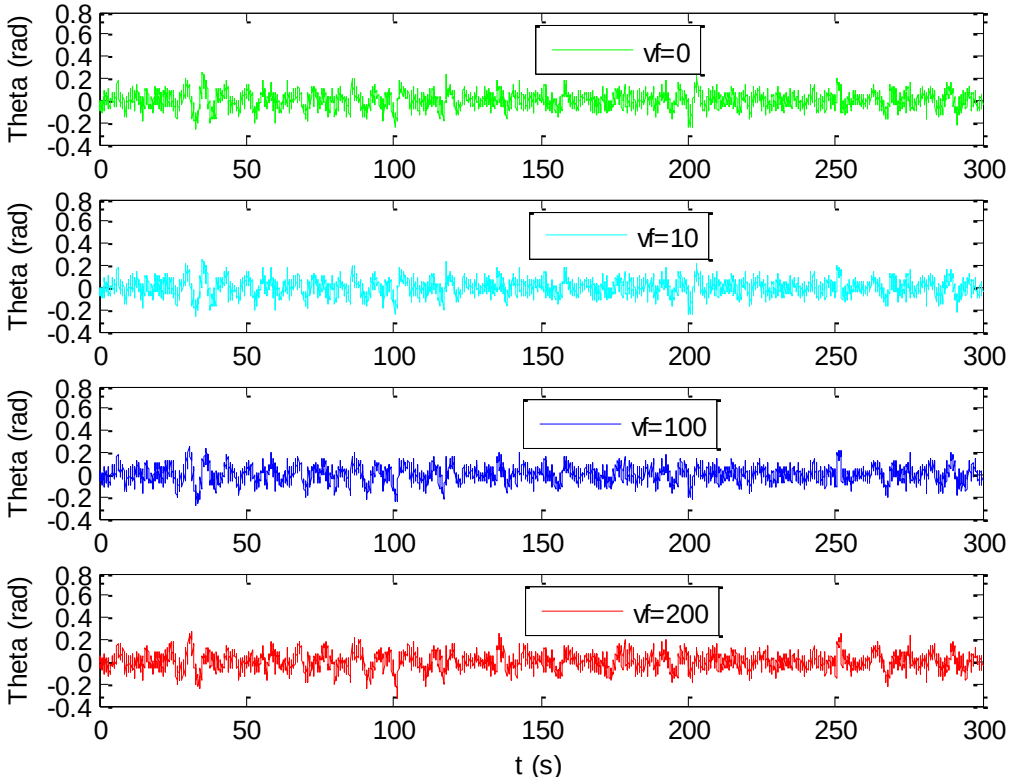

Figure 5.29 – Controlled systems (Pendulum's angle), with different variance factors.

As the stability time is longer than the previous set of data, the difference of the outputs is more visible (Figure 5.31). It is easily verified that the controlled system takes longer to react with a high variance factor, even though the cart's oscillation increase. As the system is open-loop unstable, increasing the control action weight makes the controller to act "slower", producing larger oscillations, and the consequent effect of the cart. Although the variance signal from the various controlled systems shows little difference (Figure 5.32), the control signal reduces, when the variance factor increased.

Computing the mean square value for each of the control signals from Figure 5.33, results in Table 5.1, which represents the Association between the Variance Factor and the Mean Square value of the Control signal.

Table 5.1 –Variance Factor and Control signal.

Variance Factor (vf)	Mean Square value of the Control Signal
0	0.9015
10	0.8902
100	0.8307
200	0.7968

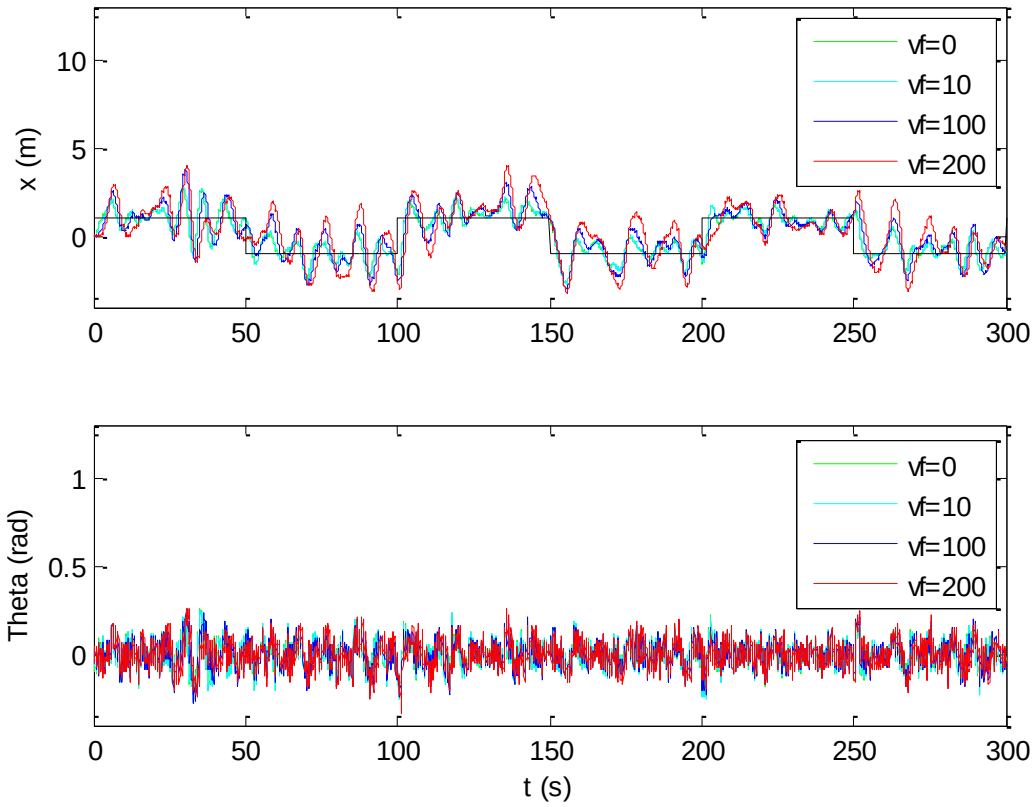

Figure 5.30 – Comparison between controlled systems, with different variance factors.

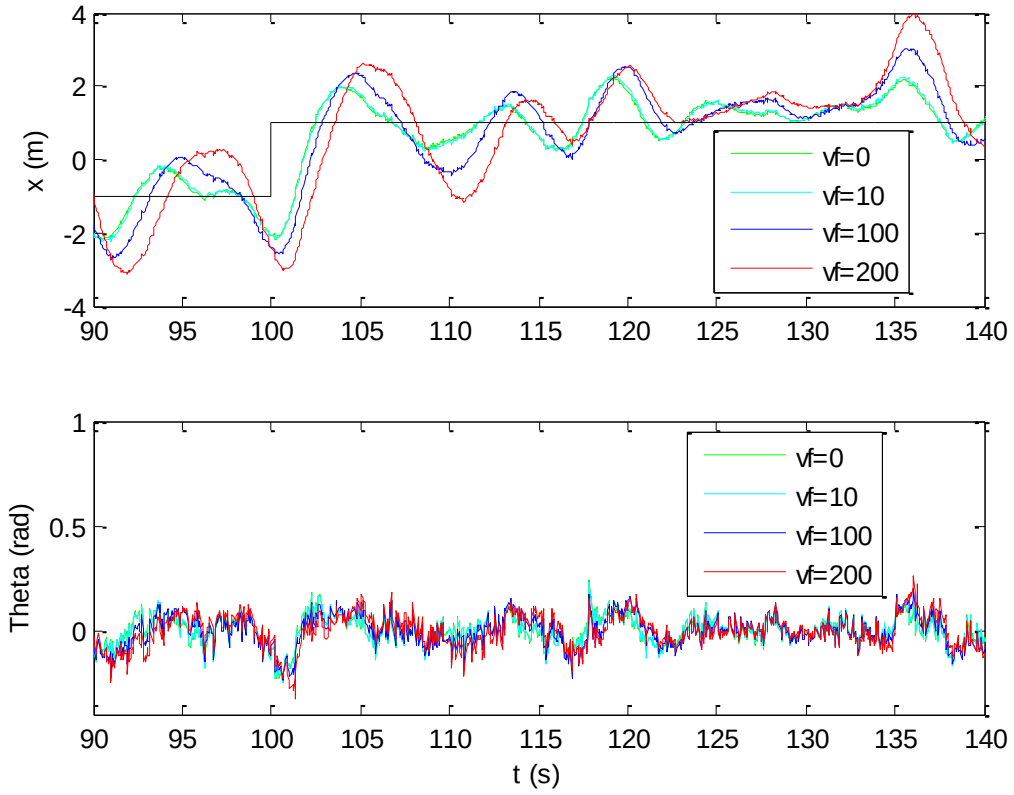

Figure 5.31 – Comparison between controlled systems,
with different variance factors, on the interval 90s – 140s.

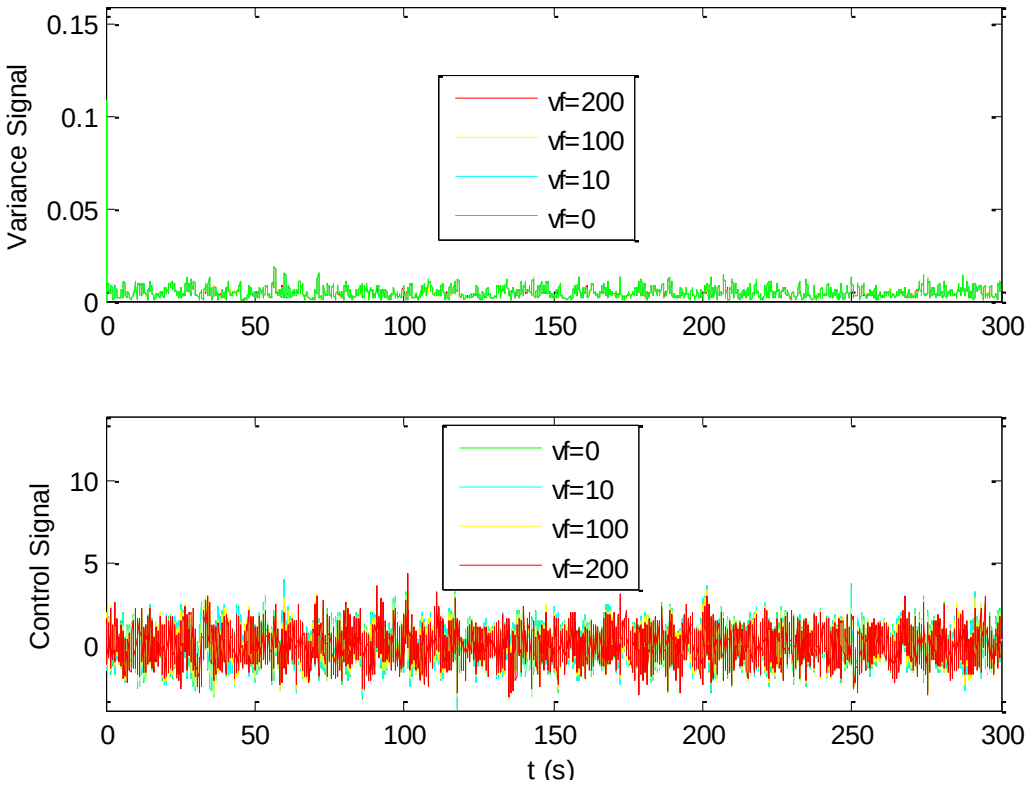

Figure 5.32 – Comparison between the controlled system's Variance and Control signals, with different variance factors.

