Sensor fusion methods for indoor navigation using UWB radio aided INS/DR

JOSÉ BORRÀS SILLERO

KTH Electrical Engineering

Master's Degree Project
Stockholm, Sweden July 2012

XR-EE-SB 2012:015

Abstract

Some applications such as industrial automation, cargo handling, warehouse managing, monitoring of autonomous robot or personnel localization, require reliable indoor positioning. In particular, the requirement is to accurately localize a mobile wireless node in real-time. In outdoor scenarios, Global Navigation Satellite Systems (GNSSs) are commonly used for positioning. Nevertheless, they present notable shortcomings for being used in indoor applications. The principal disadvantages are the low accuracy achieved and the attenuation and reflections introduced by buildings and other structures, which make impossible their use in most cases.

The fixed and smaller operational area of an indoor positioning application in comparison with the area of an outdoor application makes it possible to install a radio positioning infrastructure. Concretely, the Ultra Wide Band (UWB) positioning system considered in this thesis consists of nodes installed in fixed and known positions (slaves) and a mobile node of unknown position (master) which is the one to be localized. The operation of the system is based on calculating the distance from the master to the slaves by measuring the Round-Trip-Time of a UWB pulse. In this way an estimate of the position can be obtained, with the advantage of having bounded errors, but, in contrast, the estimations have a low dynamic range and other navigation information cannot be measured.

The master has some sensors attached, an Inertial Measurement Unit (IMU) and a Dead-Reckoning (DR) system. By propagating the navigation equations, estimates of the position and other navigational states can be obtained, but with the disadvantage of having errors that grow with time.

The solution of the problems related to the individual location systems is using information fusion algorithms. Such algorithms, by means of integrating different sources of data, in this case the UWB range measurements and the IMU and DR measurements, create a positioning system which combines the benefits of the individual systems.

In this project the aim is to develop, implement and evaluate sensor fusion algorithms. In particular, the developed algorithms are based on the Extended Kalman Filter and the Particle Filter. Furthermore, they have been adapted to different dynamic models, in order to find the algorithms which fit better with the motion of a mobile node.

The developed algorithms have been tested with simulated trajectories using Matlab, and with real experimental datasets acquired by a mobile robot. The results show the benefits of the information fusion, since the accuracies obtained in the estimations outperform the accuracies obtained with the individual systems. Also in the most unfavorable cases, when one of the sources has high errors in the measurements, the algorithms are able to discard the useless information and estimate using only the useful measurements, proving the robustness of the system.

Contents

Chapter 1

Introduction

Background

There is a multitude of applications in which reliable indoor positioning is required, such as efficient traveling, industrial automation, cargo handling, warehouse management, monitoring of autonomous robots or personnel localization. In all cases the accurate localization of a mobile wireless node is desirable.

Global navigation satellite systems (GNSSs), such as the GPS (global positioning system), are commonly used for positioning in outdoor applications, however they have important shortcomings for being used in indoor environments. First of all, the accuracy obtained, which is in the order of 3 to 10 meters, is not sufficient for indoor applications. Furthermore, buildings and other structures introduce signal attenuation and reflections, making impossible their use in most cases [1].

Taking into account that the operational area of an indoor positioning system is fixed and smaller than the area of an outdoor system, it is possible to install a radio positioning infrastructure. This infrastructure consists of known-position nodes which are called anchors or slaves in the literature. In this thesis they will be referred to as slaves.

There are some radio technologies that can be suitable for this radio positioning system. Some of them are built on pre-existing communication infrastructure, for example Wireless Local Area Network (WLAN) or a Zigbee sensor network. But other solutions are designed specifically for indoor positioning. This is the case of the system treated in this project, a pulse-based Ultra-Wideband (UWB) system.

In the indoor scenario considered in this project, a mobile node of unknown position, which is called master sends UWB pulses to the slaves, and after some fixed time, they send a response. Measuring the time between the first pulse and the response, called round-trip-time (RTT), the master is able to obtain its distance with respect to the slaves.

One of the advantages of using the UWB pulses is the fine time resolution that is achievable due to the short transition time and the short duration of the pulses. Other benefits are the resilience to multipath propagation effects and the low-power device implementation.

However, for maintaining the high accuracy of RTT measurements a good knowledge of the system is important: the delay time of the slaves response and the circuitry, propagation errors, shape of the pulses or the selection of the correct threshold for the pulse detection are features that must be taken into account. Calibration of the system during the installation of the infrastructure will be necessary to know all these parameters.

UWB systems can achieve an accuracy in the order of 10 to 50 centimeters in the best cases. That makes possible to have bounded errors when estimating the position. But there are some disadvantages in these systems. First, the low dynamic range of the solution. Besides, UWB system are sensitive to external radio disturbances which can provoke wrong measurements. There is even the possibility of being in a shadow area, where less than three slaves are visible, and then the positioning is impossible to be done due to the lack of information.

All of these disadvantages can be solved by the fusion of the range information given by the UWB system with attitude information acquired by an Inertial Measurement Unit (IMU) installed on-board of the master. Furthermore, an accuracy improvement can be obtained, so that a better performance of the positioning is achieved.

In the configuration considered in this project, the master's on-board IMU consists of an Inertial Navigation System (INS), which gives measurements about acceleration and angular rate, and a Dead-Reckoning (DR) system which provides speed and direction measurements.

The advantages of using attitude information are the increase of the dynamic range due to the high sampling rate of the inertial sensors and the elimination of shadow areas by propagating the navigation equations and obtaining a tracking of the position. But not only the position estimate is obtained, also the other navigational states can be estimated. Unfortunately, the propagation of the navigation equations contains integrations, so the error increases with time, leading to a situation of unbounded error growth.

By means of using information fusion it is possible to obtain more information than is present in any individual positioning system by integrating information from the different sources. Furthermore, the fused system has the advantages of the UWB system, that is bounded errors, and the advantages of the attitude information, which are the high dynamic range, information of more navigational states (not only position) and a self-contained estimation of all the states during short periods of time, which allows the system to track the position in shadow areas.