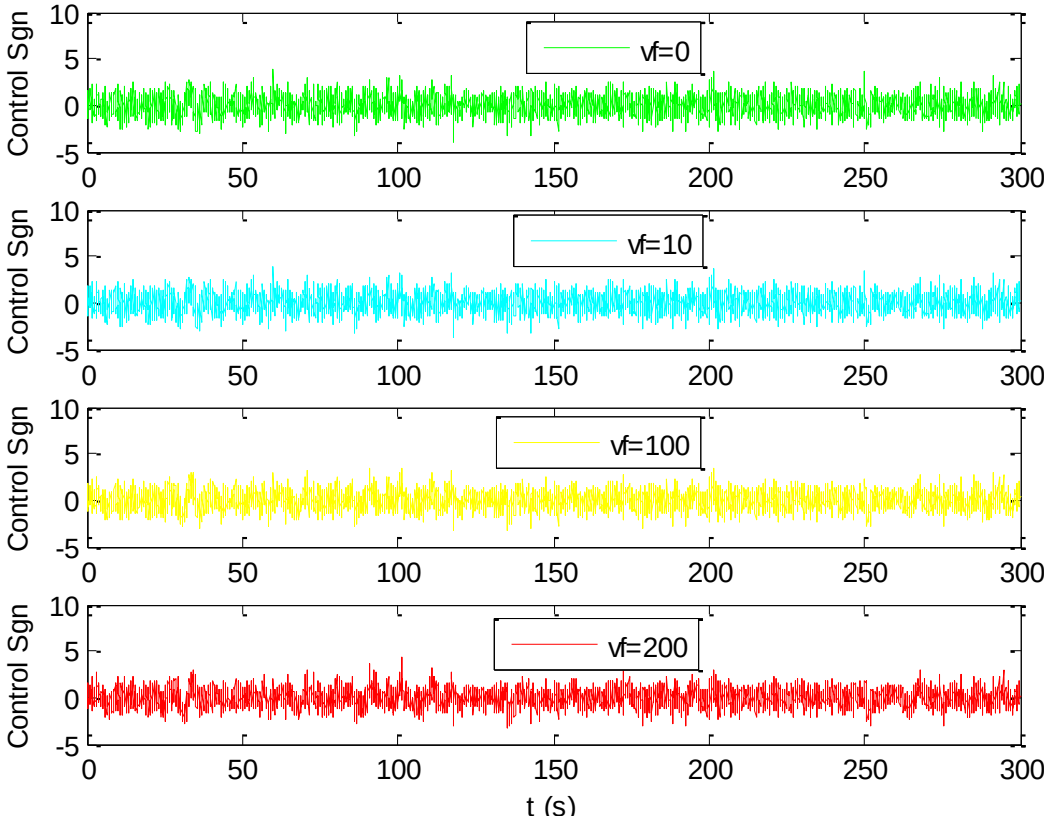

Figure 5.33 – Controlled System's Control Signal, with different variance factors.

5.9 Application – Control of a Mobile Robot's position

A LQR controller was used to control the position of a Mobile Robot (MR). The time evolution of the position set point is used to make the MR to follow a predefined trajectory.

The used MR is a 2 wheeled robot built around the Lego NXT Mindstorm, coupled with an IMU sensor mounted on the top (MR2).

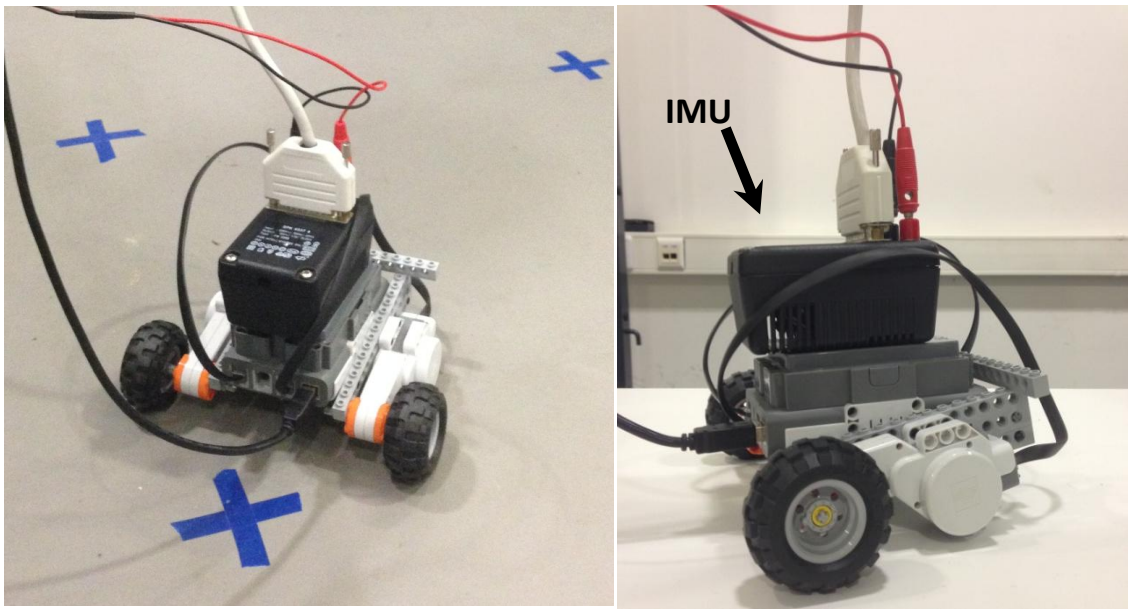

Figure 5.34 – Mobile Robot used Lego NXT Mindstorm, 2 wheeled robot, with an IMU sensor mounted.

The MR's control inputs are w, the forward speed control, and dw, the rotation speed control. The rotation is controlled through the difference between the robot's right and left wheels (dw).

The MR's actual forward and rotation speed (v_T and v_R) are obtained from the MR's odometer and the IMU's gyro sensor.

To estimate the MR's position and orientation it is used a Kalman Filter (KF). The KF functions is a sensor fusion module. No external sensors, that could provide the MR's actual position, were available during the experiments.

Due to the low friction between the floor and the wheels, the MR's wheels may slip which can cause the Kalman Filter to estimate the MR's position with larger error. The experiments that possessed excessive error were discarded. These results tended to be due to the inconsistency between the data provided by the gyroscope and the odometer.

5.9.1 MR's Dynamics

The system model considers the following variables: forward speed (v_T), rotation speed (v_R), Orientation (R) and position (x , y). The non-linearized dynamics used for the MR at each moment is:

$$\frac{d}{dt}x(t) = v_T(t)\ \cos\big(R(t)\big) \tag{5.34}$$

$$\frac{d}{dt}y(t) = v_T(t)\ \sin\big(R(t)\big) \tag{5.35}$$

The linearization and discretization of this model leads to the following set of equations

$$\begin{aligned} &\mathrm{x}(t_k) = x(t_{k-1}) + v_T(t_{k-1})\cos\big(R(t_{k-1})\big)\ (t_k - t_{k-1}) - \\ &-v_R(t_{k-1})\ v_T(t_{k-1})\sin\big(R(t_{k-1})\big)\ (t_k - t_{k-1})^2 \end{aligned} \tag{5.36}$$

$$\begin{aligned} &y(t_k) = y(t_{k-1}) + v_T(t_{k-1})\sin\big(R(t_{k-1})\big)\ (t_k - t_{k-1}) + \\ &+v_R(t_{k-1})\ v_T(t_{k-1})\cos\big(R(t_{k-1})\big)\ (t_k - t_{k-1})^2 \end{aligned} \tag{5.37}$$

The model also uses as additional state variable v_T, v_R and R. The resulting system's state matrices, between two consecutive state samples ($\Delta \mathrm{t_k} = \mathrm{t_k} - \mathrm{t_{k-1}}$), are

$$A(\Delta t_k) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & \Delta t_k & 1 & 0 & 0 \\ \cos\big(R(t_{k-1})\big)\ \Delta t_k & -v_T(t_{k-1})\sin\big(R(t_{k-1})\big)\ {\Delta t_k}^2 & 0 & 1 & 0 \\ \sin\big(R(t_{k-1})\big)\ \Delta t_k & v_T(t_{k-1})\cos\big(R(t_{k-1})\big)\ {\Delta t_k}^2 & 0 & 0 & 1 \end{bmatrix} \tag{5.38}$$

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{5.39}$$

5.9.2 MR's Trajectories

This work is intended to evaluate the MR's LQR controller, and to compare the performance achieved with and without state variance weighting. Two types of trajectories were used:

- Position set point steps, where there is a sequence of discrete "objectives" that the MR has to reach, by a specific order. The MR's "objective" changes when it gets close to the current goal. In Figure 5.35, each objective has a red or green circle around it to delimit the objective's area. Once the MR reaches that area, the current objective is changed to the next one. This results on the MR's expected trajectory being slightly different from the trajectory directly resulting from the set of "objective" points.
- Continuously time-varying set point, where the MR's "objective" varies continuously with time. This may be seen as a reproduction of the "car chase" idea. The trajectory set point from Figure 5.37 is designed to approximate the step based trajectory set point of Figure 5.35, but using x and y as smooth time functions. The (x, y) position is obtained from a polynomial approximation of the set point steps, assuming a constant linear speed of 0.133ms^{-1}. The polynomial functions used to describe the trajectory correspond to expressions (5.40) and (5.41). The time evolution of the coordinates is presented in Figure 5.36.

$$x(t) = 0{,}001293 + 0{,}1227\,t + 0{,}005685\,t^2 - 0{,}0004211\,t^3 - \\ -0{,}0001320\,t^4 + 1{,}755\times10^{-5}\,t^5 - 7{,}425\times10^{-7}\,t^6 + 1{,}052\times10^{-8}\,t^7 \tag{5.40}$$

$$y(t) = -0{,}002498 + 0{,}04288\,t - 0{,}03488\,t^2 + 0{,}008723\,t^3 - \\ -0{,}0008170\,t^4 + 3{,}740\times10^{-5}\,t^5 - 8{,}636\times10^{-7}\,t^6 + 8{,}103\times10^{-9}\,t^7 \tag{5.41}$$

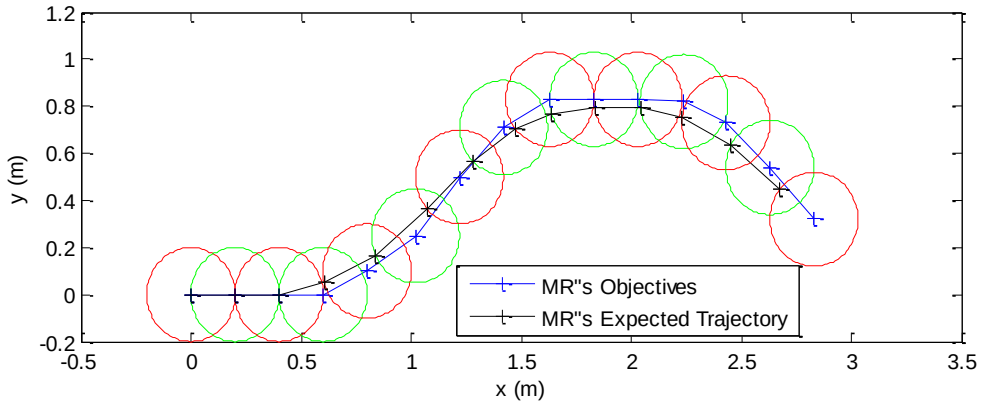

Figure 5.35 – Sequence of Set point Trajectory. Each objective has a red or green circle limiting the objectives area. Once the MR reaches that area, the current objective is replaced by the next one.

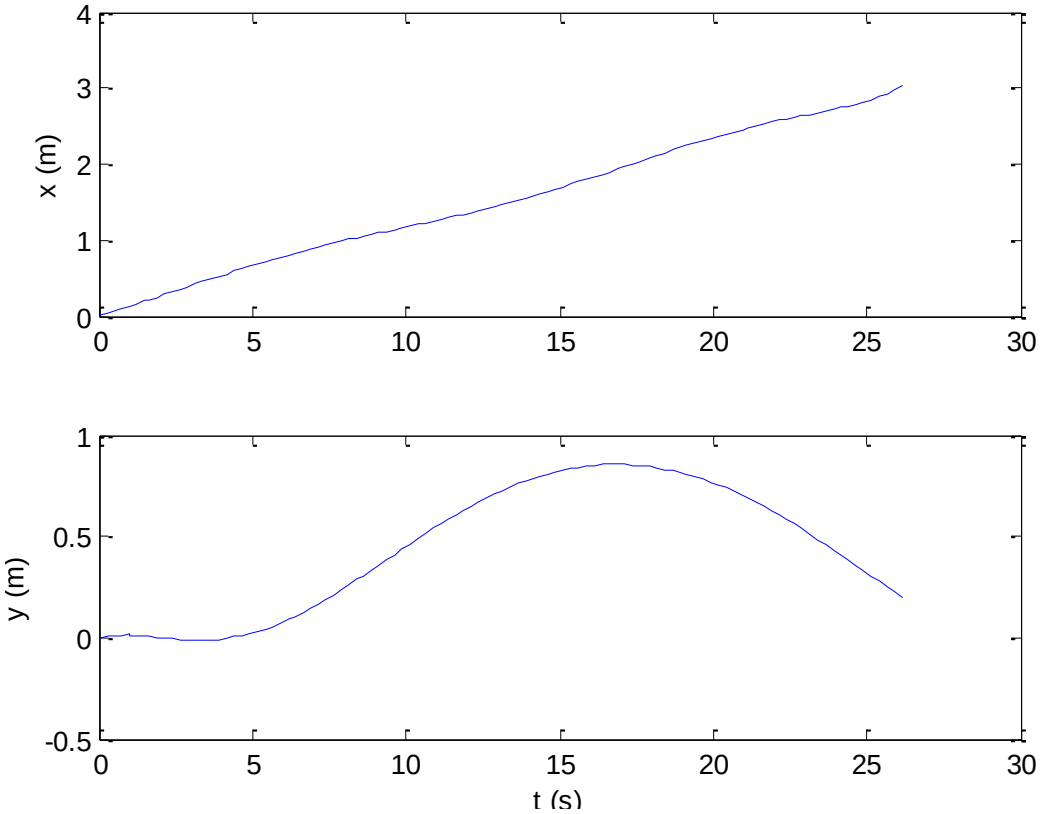

Figure 5.36 – Time variant Trajectory.

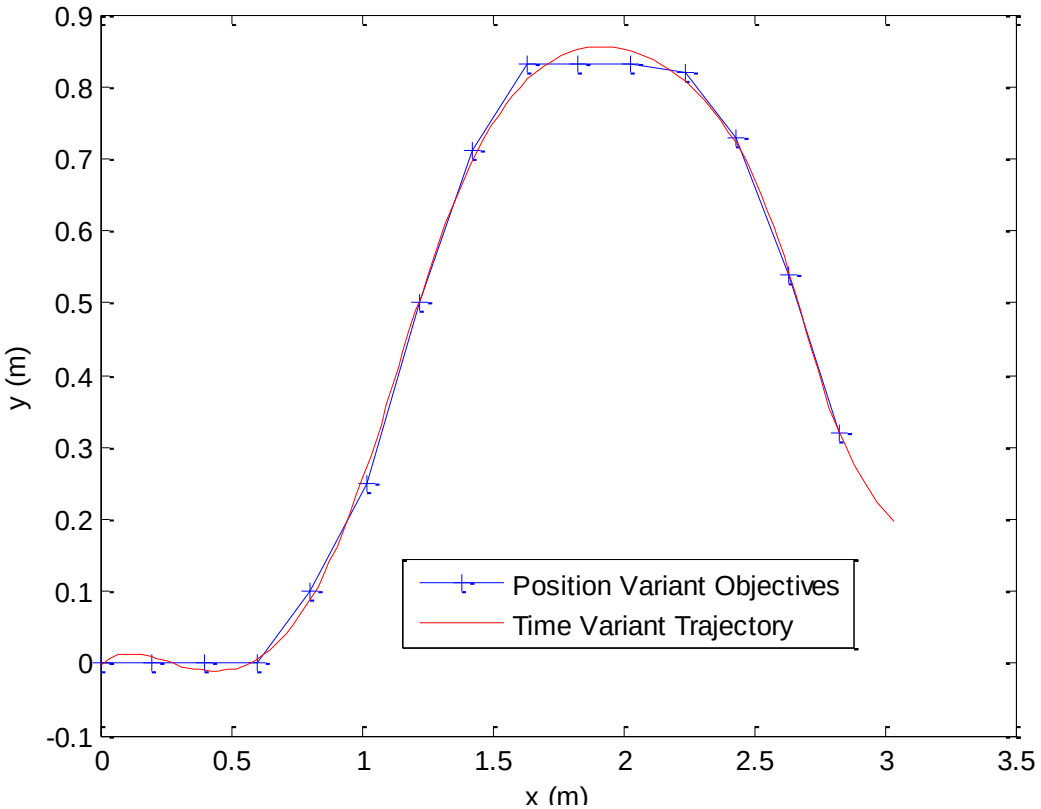

Figure 5.37 – Comparing Position variant Trajectory and Time variant trajectory.

This work is focused on the study the MR's position estimation rather than its actual position. This is due to the lack of external sensors to indicate the MR's actual position. The reference tracking method used on this controller is a serial gain block.

The trajectory (positional) graphs presented don't have the same scales on both axis so the difference between the different trajectories will be more visible.

5.9.2.1 Experiment 1 – Step Performance

This experiment aims to demonstrate the MR's reaction to a step response.

As the MR's output consists in position (x and y), two step signals were performed at the same time, one for the position x and the other for the position y. The MR initializes at the position $(0.0,0.0)$ and ends once it reaches the position $(1.0,0.3)$. The results are presented in Figure 5.38 and Figure 5.39. The weight and noise covariance matrices used for all these experiments were,

$$Q = \begin{bmatrix} 3.4\,e-6 & 0 & 0 & 0 & 0 \\ 0 & 3.2\,e-5 & 0 & 0 & 0 \\ 0 & 0 & 1\,e-7 & 0 & 0 \\ 0 & 0 & 0 & 1\,e-8 & 0 \\ 0 & 0 & 0 & 0 & 1\,e-8 \end{bmatrix}$$

$$Q_O = \begin{bmatrix} 1\,e-4 & 0 & 0 & 0 & 0 \\ 0 & 1\,e-4 & 0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 0 & 1.0 \end{bmatrix}$$

$$R = \begin{bmatrix} 1\,e-6 & 0 \\ 0 & 1\,e-5 \end{bmatrix} \qquad R_O = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.01 \end{bmatrix}$$

The variance factor used on these experiments is

$$v_f = 5000$$

On both runs of this experiment, with and without the use of variance weighting on the controller, the MR executes a curve right in the beginning of the movement, as it may be seen in Figure 5.39. This stems from the MR's initial orientation of 0 rad.

Both controllers present a similar behavior. On the R plot of Figure 5.39, the orientation from the controller without variance weighting starts to change a earlier. However, the resulting trajectory is not as close to the straight line as the other controller.

On both controllers the estimated final position is beyond the desired position:

- Without Variance weighting, the position estimate exceeds the set point by 1.6 cm, but the MR's actual position exceeds it by 2.5 cm.
- With Variance weighting, the position estimate exceeds the set point by 0.4 cm, but the MR's actual position is short by 6.0 cm.

This error in estimation may be associated with the wheels slipping due to the lack of friction with the floor.

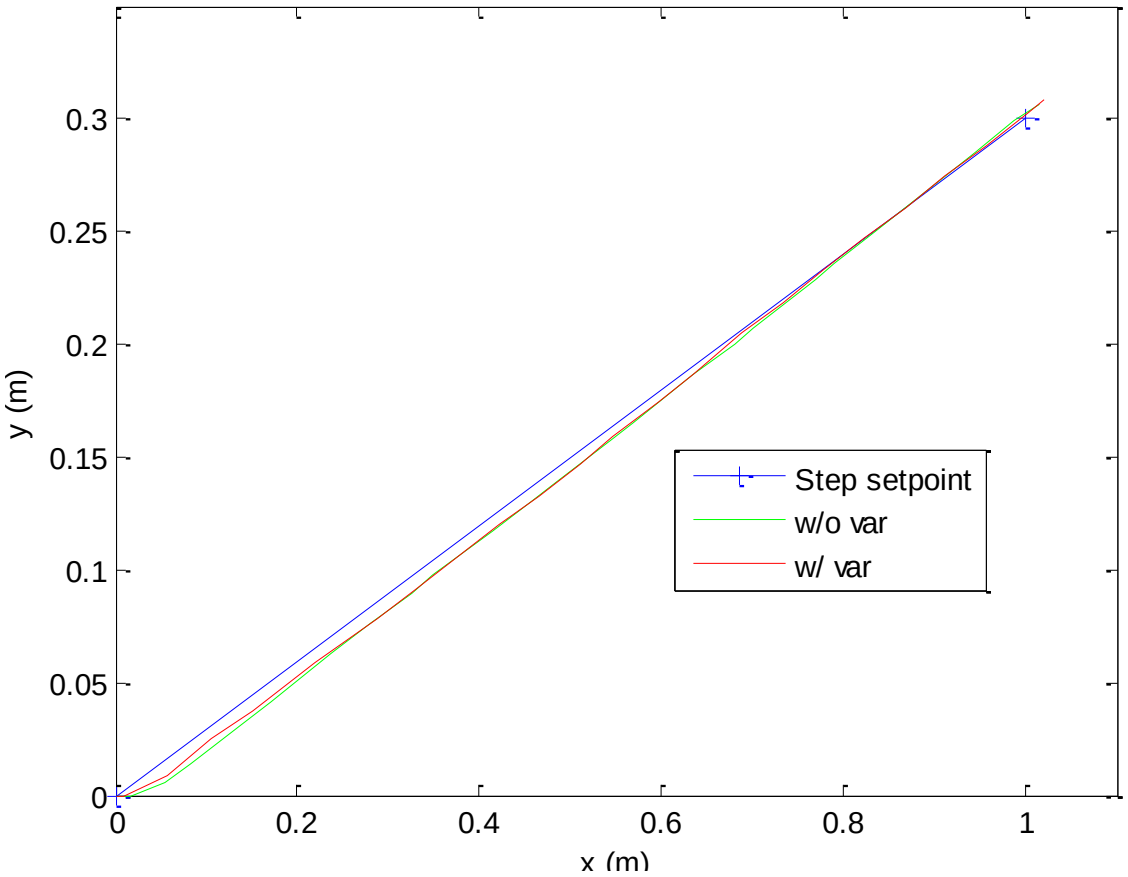

Figure 5.38 – Comparison between controlled MR,
with and without variance adaptation, in response to a step.