As ranging and inertial navigation have a nonlinear dependence on the navigational

states, the optimal information fusion method is difficult to derive in the general case.

Method and purpose of the thesis

The aim of this project will be the obtainment of good and accurate estimates of the master position and, if possible, other navigational states such as velocity or orientation. The methods will be based on two different classes of information fusion algorithms, which will be developed and evaluated both on simulated data and on datasets obtained from experimental tests. The first one uses the Extended Kalman Filter (EKF) which linearizes the model equations. The second one is performed with a Particle Filter (PF) based on a Monte Carlo method.

At the end, if results are good, indoor positioning applications could take advantage of the strong performance of these methods, as other positioning systems has done before using techniques as GPS aided INS or submarine positioning with a GPS/INS/SONAR. Other examples similar to the one treated in this project can be seen in [1] [2] or [3].

Thesis outline

The Thesis report will be divided in six chapters, apart from this introduction. In Chapter 2, the constituent parts of the system architecture are described, together with their advantages and shortcomings, and finally a global vision of the fusion system is given. In Chapter 3, a general introduction to the information fusion algorithms is done. In Chapter 4, the specific algorithms developed in this project are presented and described. In Chapter 5, the results of testing the algorithms in Matlab simulations will be shown and discussed. Chapter 6 is dedicated to the results obtained during the testing of the algorithms with real data taken from a mobile-robot. Finally, in Chapter 7, the conclusions about the work done and the results obtained will be summed up.

Chapter 2

System description

The complete configuration of the positioning system is described in this chapter: the UWB ranging system, the INS and the DR systems, the navigation equations, and also an explanatory section about the reference frames and rotation matrices between these frames, are presented. First of all, the notation used in this report is summarized.

Notation

In Table 2.1 the notational conventions used in this report are described. Furthermore, when the variables appear in the report, their meaning is explained.

x	non-bold face variables denote scalars
$\boldsymbol{x}$	small boldface letters denote vectors
$\boldsymbol{A}$	capital boldface letters denote matrices
$\hat{\boldsymbol{x}}$	denotes the estimated value of $\boldsymbol{x}$, or the a posteriori estimate of $\boldsymbol{x}$
$\hat{\boldsymbol{x}}^-$	denotes the a priori estimate of $\boldsymbol{x}$
$\tilde{\boldsymbol{x}}$	denotes the measured value of $\boldsymbol{x}$
$\delta\boldsymbol{x}$	denotes the error $\boldsymbol{x} - \hat{\boldsymbol{x}}$
$\boldsymbol{x}_k$	denotes the value of $\boldsymbol{x}$ at time step k
$\boldsymbol{x}^a$	denotes vector $\boldsymbol{x}$ represented with respect to frame a
$\boldsymbol{T}_a^b$	denotes the transformation/rotation matrix from reference frames a to b

Table 2.1: Notational conventions

As the algorithms are implemented in discrete-time, the estimates will evolve using a discrete-time increment T. When denoting a variable in a certain time step a subscript k will be used.

The transformation/rotation matrices used for changing vectors from one reference

frame to another are denoted as $\boldsymbol{T}_a^b$. When the rotation matrix is calculated using estimated values of angles, it will be written as $\hat{\boldsymbol{T}}_a^b$. As explained in Section 2.2.3, only two reference frames are needed in this project. The symbol l represents the local or navigational frame, and b denotes the body frame.

The numerical values of the navigational states and the measurements will be always given in units from the International System of Units (SI). Only the accuracy values obtained in Chapter 5 and 6 can be written in cm (centimeters), and some references to times can be given in ms (milliseconds).

2.1 Ultra-Wideband (UWB) System

Pulse-based Ultra-Wideband is an interesting technology for indoor radio positioning applications. Its fine time resolution, resilience to multipath propagation and low power consumption are the most important features.

An UWB system is defined by the Federal Communications Commission Regulation [4] as any intentional radiator having a fractional bandwidth greater than 20% or an absolute bandwith greater than 500 MHz. The way to obtain such a wide spectral content is by generating pulses with short transition times, less than 1 ns. It is this low transition time which provides a good resolution when a pulse is auto-correlated, as the resolution is proportional to the inverse of the bandwidth. So a pulse with a bandwidth of 500 MHz allows a resolution of 30 cm theoretically. The maximum operating range of a UWB system is in the order of 10 to 100 m.

The system considered in this project is composed of several fixed nodes of known position, called slaves, and a mobile node, the master, the one that needs to estimate its position. Concretely four slaves are used.

The principle of operation of the UWB ranging system will be briefly explained below, as this project is only intended to simulate an indoor positioning application and not to develop a real system. Further information of a real UWB system can be seen in [1], and also for a real commercial system, which belongs to Ubisense in [5].

The master has a transceiver which is able of measuring the round-trip-time (RTT) of UWB pulses. The slaves send an UWB pulse in response to a another UWB pulse after a fixed and known delay.

First of all, the slaves are in a waiting mode. Then the master sends a sequence which identifies one of the slaves (addressing). That slave changes into a response mode, while the others become disabled for a period of time. Now the master sends a UWB pulse. When the chosen slave receives the pulse, after a fixed delay time, it emits a UWB pulse. Once the master has received the response, it is able to measure the RTT. This procedure can be repeated several times, and the master calculates the mean of the measurements. With this information the distance between the master

and the slave can be calculated easily.

Once the master has finished with the slave, all the slaves change to the waiting mode again (the disabling time of the non-chosen slaves is the same as the communication time for obtaining a RTT measurement). Then the master repeats the procedure, and takes the four ranging measurements. This sequence is repeated every 100 ms in this project.