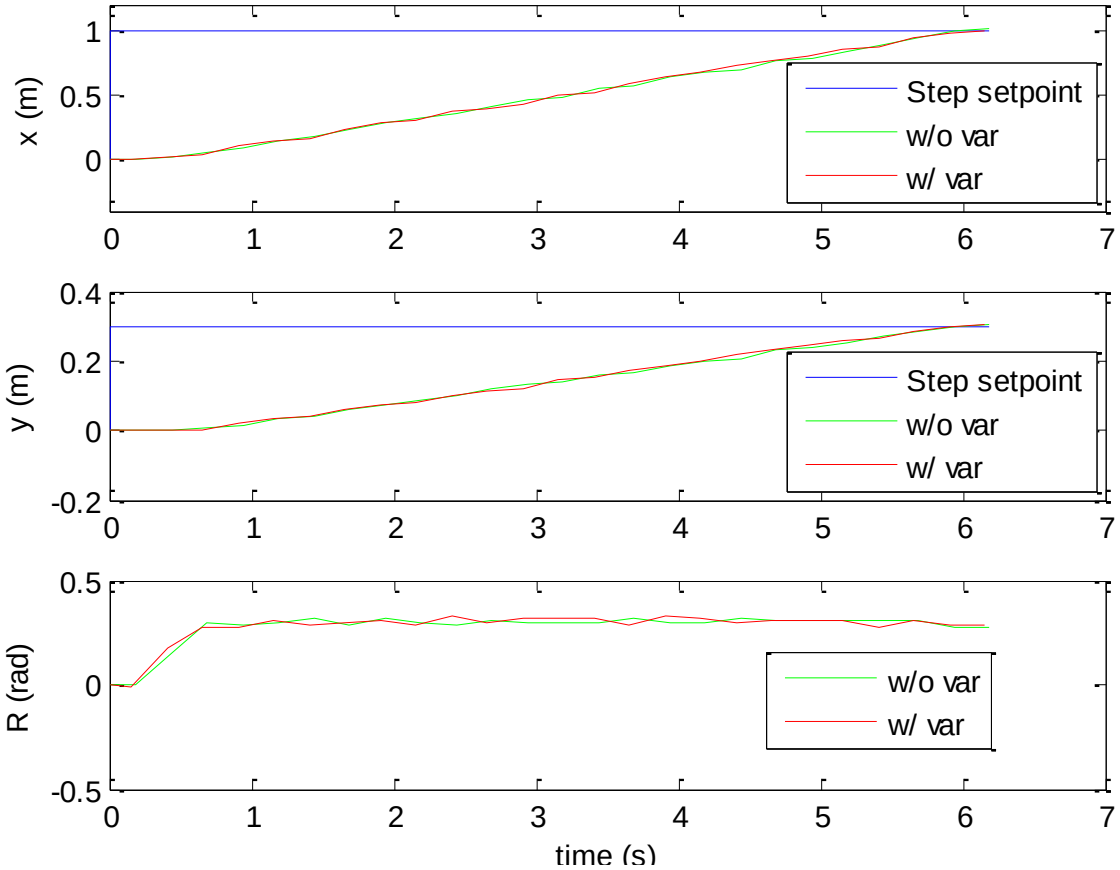

Figure 5.39 – Comparison between controlled MR,
with and without variance adaptation, in response to a step.

5.9.2.2 Experiment 2 – Sequence of steps on the set point

In this experiment, the MR is given a sequence of target positions to be reached. Once the MR distance to the first target position is under $20\ cm$ the set point is replaced by the next target on the sequence. The procedure is repeated until the last target on the sequence is reached. This set point policy aims the robot to drive close to target positions without actually crossing through them.

In Figure 5.40, the path labeled objective position corresponds to the perfect trajectory the MR should perform if it was able to instantly change its orientation towards each new target. This trajectory does not take into account the MR's dynamics.

The MR's close loop dynamics corresponds to a lag time that leads the robot closer to the targeted positions than the *Expected Trajectory*, from Figure 5.35. On both runs, with and without the use of variance adapter on the controller, the MR estimated gets close to, or even over, the objectives.

The controller with variance adapter performs better on following the objectives than the controller without variance adapter. The variance adapter slows down the MR's movement speed and delays the actuation of the rotation speed, allowing the MR to perform a tighter curve with smaller oscillation.

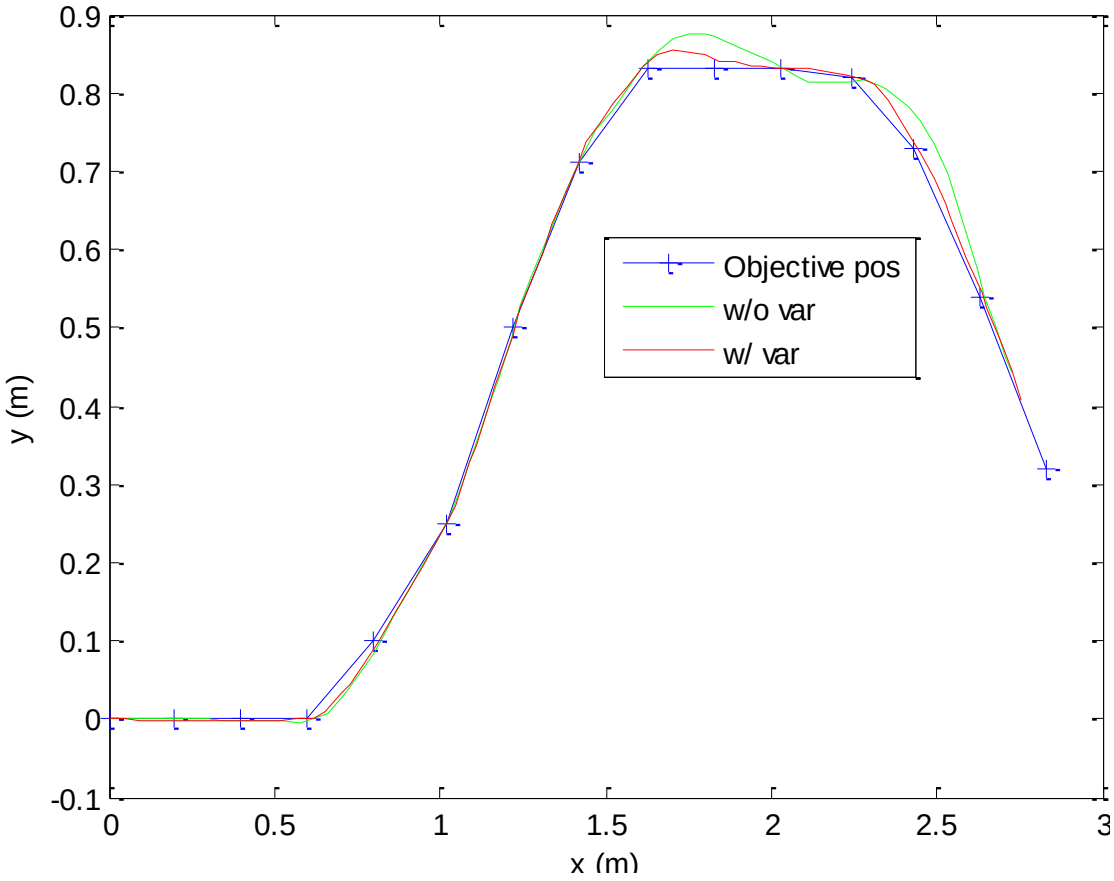

Figure 5.40 – Comparison between controlled MR, with and without variance adaptation, and Expected Trajectory.

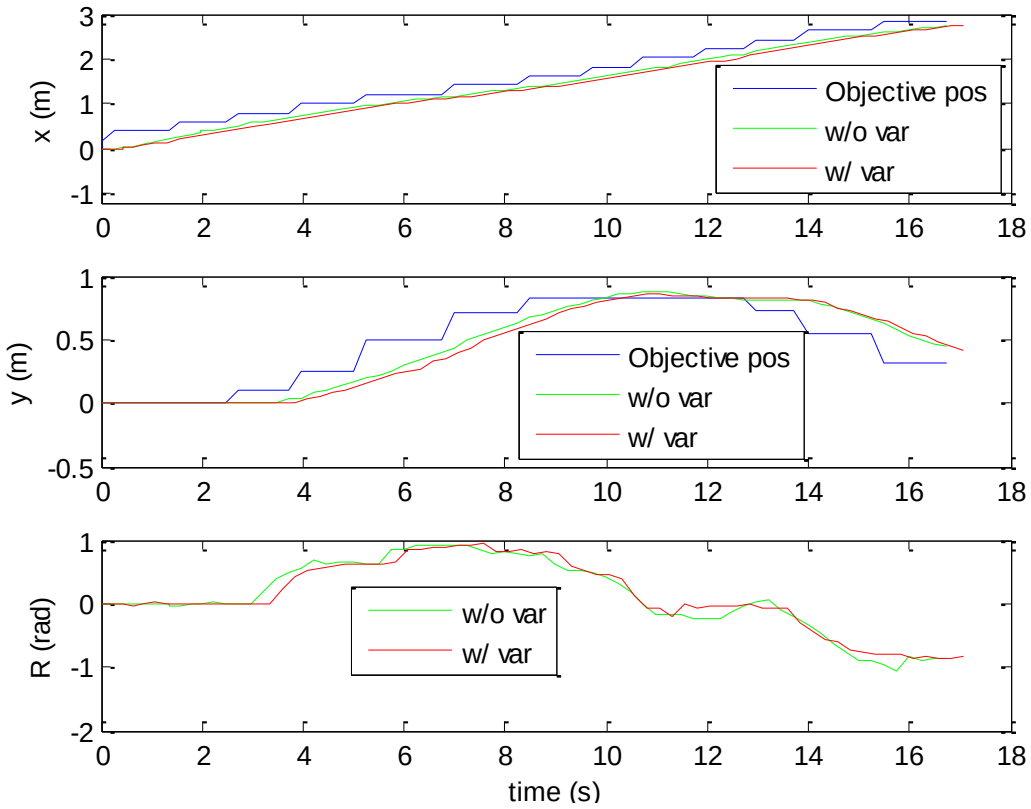

Figure 5.41 – Comparison between controlled MR, with and without variance adaptation.

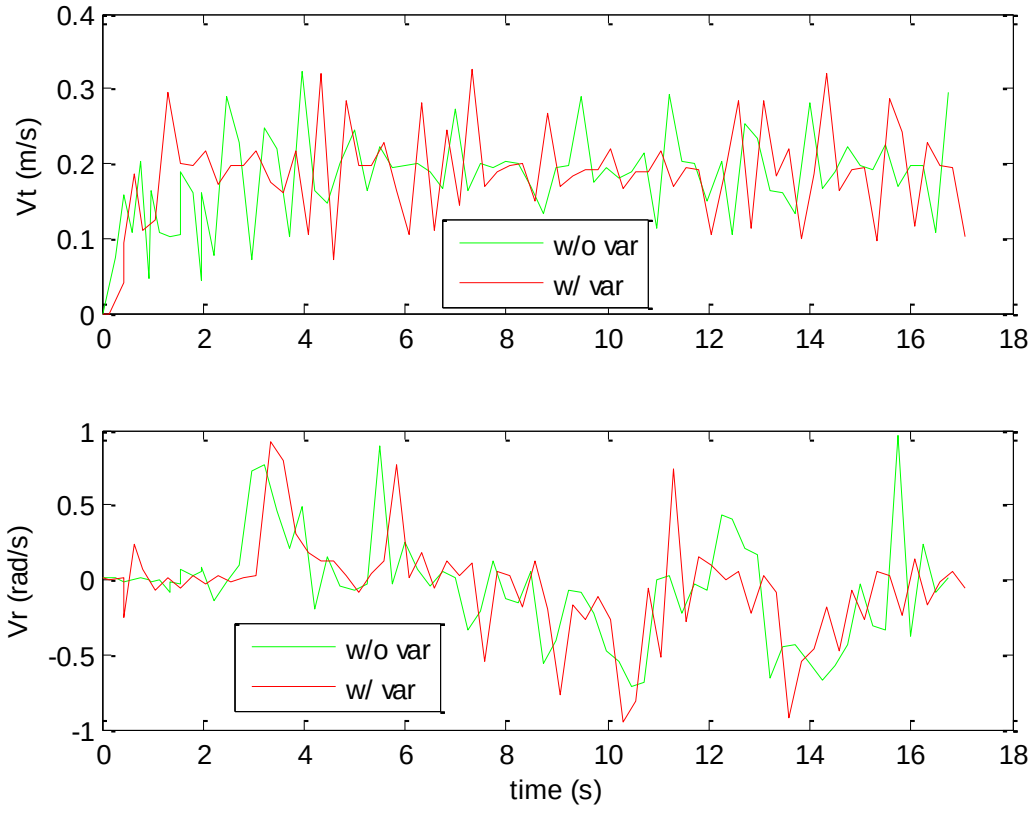

Figure 5.42 – Comparison between Control Signals applied on the MR, with and without variance adaptation.