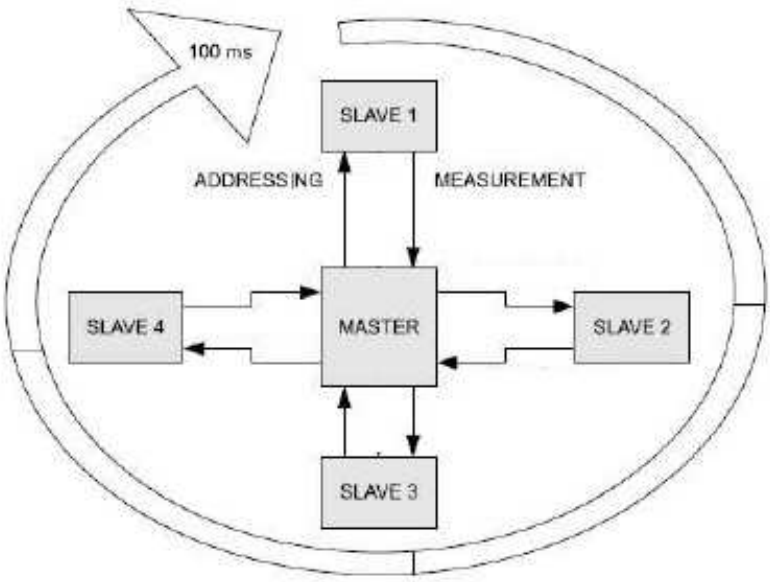

Figure 2.1: Operation of the system

This is not the only possible configuration for a UWB system. For example, it is possible to realize a configuration in which there is a slave which coordinates the activities of the other slaves and calculates the position of the master.

The RTT measurements depend on the distance, the delays in the circuitry and the intentional delays, the detector rise-time, etc. But tolerances in components or the length of the cabling can make the latency different in each slave. So a calibration would be necessary after installing the infrastructure. However, random errors will persist in the measurements. These errors can be treated as white Gaussian noise:

$$t_{RTT} = \frac{2 * D_{m-s}}{c} + t_{delaymaster} + t_{delayslave} + n_{RTT} \quad [s]$$

In this relation, the D_{m-s} is the distance between the master and the slave, and $t_{delaymaster}$ and $t_{delayslave}$ are the hardware delays in the master and in the slave, respectively. The n_{RTT} term represents the white Gaussian noise of the measurements. The c is the speed of light.

In [1], the standard deviation of the error is calculated empirically, showing a linear dependence with the distance:

$$\sigma_{RTT} = \sigma_0 + k * range = 0.25 + 0.01 * range \quad [ns]$$

A change of units is necessary so as to obtain the standard deviation of the range measurements:

$$\sigma_{range} = (0.25 + 0.01 * range) * 10^{-9} * \frac{c}{2} \quad [m]$$

This value will be used in the simulations for characterizing the standard deviation of the noise in the ranging measurements.

2.2 Inertial Navigation System (INS)

The inertial navigation System can be divided into a sensor part, the Inertial Navigation Unit (IMU), and a computational part which is responsible to estimate the navigational states by propagating the navigation/mechanization equations, as shown in Figure 2.2.

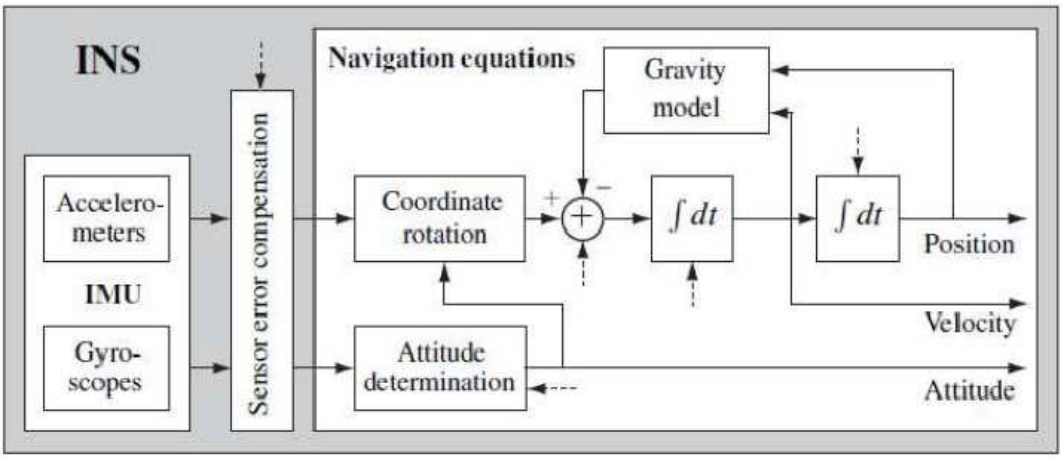

Figure 2.2: Diagram of a strap-down INS. Calibration data can be inserted in the points indicated by the dashed arrows [1].

2.2.1 Inertial Measurement Unit (IMU)

The IMU consists of an accelerometer and a gyroscope in a strap-down configuration. In a strap-down system the accelerometer and the gyroscope are mounted in body

coordinates, so they always give measurements related to the IMU axes and this is called the body frame [6], [7]. Then a matrix rotation has to be used to rotate the measurements to the local or navigational frame where the application works. This system reduces the size, cost, power consumption and complexity in comparison with a gimballed system, in which the body frame is rotated mechanically to keep the alignment with the navigational frame.

The accelerometer is able to measure the acceleration in the three axes of the IMU body frame, and the gyroscope measures the angular rate of rotation in the three axes as well. However, this indoor application is only designed to work in 2D, in the X and Y components of the navigational frame.

IMU measurements are seriously affected by several sources of error. Sensors' quality is one of the most important factors. The lower the quality, the more errors they have. Concretely they can suffer from bias and drifts. It causes the measures to have small offsets which cause very bad effects in the accuracy of the system. So, for a proper reading of measurements, a calibration of the sensors is needed. Thus, the offset can be removed. But sensors can be sensitive to temperature, changing the values of the bias and drifts, and that has to be taken into account.

Another important source of error in a strap-down system are the vibrations. If the floor is not completely flat, measurements will be continuously affected by outliers due to the high peaks in the acceleration that bumps can cause. Moreover, a rough continuous surface adds a noise to the measurements.

In conclusion, for having proper inertial measurements, a previous calibration is needed and some data processing like a low-pass filtering or a control of outliers. In this way the offsets and the noise caused by vibrations can be attenuated.