To have a better notion of the variance effect on the MR's control other runs were perform with a higher variance factor. On the new run, using a variance factor of 500000 is compared, on the Figure 5.43 - Figure 5.47, with the two previous runs, one without variance adapter and the other with variance adapter ($v_f = 5\,000$). Although the new run ($v_f = 500\,000$) moves closer to the objective position, than the other two runs, it doesn't mean it is an improvement. The MR took 4 seconds longer to complete the trajectory than the other two, that's about 23% longer.

In this experiment, when the MR reaches close to the current objective, the objective is replaced by the next one on the sequence. This way the moment in which an objective position is used as set point may vary from run to run, as it is shown when the new run ($v_f = 500\,000$) moves slower than the other two (Figure 5.44 and Figure 5.45).

The high variance factor affects the forward speed with higher impact than the rotation speed, as it shows in Figure 5.47. This slow forward speed, although it improves the path performed, it slows down the controlled system.

These runs show the existence of a trade-off, between the curve performance and the time it takes to reach the objective. It is required a certain care when choosing the right variance factor.

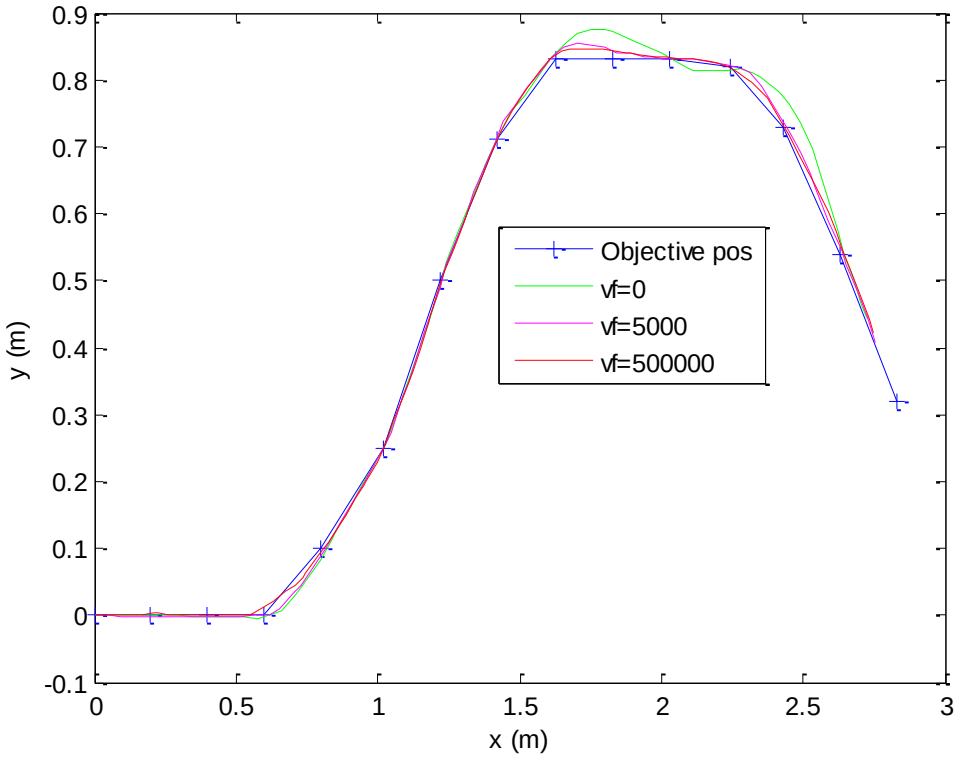

Figure 5.43 – Comparison between controlled MR, with different variance factors.

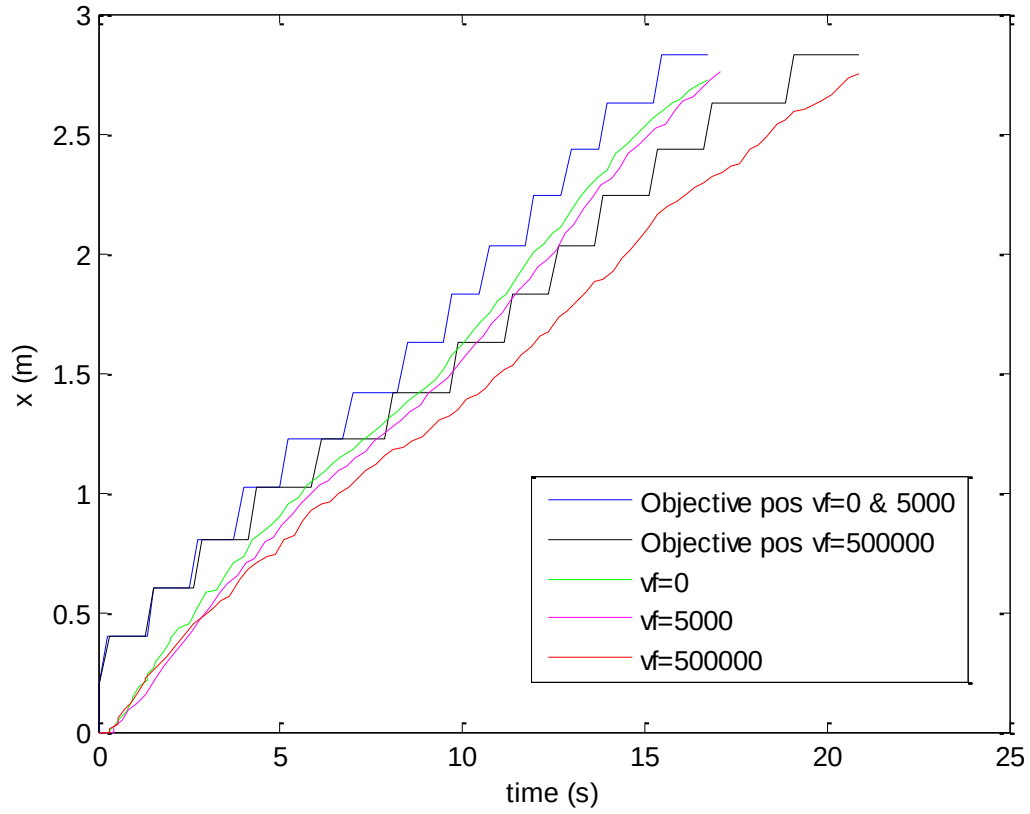

Figure 5.44 – Comparison between controlled MR's position $\boldsymbol{x}$, with different variance factors.

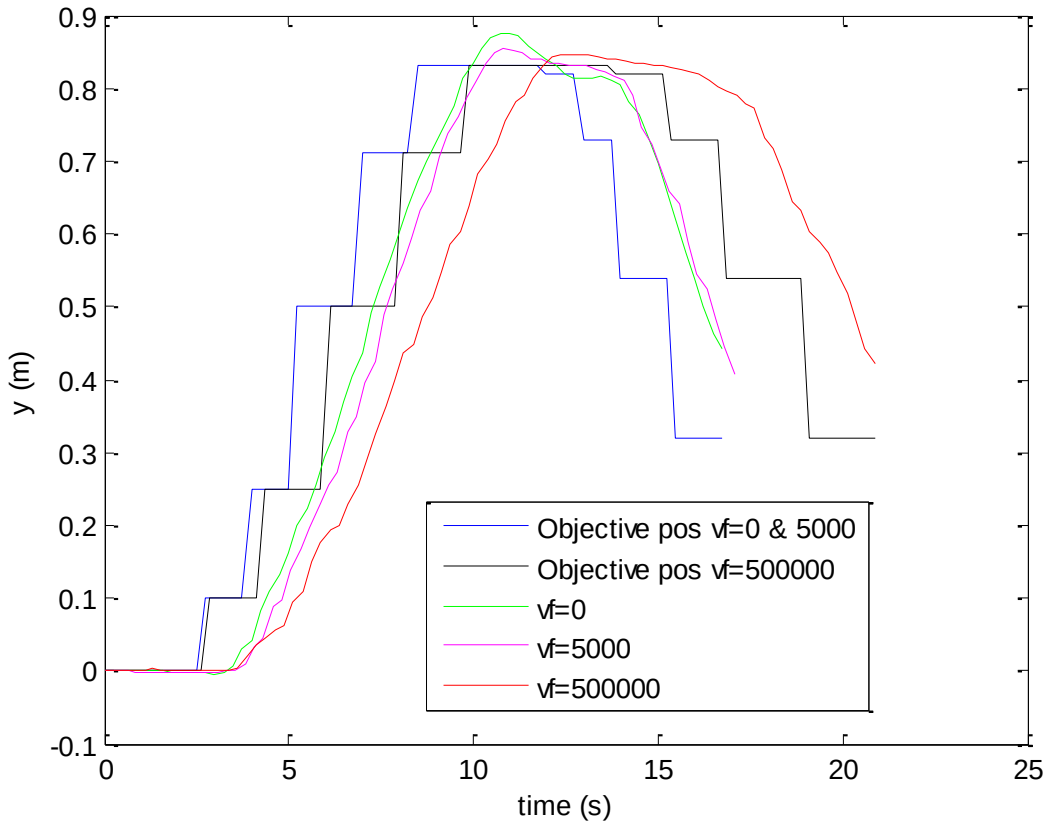

Figure 5.45 – Comparison between controlled MR's position $\boldsymbol{y}$, with different variance factors.

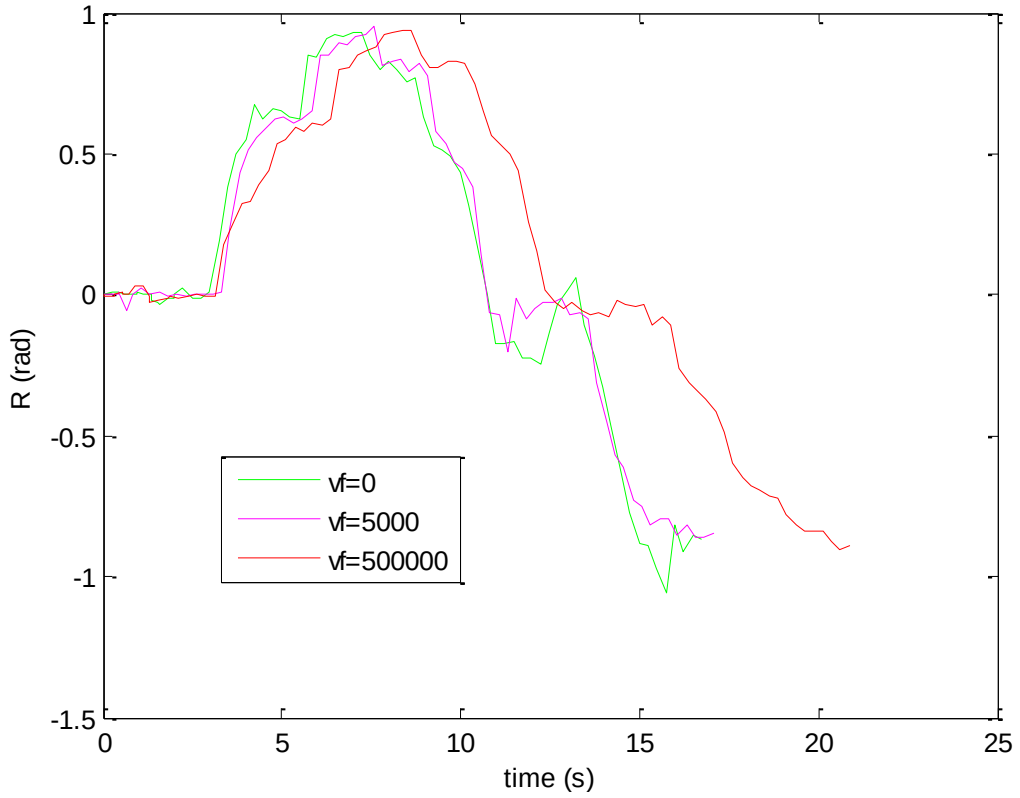

Figure 5.46 – Comparison between controlled MR's orientation ***R***, with different variance factors.