The IMU that is supposed to be used in the simulations is a 3DM-GX2 of Microstrain® [8], a picture of which is shown in Figure 2.3. A reason for that is that the algorithms developed in this project will be tested subsequently with real data, and the inertial measurements were extracted using this device.

The errors added to the measurements in the simulations are presented in the following table, based on the 3DM-GX2 specifications:

Error	**Accelerometer**	**Gyroscope**
Bias	$\pm 0.1 \frac{m}{s^2}$	$\pm 0.0035 \frac{rad}{s}$
Nonlinearity	0.2 %	0.2 %

Table 2.2: Errors in measurements based on 3DM-GX2 specifications

Figure 2.3: Image of the Microstrain Gyro Enhanced Orientation Sensor 3DM-GX2

2.2.2 Navigation Equations

So as to estimate the position and other navigation states such as velocity, orientation or acceleration, navigation equations must be used. They describe the propagation of the navigational states in time, taking into account the values of the previous states and IMU measurements.

Navigation equations are given as differential equations, so integration is necessary for their propagation. However, if measurements are assumed to be constant over a sampling period, equations can be discretized.

An INS is able to estimate, as a stand-alone system, the master's navigation states by propagating these navigation equations. But integration implies that errors grow with time, so the error will be unbounded.

In [3] and [7] a derivation of the navigation equations can be seen, though a simpler version will be considered here, as shown in Figure 2.4.

In the equations, master's navigation states are designated as $\boldsymbol{x}$ for the position, $\boldsymbol{v}$ for the master's velocity and $\boldsymbol{\theta}$ is the master's orientation. Measurements are indicated with a $\sim$. The acceleration is written as $\tilde{\boldsymbol{a}}$ and the angular rate as $\tilde{\boldsymbol{\omega}}$.

The superscript l indicates that the value is in the local frame. The subscript k reports that the value is from the current time step, and the subscript $k+1$ is for values of the next time step. The time difference between these two steps is dt_k.

Finally, $\mathbf{I}$ is the 2x2 diagonal matrix and $\mathbf{0}$ is a 2x2 matrix of zeros.

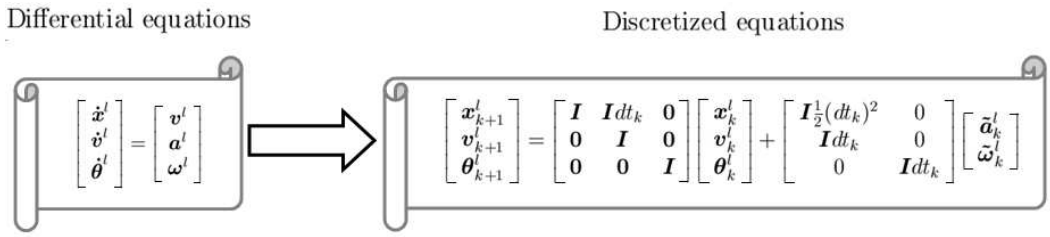

Figure 2.4: Navigation equations in differential and discretized mode

2.2.3 Frames and Rotations

The local frame of reference (or navigation frame) of this application is the surface in which the indoor positioning has to be performed. So it can be considered a 2D application as the position will be given in X and Y components. But the model can easily be adapted for 3D applications, only by adding the Z component corresponding to the perpendicular axes to the surface pointing up.

Inertial measurements are given in the instrument frame of reference, as sensors resolve their measurements along the sensitive axes of the instruments, the 3DM-GX2 in this case. The IMU is installed on a platform on the master. That is the platform frame and, for simplicity, a perfect alignment with the instrument frame will be supposed.

The body frame is the master's frame of reference. As the platform is attached to the master, the alignment between the body and the platform frame has to be considered. Again for simplicity, this alignment will be supposed perfect. So measurements will be given in the body frame.

Since the only misalignment can be between the master's body frame of reference and the local frame in which the applications works, a rotation must be used so that the measurements in body coordinates can be applied to resolve the navigation states in the local system of coordinates. Further information about reference frames can be read in [6].

The transformation is a simplification of the general 3D rotation matrix:

$$\boldsymbol{R}_l^b = \begin{vmatrix} cos\theta cos\psi & -cos\theta sin\psi + sin\phi sin\theta cos\psi & sin\phi sin\psi + cos\phi sin\theta cos\psi \\ cos\theta sin\psi & cos\theta cos\psi + sin\phi sin\theta sin\psi & -sin\phi cos\psi + cos\phi sin\theta sin\psi \\ -sin\theta & sin\phi cos\theta & cos\phi cos\theta \end{vmatrix}$$

This transformation involves the roll ϕ, pitch θ and yaw ψ angles, corresponding to counterclockwise rotations in the X, Y and Z axes, respectively. As the application works in 2D, in X-Y plane, roll and pitch angles are supposed to be zero, and the orientation of the sensor with respect to the local frame is indicated by the yaw angle

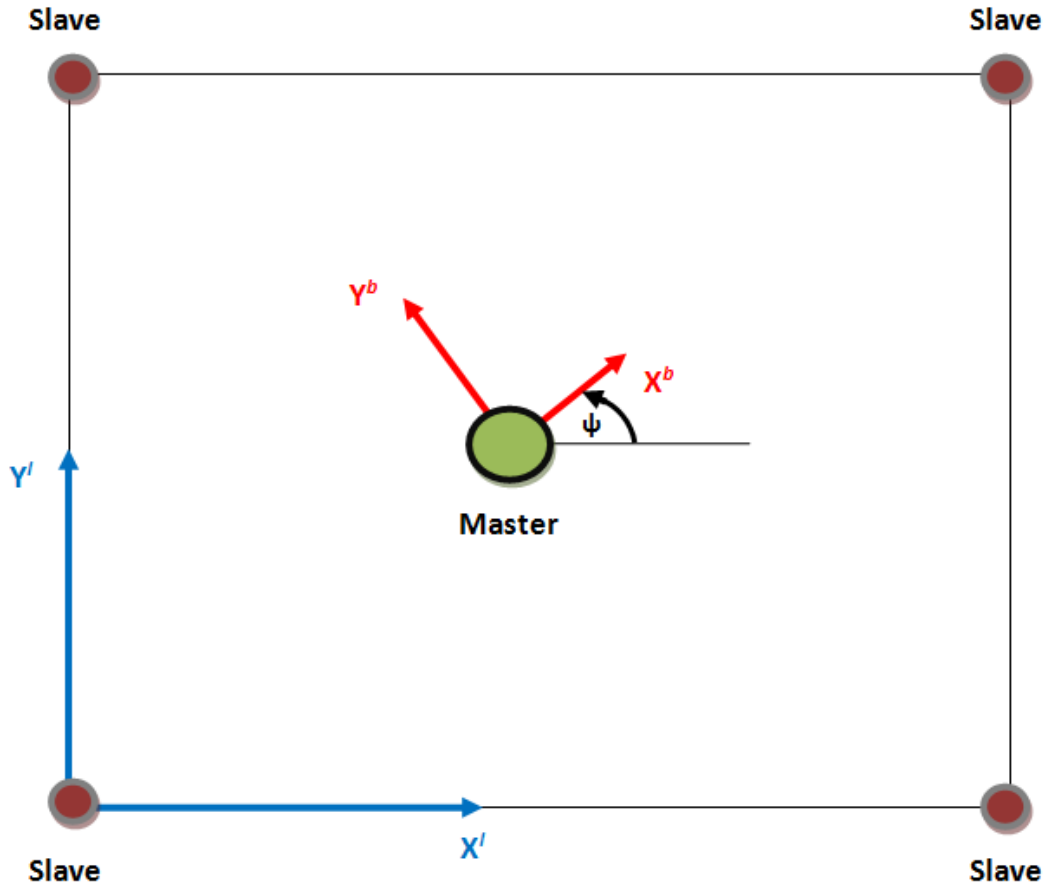

Figure 2.5: System's frames scheme. Local frame is in blue and body frame in red.

(obtainable using the navigation equations and the angular rate in the Z component). With these considerations, the rotation matrix is simplified as follows:

$$\boldsymbol{R}_l^b = \boldsymbol{T}_l^b = \begin{vmatrix} cos\psi & -sin\psi & 0 \\ sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

$\boldsymbol{T}_l^b$ is the rotation matrix that goes from local frame (the origin frame is indicated by the subscript, l of local in this case) to body frame (the final frame is indicated by the superscript, b of body in this case). Nevertheless, the desired transformation is the one that goes from body to local coordinates which corresponds to the inverse of this matrix. But the rotation matrix is an orthogonal matrix, so its inverse is equal to its transpose.

$$\boldsymbol{T}_b^l = \begin{vmatrix} cos\psi & sin\psi & 0 \\ -sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

$\boldsymbol{T}_b^l$ will be the matrix used for changing coordinates.

2.3 Dead-Reckoning System

A Dead-Reckoning system is one that is able to calculate its own current position by using its previous position and measurements of speed and direction. The bad point is that it is subject to cumulative errors like the INS.

There are some different ways to obtain the measurements. The direction can be calculated with a compass or a magnetometer for example. However, in this project a encoder-based DR system will be supposed [6]. The reason is that the real speed and orientation data used later is captured with this system, since the UWB master node was placed on a mobile robot.

The operation is simple. By knowing wheels' angle of rotation ϕ_R, their radius R_R and the sampling time T (see Figure 2.6), the speed v can easily be calculated, assuming no wheel slip, as follows:

$$v = \frac{S_R(t)}{T} = \frac{R_R \phi_R(T)}{T}$$

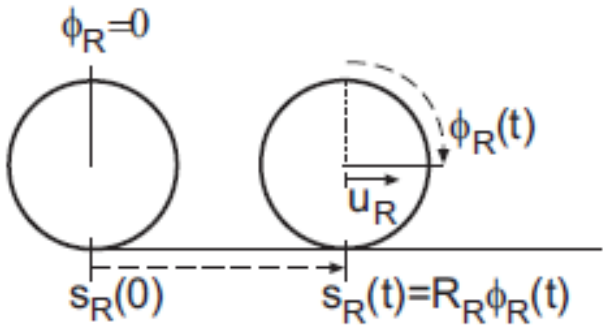

Figure 2.6: Encoder's principle of operation. R_R is wheel's radius, $\theta_R(t)$ is the angle rotated and $S_R(t)$ is the distance traveled in a sampling time. Speed is the result of dividing the distance by the sampling time.

The variation of orientation is calculated from the differences of the speeds measured by the two encoders on the two wheels, and is added to the previous orientation value,

obtaining the current orientation:

$$\theta_k = \theta_{k-1} + \Delta\theta$$

In this way the measurements obtained are the module of speed v and the direction of the master θ. For calculating the two components of the velocity in the local frame, the following expression is needed:

$$\boldsymbol{v}^l = \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \end{bmatrix}$$

DR system has to propagate the following navigation equation to estimate the position as a stand-alone system:

$$\boldsymbol{x}^l_{k+1} = \boldsymbol{x}^l_k + dt_k\tilde{\boldsymbol{v}}^l$$

Last of all, errors in simulations will be characterized as Gaussian white noise, with a standard deviation $\sigma_v = 0.05\frac{m}{s}$ for the speed and a standard deviation $\sigma_\phi = \frac{\pi}{8}$ rad for the orientation [2].

2.4 Fusion

As noted, the UWB system takes RTT measurements of the four slaves every 100 ms. This fact implies having a low sampling rate of 10 Hz and a poor dynamic range. Another disadvantage is that the UWB system is not able to provide attitude information, only gives the position. On the other hand, the errors obtained are bounded.

The INS and the DR are able to provide self-contained estimations of master's navigational states, and their high sampling rate make them capable of handling high dynamics. But errors grow with time due to navigation equations propagation, so they are unbounded.

The fusion of these two systems compensates the shortcomings of both and yields better performance than possible from any individual system [1].