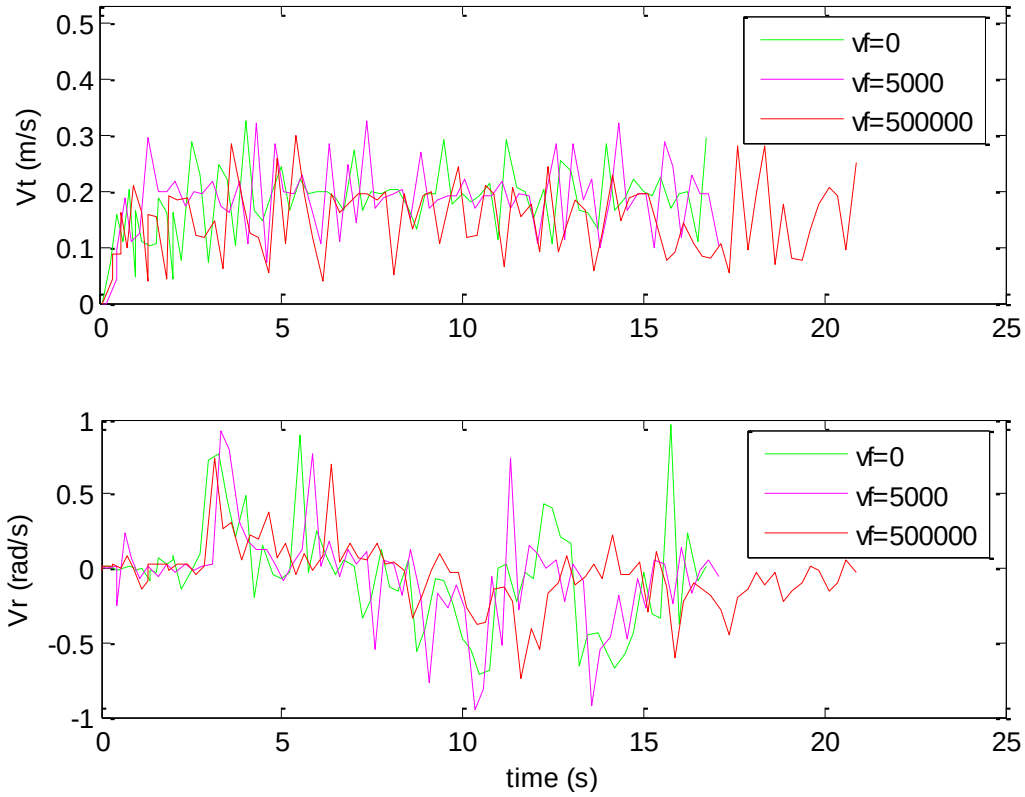

Figure 5.47 – Comparison between Control Signals applied on the MR, with different variance factors.

5.9.2.3 Experiment 3 – Time Varying Trajectory

In this experiment, the MR's objective has a time variant position, which the MR aims to follow it. This idea is a reproduction of a car chase, one car following another. The objective's position is represented by the equations (5.40) and (5.41).

$$x(t) = 0{,}001293 + 0{,}1227\,t + 0{,}005685\,t^2 - 0{,}0004211\,t^3 - \\ -0{,}0001320\,t^4 + 1{,}755 \times 10^{-5}\,t^5 - 7{,}425 \times 10^{-7}\,t^6 + 1{,}052 \times 10^{-8}\,t^7 \tag{5.40}$$

$$y(t) = -0{,}002498 + 0{,}04288\,t - 0{,}03488\,t^2 + 0{,}008723\,t^3 - \\ -0{,}0008170\,t^4 + 3{,}740 \times 10^{-5}\,t^5 - 8{,}636 \times 10^{-7}\,t^6 + 8{,}103 \times 10^{-9}\,t^7 \tag{5.41}$$

In Figure 5.40 and Figure 5.41 are presented two different tests, one with the use of variance adaptation on the controller and the other without it.

As in the previous tests the orientation (R) set point is defined recursively assuming that, at each moment, there is some offset between the MR actual position and the (x, y) target. The set point for R is computed so that the robot is oriented towards the (x, y) target. This orientation set point policy is responsible for the fast orientation changes that are visible during the first seconds of each test, when the position offset is small.

The two tests present a similar behavior, for the exception of the first 3 seconds. This close approximation indicates that the state variance is very little, causing very little effect on the controller. This initial oscillation demonstrates the variance effect, by reducing the amplitude of oscillation of the trajectory performed by the MR.

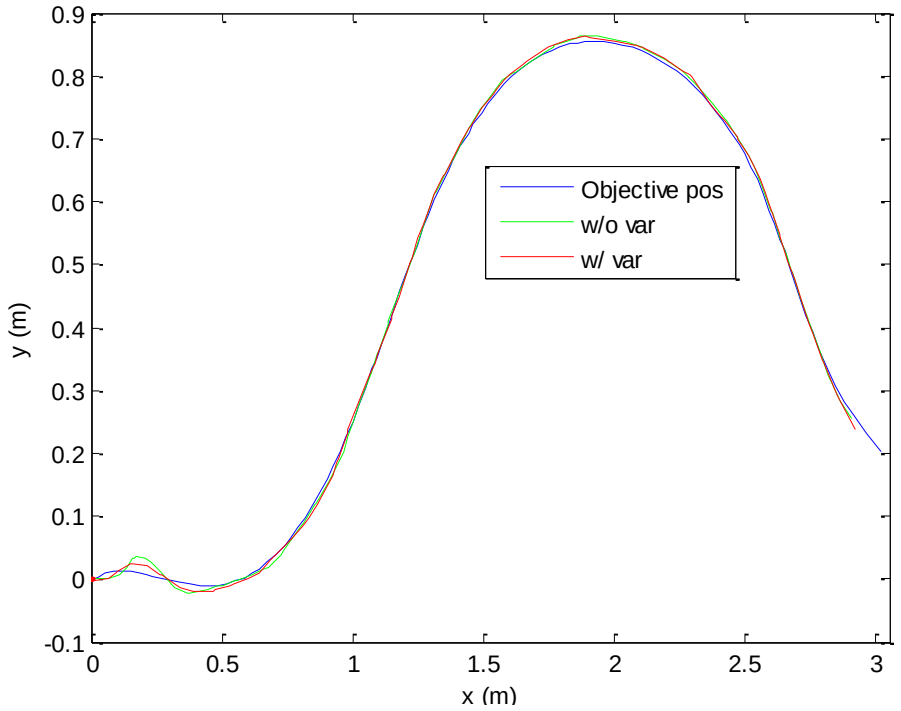

Figure 5.48 – Comparison between controlled MR, with and without variance adaptation.

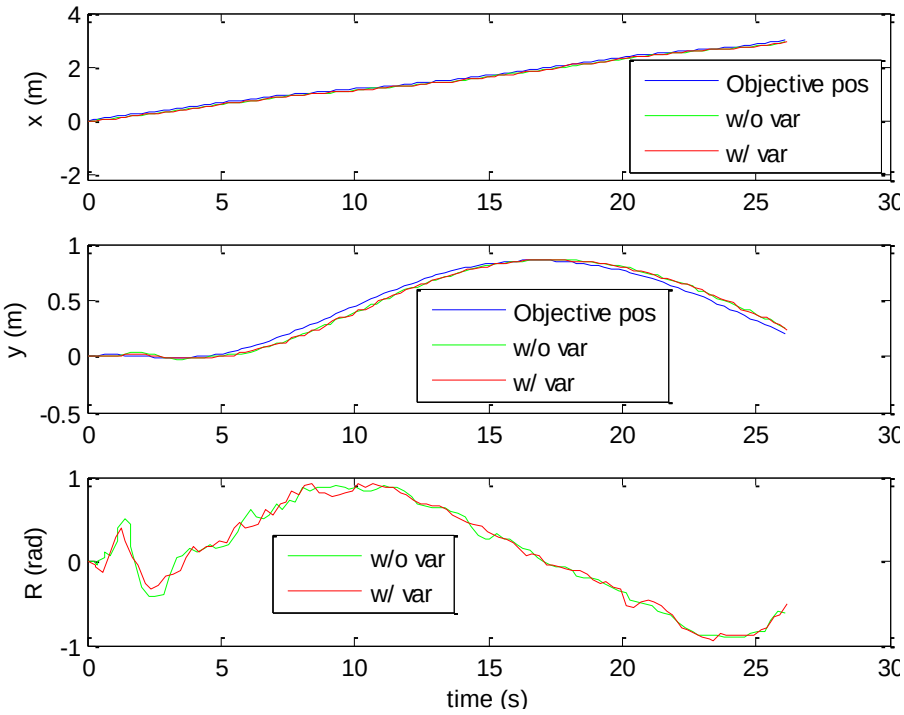

Figure 5.49 – Comparison between controlled MR, with and without variance adaptation.

A third run was performed, with $v_f = 500000$. On a test presented on the previous section this modification leads to a decrease on the overall speed of the robot. This new test aims to verify if a similar effect occurs while tracking this set point signal.

In Figure 5.50, at first glance, it seems this new run performs as well as the other two, or even better, taking into account the initial oscillation.

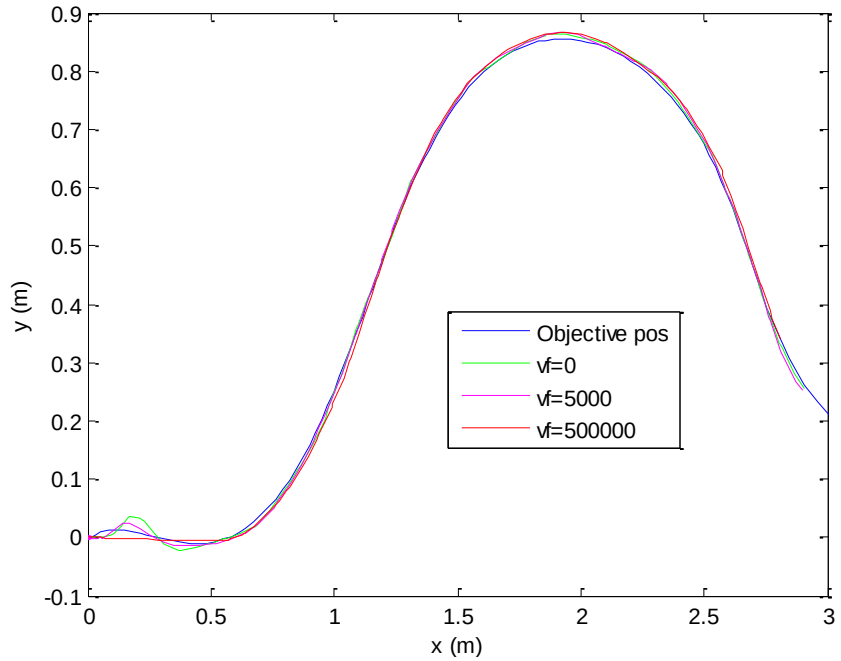

Figure 5.50 – Comparison between controlled MR, with different variance factors.