As the inertial navigation and the ranging have a nonlinear dependence on the navigation states, the optimal information fusion is difficult to obtain in the general case. However, there are several algorithms which approximate the optimal solution.

The two algorithms implemented in this project are the Extended Kalman Filter (EKF) and the Particle Filter (PF). In the following sections, first of all, an intro-

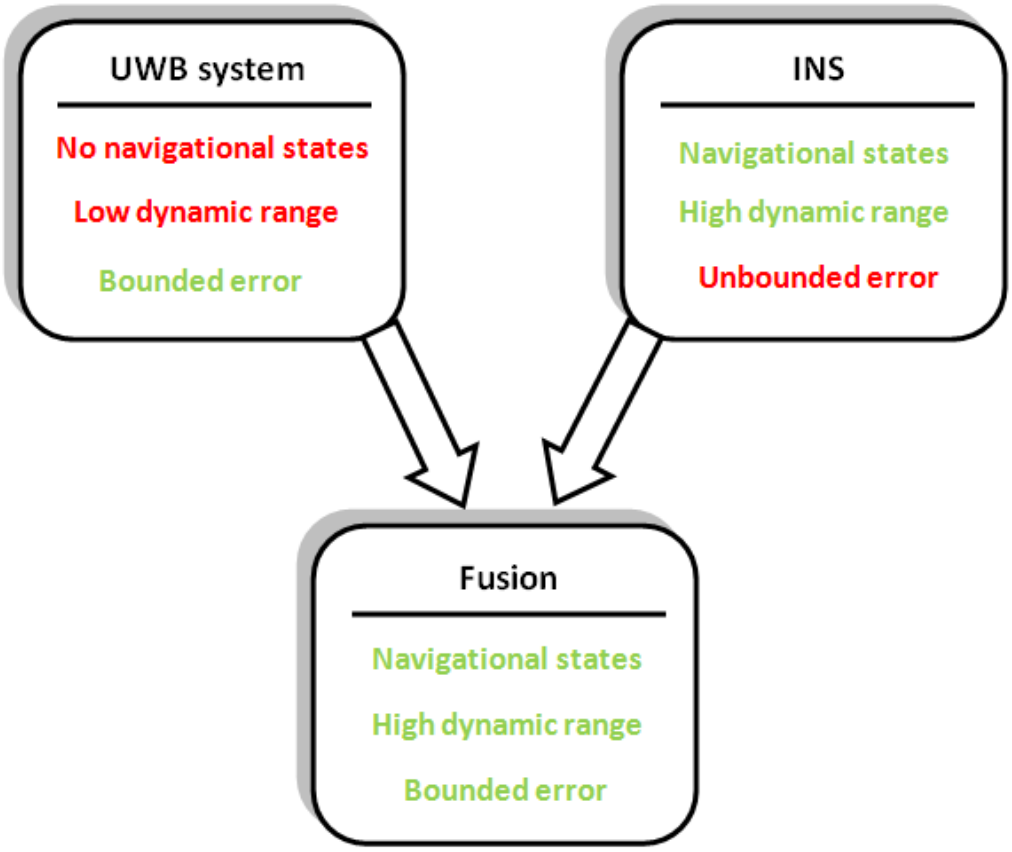

Figure 2.7: Summary of the properties in the three systems.

duction to the use of these algorithms is presented, and then the specific developed algorithms are explained.

Chapter 3

Introduction to Kalman Filter and Particle Filter

In this chapter the operation of the Kalman Filter, Extended Kalman Filter and Particle Filter is explained. Also the equations used in their algorithms are shown, and the consequences of tuning some of their parameters.

3.1 Kalman Filter

The Kalman filter is a set of mathematical equations. Its function is to estimate the state of a process, and it can do this in an efficient computational recursive way, in which the mean of the squared error is minimized. It supports estimations of past, present, and even future states, though the nature of the modeled system is unknown [6],[9],[10].

3.1.1 Process and estimations

The Kalman Filter treats the problem of estimating the state $\boldsymbol{x} \in R^n$ of a discrete-time process that can be characterized by the linear difference equation:

$$\boldsymbol{x}_k = \boldsymbol{A}\boldsymbol{x}_{k-1} + \boldsymbol{B}\boldsymbol{u}_{k-1} + \boldsymbol{w}_{k-1} \tag{3.1}$$

The estimation also needs a measurement $\boldsymbol{z} \in R^m$, in the following way:

$$\boldsymbol{z}_k = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{v}_k \tag{3.2}$$

The random variable $\boldsymbol{w}_k$ represents the process noise, that is the error between the

characterization of the process and the real process. The other random variable, $\boldsymbol{v}_k$, represents the measurement noise, that can be considered as the error between the estimation of the measurement $\boldsymbol{H}\boldsymbol{x}_k$ and the real measurement. Both variables are assumed to be independent of each other, Gaussian and white:

$$p(w) \sim N(\mathbf{0}, \boldsymbol{Q})$$

$$p(v) \sim N(\mathbf{0}, \boldsymbol{R})$$

In this general introduction, the process noise covariance $\boldsymbol{Q}$ and the measurement noise covariance $\boldsymbol{R}$ are assumed to be constant over the iterations. Nevertheless, they might change in every time step (in every iteration).

The $n \times n$ matrix $\boldsymbol{A}$ is the state transition model which relates the state at the previous time step $k-1$ to the state at the current step k. The $n \times l$ matrix $\boldsymbol{B}$ is the control-input model which relates the optional control input $\boldsymbol{u} \in R^l$ to the state. The $m \times n$ matrix $\boldsymbol{H}$ is the observation model relating the state to the measurement $\boldsymbol{z}_k$.

Now, two definitions are introduced. The first one, $\hat{\boldsymbol{x}}_k^- \in R^n$, is the a priori state estimate at step k given knowledge of the process prior to step k. The second one is $\hat{\boldsymbol{x}}_k \in R^n$ and it refers to the a posteriori state estimate at step k given measurement z_k.