The reason why this run didn't perform the initial oscillation is because the robot is kept still during that time (the initial 2 seconds), as it may be seen in Figure 5.51 -Figure 5.54.

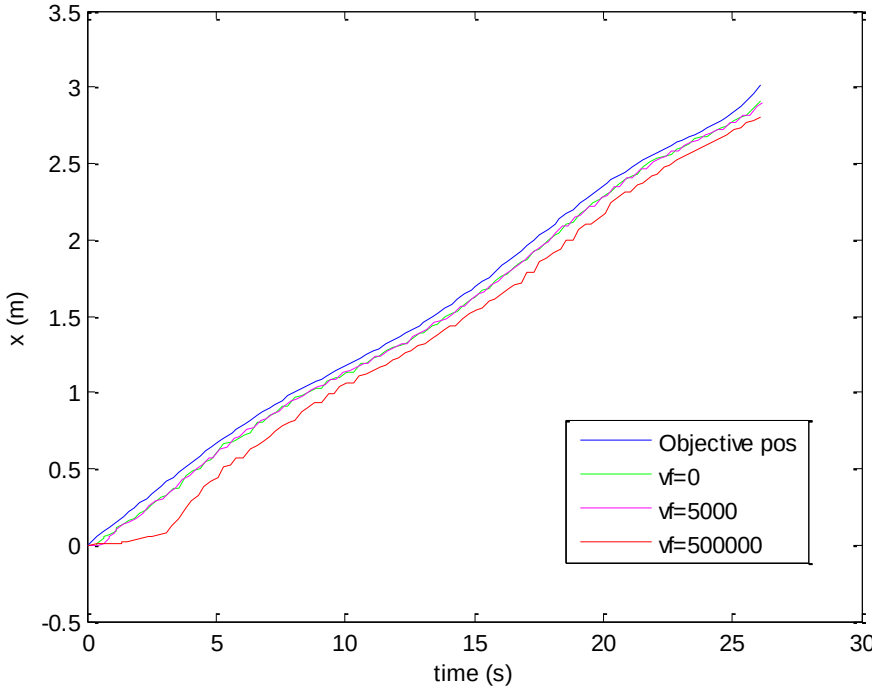

Figure 5.51 – Comparison between controlled MR's position x, with different variance factors.

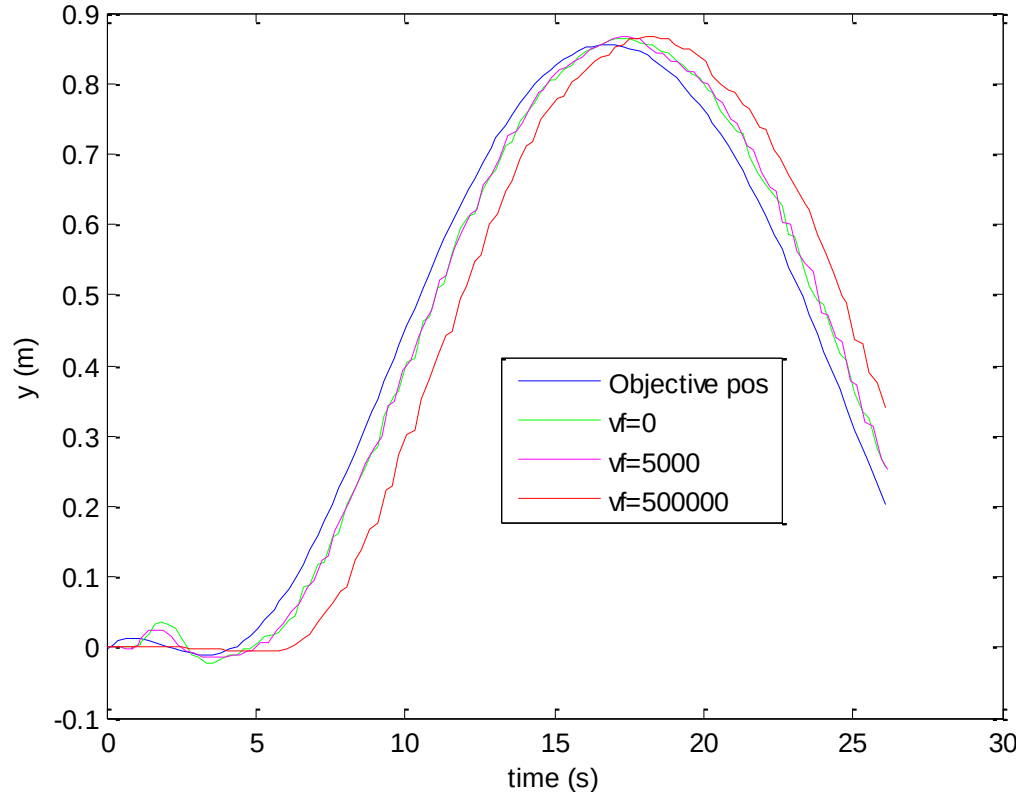

Figure 5.52 – Comparison between controlled MR's position *y*, with different variance factors.

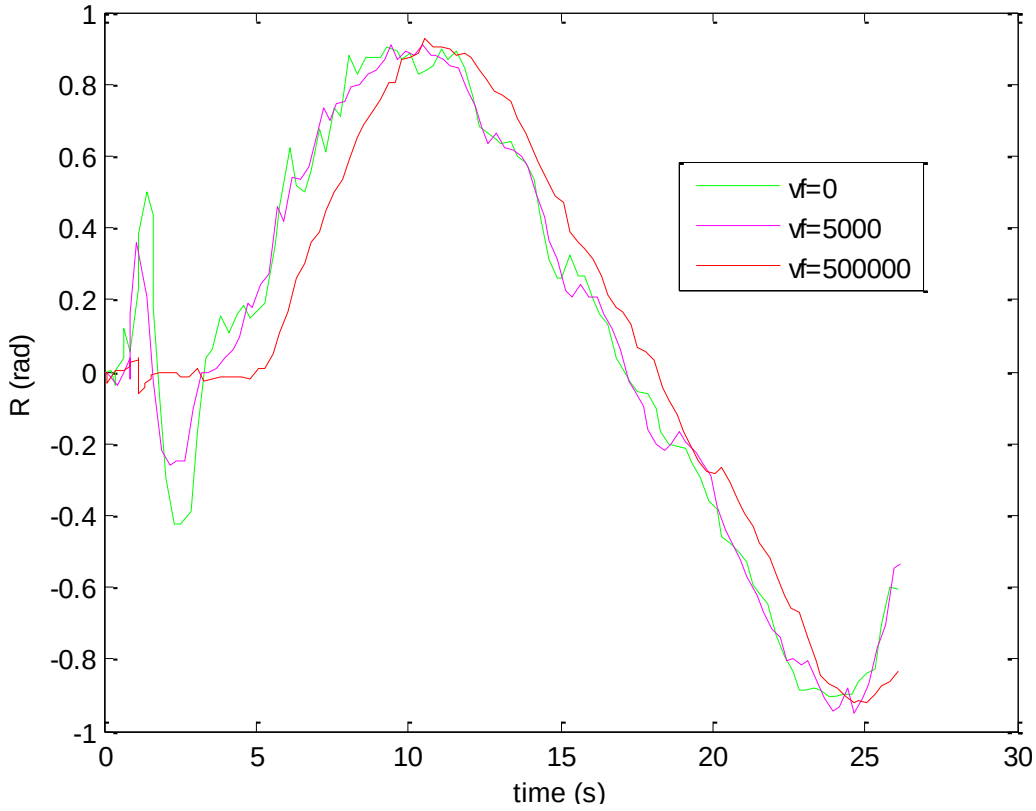

Figure 5.53 – Comparison between controlled MR's orientation *R*, with different variance factors.

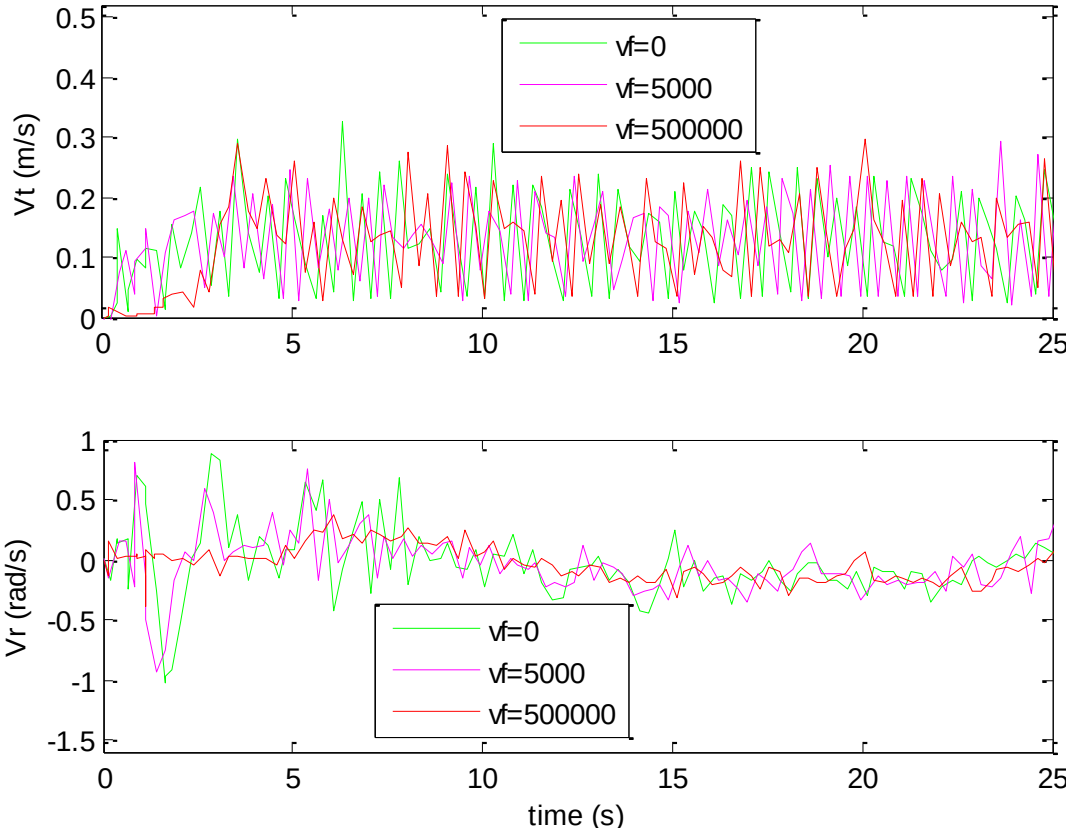

Figure 5.54 – Comparison between Control Signals applied on the MR, with different variance factors.

During the tests, neither of the runs is ever on top of the objective position, for the exception of the initial instant. But they always keep a constant distance from it. The two first runs ($v_f = 0$ and $v_f = 5000$) always keep a distance of about $8cm$. The third ($v_f = 500000$) kept a distance of about $20cm$, making it about 150% further compared to the other 2.

5.10 Summary

The chapter presented the basic structure of a LQR controller, using a Kalman Filter as state observer. Two different approaches were used to track the controller's set point: integral effect and series gain compensation. The latter is applied to the MR controller. A number of simulation studies were presented on both linear and nonlinear dynamics. The controller was tested on a small lab-scale MR.

A modification was introduced to the standard LQR controller, in order for the variance of the state estimate to affect the control action. Comparisons were made between the controller with and without this variance adaptation. For comparison were used simulations, such as the non-linear model of the inverted pendulum, and experiments using a mobile robot.

With these results it was confirmed that the variance adaptation on the controller improved the system's performance on the tracking of the desired reference or set point. This adapter also brings the "slow down" of the controller's action. On some applications a slower controlled system is a desirable feature.

However on other situations the "slow down" of the controller's action may be seen as a drawback effect. This is for example, the case of the unstable inverted pendulum where it causes a drop on performance, conducing to an increase on the overshoot.

6 Conclusion

This dissertation addresses the problem of developing an Autonomous Mobile Robot (AMR). The robot is aimed to work within a partially unknown environment that suffers non-deterministic time driven changes. The dissertation covered a set of topics related to this subject.

For an AMR to move around the surrounding environment needs to known its position and the surrounding environment, so it needs a map. As the map of the surrounding environment isn't always available, so the MR has to construct its own. For that, it needs to be able to sense its surroundings, using obstacle sensing devices such as ultrasonic sensors, and to estimate its position, using maps and displacement sensors such as IMU and odometry.

The obstacle detection sensor installed on the MR is an ultrasonic sensor on a rotating support. The sensor was tested and its main features either as an obstacle detector and a distance sensor were analyzed. The possibility of using the sensor for finding the obstacles profile was also addressed. It was found that the available sensor is adequated only for obstacle detection. However its detection range is strongly dependent on the obstacle surface material and geometry. Its use as a distance measuring device is compromised by its inaccuracy over distances shorter than $1m$. For that purpose a different sensor is required.

Obstacle detection and mapping are two closely subjects. Detected obstacles may either be part of the already known map or be newly discovered environment features. Their position may also be used to help finding the robot position on the map.

The dissertation briefly addressed the SLAM problem. A general and simplistic EKF-SLAM algorithm, which uses Feature-Based Mapping was presented.

The Decision Making problem is an important subject for the development of the AMR. Within this topic the planning of the robot trajectory over the working area is addressed. A Path Planning Algorithm was developed through the use of Reinforcement Learning. Using a Grid Based Map, it was able to produce a trajectory for the MR to follow from the "current" to the "desired" position.

Robot Localization using Sensor Fusion is a way for finding the robot's position within the environment. The robot's localization is done through information provided by sensors and systems, such as Inertial, Image, and Odometry. This work took an interest on the possibility of

sensors working asynchronously, as sometimes happens when using an image based sensor. Experiments using two different Mobile Platforms were presented. Their position and path taken were estimated offline. The information, provided by one of the MRs, was used on three different situations of Data Fusion. From these experiments was concluded that the Data Fusion period mustn't be too large, for the prediction error tends to grow. This work lead to the publication of a conference paper [Coito14].

The Problem of controlling the robot's movement has been dealt with as well. A LQR based controller, together with an EKF state observer, was used with the aim of tracking the desired trajectory. The controller was modified in order to manage the robot's speed depending on the position uncertainty. This controller was tested in a number of simulations using different plants. The controller was also applied to the real time control of the trajectory from a MR.

This work approached the topic of implementing an autonomous mobile robot by addressing several relevant subjects. Any of these subjects requires a deeper research, and needs a more careful study, with special attention to sensorial perception, focusing on other types of sensors, or on their combination. Above all, it is also necessary to develop a Supervising System, capable of connecting together the various tools, to manage them into a coherent system, to analyze their performance and detect the various possible faults.

For future work, a deeper attention needs to be given to the various topics related to this subject.

One of the requirements is the study of a more flexible obstacle detection system. The possibility is the study of a more flexible ultrasonic sensor, capable of providing the full emitted and received signals, and the management of the emitted signal's power. The use of other sensors, such as infrared laser sensors and cameras, or a combination of several sensors into a multisensory obstacle detection system should be studied.

Another important aspect for future development is the implementation of a mapping system. The use of different types of sensors is useful for characterization of the environment and for localization purposes.

In the next step it is planned to improve the localization system, based on the fusion of inertial and odometry sensors, and to complete the sensor fusion, by the inclusion of the visual odometry system, such as the one proposed in [Coito14].

An important element, still missing, is the supervising system, capable of coordinating all the remaining elements and allowing the robot to autonomously perform complex tasks.

References

Aranib04	Aranibar, D., Alsina, P.: Reinforcement Learning-Based Path Planning for Autonomous Robots. In: Encontro Nacional de Robótica Inteligente no XXIV Congresso da Sociedade Brasileira de Computação, Salvador, Brazil (2004)
Armest04	Armesto, L., Chroust, S., Vincze, M., Tornero, J.: Multi-rate fusion with vision and inertial sensors. In: IEEE International Conference on Robotics and Automation, pp. 193 - 199. IEEE Press, New York (2004)
Ayteki10	Aytekin, M., Mao, B., Moss, C.: Spatial perception and adaptive sonar behavior. Journal of the Acoustical Society of America, 128-6, pp. 3788-3798 (2010)
Bailey06	Bailey, T., Durant-Whyte, H.: Simultaneous Localisation and Mapping (SLAM): Part II State of the Art. IEEE, Robotics & Automation Magazine, 13-3, pp. 108-117 (2006)
Bancro11	Bancroft, J. B., Lachapelle, G.: Data Fusion Algorithms for Multiple Inertial Measurement Units. Sensors. 11-7, pp. 6771-6798 (2011)
Barraq91	Barraquand, J., Latombe, J. C.: Robot Motion Planning: A Distributed Representation Approach. The international Journal of Robotics Research, 10-6, pp. 628-649 (1991)
Borens88	Borenstein, J., Koren, Y.: Obstacle avoidance with ultrasonic sensors. IEEE Journal of Robotics, 4-2, pp. 213-218 (1988)
Bruce02	Bruce, J., Veloso, M.: Real-Time Randomized Path Planning for Robot Navigation. IEEE International Conference on intelligent Robots and Systems, 3, pp. 2383-2388 (2002)
Calond06	Calonder, M.: EKF SLAM vs. FastSLAM - A Comparison. Swiss Federal Institute of Technology, Lausane (2006)
Cardin07	Cardin, S., Thalman, D., Vexo, F.: A wearable system for mobility improvement of visually impaired people. The Visual Computer: International Journal of Computer Graphics, 23-2, pp. 109-18 (2007)
Cho11	Cho, Bong-Su, Moon, Woo-sung, Seo, oo-Jin, Baek, Kwang-Ryul: A dead reckoning localization system for mobile robots using inertial sensors and wheel revolution encoding. Journal Mechanical Science and Technology, 25-11, pp. 2907-2917 (2011)
Coito14	Coito, F., Eleutério, A., Valtchev, S., Coito, F.: Tracking a Mobile Robot Using Vision and Inertial Sensor. DoCEIS, Technological Innovation for Collective Awareness Systems, pp. 201-208 (2014)
Dijkst59	Dijkstra, E.: A Note on Two Problems in Connexion with Graphs, Numerische MAthematik, 1-1, pp. 269-271 (1959)

References

Durant06	Durant-Whyte, H., Bailey, T.: Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. IEEE, Robotics & Automation Magazine, 13-2, pp. 99-110 (2006)
Hart68	Hart, P., Nilson, N., Raphael, B.: A Formal Basis for Heuristic Determination of Minimum Cost Paths. IEEE Transitions on Systems Science and Cybernetics, 4-2, pp. 100-107 (1968)
Hol07	Hol, J., Schön, T., Luinge, H., Slycke, P., Gustafsson, F.: Robust real-time tracking by fusing measurements from inertial and vision sensors. J. Real-Time Image Proc. 2-3, pp. 149-160 (2007)
Hu10	Hu, Y., Duan, Z., Zhou, D.: Estimation Fusion with General Asynchronous Multi-Rate Sensors. IEEE Trans. Aerosp. Electr. Sys. 46-4, pp. 2090 - 2102 (2010)
Kim13	Kim, S. J., Kim, B. K.: Dynamic Ultrasonic Hybrid Localization System for Indoor Mobile Robots. IEEE Trans. Ind. Elect. 60-10, pp. 4562 - 4573 (2013)
Kurt12	Kurt-Yavuz, Z. Yavuz, S.: A Comparison of EKF, UKF, FastSLAM2.0, and UKF-based FastSLAM Algorithms. IEEE 16th International Conference on Intelligent Engineering Systems, Lisbon, pp. 37-43 (2012)
LaValle98	LaValle, S.: Rapidly-exploring Random Trees: A new tool for path planning. Report No. TR 98-11, Computer Science Department, IOWA State University (1998)
Lee09	Lee, T., Shirr, J., Cho, D.: Position Estimation for Mobile Robot Using In-plane 3-Axis IMU and Active Beacon. In: IEEE International Symposium on Industrial Electronics, pp. 1956--1961. IEEE Press, New York (2009)
Lucas10	Lucas, A., Christo, C., Silva, M.P., Cardeira, C.: Mosaic based flexible navigation for AGVs. In: IEEE International Symposium on Industrial Electronics. pp. 3545 - 3550. IEEE Press, New York (2010)
Marin13	Marín, L., Vallés, M., Soriano, Á., Valera, Á., Albertos, P.: Multi Sensor Fusion Framework for Indoor-Outdoor Localization of Limited Resource Mobile Robots. Sensors, 13-10, pp. 14133-14160 (2013)
Namins13	Naminski, M.: An Analysis of Simultaneous Localization and Mapping (SLAM) Algorithms. Mathematics , Statistics, and Computer Science Honors Project, Paper 29 (20113)
Ping13a	PING))) Ultrasonic Distance Sensor (#28015). Parallax Inc. (2013)
Ping13b	Detect Distance with the Ping)))(TM) Ultrasonic Sensor. Parallax Inc. (2013)
Ribeir04	Ribeiro, M.: Kalman and Extended Kalman Filters: Concept, Derivation and Properties. Institute for Systems and Robotics, Instituto Superior Técnico, Portugal (2004)

References

Riisga05	Riisgaard, S., Blas, M.: SLAM for Dummies, A Tutorial Approach to Simultaneous Localization and Mapping. MIT OCW (2005)
Rosten10	Rosten, E., Porter, R., Drummond, T.: Faster and Better: A Machine Learning Approach to Corner Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32-1, pp. 105-1019 (2010)
Semush12	Semushin, I.; Adaptation in Stochastic Dynamic Systems – Survey and New Results III: Robust LQ Regulator Modification. Int. J. Communications and Systems, Russia (2012)
Skrzyp09	Skrzypczyński, P.: Simultaneous Localization And Mapping: A Feature–Based Probabilistic Approach. International Journal of Applied Mathematics and Computer Science, 19-4, p.575-588 (2009)
Stentz94	Stentz, A.: Optimal and Efficient Path Planning for Partially-Known Environments. Proceedings of IEEE International Conference on Robotics and Automation, 4, pp. 10-3317 (1994)
Sutton98	Sutton, R. Barto, A.: Reinforcement Learning; An Introduction. MIT Press(1998)
Tennina11	Tennina, S., Valletta, M., Santucci, F., Di Renzo, M., Graziosi, F., Minutolo.: Entity Localization and Tracking: A Sensor Fusion-based Mechanism in WSNs. In: 13th IEEE 13th International Conference on Digital Object Identifier, pp. 983 – 988. IEEE Press, New York (2011)
Thrun05	Thrun, S., Bugard, W., Fox, D.: Probabilistic Robotics. MIT Press (2005)
Valenc13	Valencia, R., Morta, M., Andrade-Cetto, J., Porta, J.: Planning Reliable Paths with Pose SLAM. IEEE Transactions on Robotics, 29-4, pp. 10500-1059 (2013)