The a priori and a posteriori estimate errors are:

$$\boldsymbol{e}_k^- \equiv \boldsymbol{x}_k - \hat{\boldsymbol{x}}_k^-$$

$$\boldsymbol{e}_k \equiv \boldsymbol{x}_k - \hat{\boldsymbol{x}}_k$$

With these errors, the a priori estimate error covariance and the a posteriori estimate error covariance can be calculated respectively as:

$$\boldsymbol{P}_k^- = E[\boldsymbol{e}_k^- \boldsymbol{e}_k^{-T}]$$

$$\boldsymbol{P}_k = E[\boldsymbol{e}_k \boldsymbol{e}_k^T]$$

When deriving the equations for the Kalman Filter, the first step is finding an equations that computes an a posteriori estimate $\hat{\boldsymbol{x}}_k$ as a linear combination of an a priori estimate $\hat{\boldsymbol{x}}_k^-$ and a weighted difference between an actual measurement $\boldsymbol{z}_k$ and a measurement prediction $\boldsymbol{H}\hat{\boldsymbol{x}}_k^-$ calculated using the a priori state estimate. In [11] the justification can be found as well as developments presented later in this introduction.

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}(\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_k^-)$$

The difference $(\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_k^-)$ is called the innovation and measures the discrepancy between the predicted measurement and the observed measurement. If the innovation is zero, it means that they are in complete agreement, so the predicted measurement is perfect due to the a priori state estimate is also perfect.

The $n \times m$ matrix $\boldsymbol{K}$ is the Kalman gain that minimizes the a posteriori error covariance $\boldsymbol{P}_k$. Details of the calculation are given in [11]. The general expression of K in the current time-step k is:

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}^T (\boldsymbol{H}\boldsymbol{P}_k^- \boldsymbol{H}^T + \boldsymbol{R})^{-1}$$

Looking at this expression some information about Kalman Filter behavior can be extracted. When the a priori error covariance $\boldsymbol{P}_k^-$ approaches zero, the Kalman gain is also tending to zero, so the innovation is not trusted. Another way to see it is that if $\boldsymbol{P}_k^-$ tends to zero is because the a priori state estimate is very similar to the real value of the state, and then it is better to trust the estimate rather than the measurement.

In the other hand, if the measurement noise covariance $\boldsymbol{R}$ approaches zero, it means that measurements have little error, so it should be better to trust them more than the a priori state estimate. This also happens when $\boldsymbol{P}_k^-$ is very large, meaning that the measurements have small error compared to the prediction. In this case the Kalman gain $\boldsymbol{K}_k$ tends to $\boldsymbol{H}^{-1}$.

3.1.2 Algorithm

The Kalman Filter estimates a process using a feedback control. First, the filter estimates the process state at certain time and then obtains feedback in the form of noisy measurements. Taking this into account, the equations can be divided into two groups: the time update equations and the measurement update equations. The first ones project forward in time the current state and error covariance estimates in order to obtain the a priori estimates for the next time step. After that, the second group of equations introduce the feedback, concretely they use a new observed measurement for correcting the a priori estimate and obtain the a posteriori estimate.

This algorithm can also be thought as a predictor-corrector algorithm, in which the time update equations are the predictor equations, while the measurement update equations are the corrector equations.

The algorithm and the equations are shown in Figure 3.1.

The first equation in the time update equations is the one responsible for projecting the state ahead using the previous estimate state and optionally an input. The second

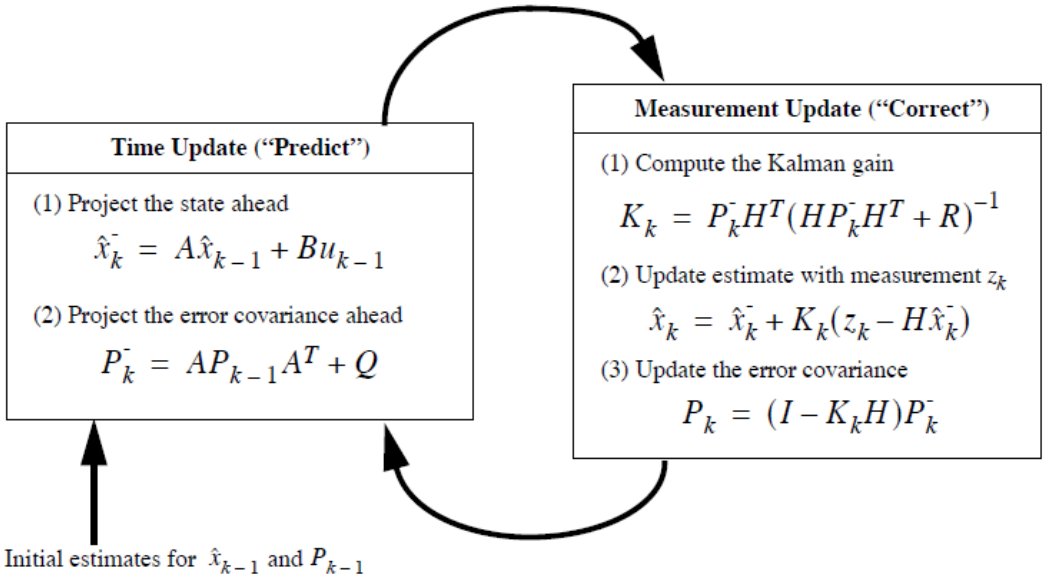

Figure 3.1: Operation of the Kalman Filter showing prediction and correction phases [9].

equation is the one that projects the error covariance ahead.

In the measurement update phase, the first thing to do is to calculate the Kalman gain $\boldsymbol{K}_k$. Then it is necessary to measure the process to obtain $\boldsymbol{z}_k$. With the measurement $\boldsymbol{z}_k$ and the measurement estimation $\boldsymbol{H}\hat{\boldsymbol{x}}_k^-$, the innovation is obtained. Now using the innovation and the Kalman gain it is possible to generate an a posteriori state estimate, as shown in the second equation of the measurement update. The third and last equation generates an a posteriori error covariance estimate.

After the time and measurement update (or prediction and correction phase) the procedure is repeated, projecting the previous a posteriori estimates to obtain the new a priori estimates.

3.1.3 Filter parameters and tuning

Initial estimates for the state $\hat{\boldsymbol{x}}_0$ and for the estimate error covariance $\boldsymbol{P}_0$ have to be provided for the filter's initialization. If the initial state estimate might be incorrect, a higher initial estimate error covariance should be used, as this indicates that the initial state estimate is wrong and is not trusted. Besides, a high $\boldsymbol{P}_0$ will permit a faster convergence of the filter. In contrast, the estimations are not going to be smooth and the error in them will be higher in the initial phase of the trajectory. So there is a trade-off in choosing the initial estimate error covariance.

Another parameter that is related to the convergence and the smoothness of the estimations is the process noise covariance $\boldsymbol{Q}$. It has to be said that is difficult to determine its value since direct observation of the process we are estimating is complex (usually impossible). So if the knowledge of the process model is low, a high uncertainty has to be injected in the filter via a high value of $\boldsymbol{Q}$, and in this case measurements are going to be more reliable than the model. In contrast, if the process model is well known, a low $\boldsymbol{Q}$ will permit a robustness in the filter to outliers in measurements, but a slower convergence due to smoother changes in estimations.

In conclusion, the values of $\boldsymbol{P}_0$ and $\boldsymbol{Q}$ have to be chosen taking into account the initial knowledge of the state and the process model, and they will affect the speed of convergence, the smoothness in estimation changes, and the robustness against wrong measurements.

The measurement noise covariance $\boldsymbol{R}$ can be measured before operating the filter. Since it is possible to measure the process anyway, some-off line sample measurement can be taken in order to determine the variance of the measurement noise. It's obvious that if $\boldsymbol{R}$ values are high, measurements will not be trusted, and the filter will follow the process model. In other words, predictions contribution in estimations will be more important than corrections contribution.

This matrix $\boldsymbol{R}$ might be constant or not. In case that the variance of the measurement noise changes while filter's operation, it can be replaced by the new covariance matrix. Even if the $\boldsymbol{R}$ changes in every iteration it can always be replaced.

3.2 Extended Kalman Filter

3.2.1 Process and estimation

As explained before, the Kalman Filter tries to estimate the state $\boldsymbol{x} \in R^n$ of a discrete-time process governed by a linear stochastic difference equation. But a problem appears when the process and/or the measurement relationship to the process is non-linear. This situation happens very frequently, and the solution is a Kalman Filter that linearizes about an estimate of the current state: the Extended Kalman Filter (EKF).

The procedure is similar to a second-order Taylor series expansion. The estimation is linearized around the current estimate using the partial derivatives of the process and measurement functions.

Some previous definitions about the Kalman Filter now are changed. Now a process with a state vector $\boldsymbol{x} \in R^n$ is assumed, but now the process is modeled by a non-linear stochastic difference equation:

$$\boldsymbol{x}_k = f(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{w}_{k-1})$$

And the measurement $\boldsymbol{z} \in R^m$, has also a non-linear relation with the state in the following way:

$$\boldsymbol{z}_k = h(\boldsymbol{x}_k, \boldsymbol{v}_k)$$

The process and measurement noise remain as $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ respectively, with the same probability function. The non-linear function f relates the state at previous time step $k-1$, an optional input $\boldsymbol{u}_{k-1}$ and the process noise $\boldsymbol{w}_k$, to the state at current time step k. On the other hand, the non-linear function h relates the actual state $\boldsymbol{x}_k$ to the measurement $\boldsymbol{z}_k$.

The state vector and the measurement have to be approximated without using the noises, because it is not possible to know their values:

$$\tilde{\boldsymbol{x}}_k = f(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_{k-1}, 0) \tag{3.3}$$

$$\tilde{\boldsymbol{z}}_k = h(\tilde{\boldsymbol{x}}_k, 0) \tag{3.4}$$

The $\sim$ sign indicates an approximation, and $\hat{\boldsymbol{x}}_{k-1}$ is the a posteriori estimate of the previous time step.

It is important to note that after the non-linear transformations the various random variables do not conserve the normal distribution. The Extended Kalman Filter is an state estimator that only approximates the optimality of Bayes' rule by linearization (see section 3.3).

3.2.2 Derivation of equations and algorithm

To estimate a process with non-linear relationships, it is necessary to write new governing equations that linearize an estimate about (3.3) and (3.4):

$$\boldsymbol{x}_k \simeq \tilde{\boldsymbol{x}}_k + \boldsymbol{A}(\boldsymbol{x}_{k-1} - \hat{\boldsymbol{x}}_{k-1}) + \boldsymbol{W}\boldsymbol{w}_{k-1}$$

$$\boldsymbol{z}_k \simeq \tilde{\boldsymbol{z}}_k + \boldsymbol{H}(\boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k) + \boldsymbol{V}\boldsymbol{v}_k$$

Here, $\boldsymbol{x}_k$ is the actual real state and $\boldsymbol{z}_k$ is the observed measurement vectors, $\tilde{\boldsymbol{x}}_k$ and $\tilde{\boldsymbol{z}}_k$ are the approximate state and measurement vectors from (3.3) and (3.4), $\hat{\boldsymbol{x}}_k$ is

an a posteriori estimate of the state at step k, and $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ are the process and measurement noise respectively.

$\boldsymbol{A}$ is the Jacobian matrix of partial derivatives of f with respect to $\boldsymbol{x}$, where each element is like follows:

$$\boldsymbol{A}_{[i,j]} = \frac{\delta f_{[i]}}{\delta x_{[j]}}(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_{k-1}, 0)$$

$\boldsymbol{W}$ is the Jacobian matrix of partial derivatives of f with respect to $\boldsymbol{w}$:

$$\boldsymbol{W}_{[i,j]} = \frac{\delta f_{[i]}}{\delta w_{[j]}}(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_{k-1}, 0)$$

$\boldsymbol{H}$ is the Jacobian matrix of partial derivatives of h with respect to $\boldsymbol{x}$:

$$\boldsymbol{H}_{[i,j]} = \frac{\delta h_{[i]}}{\delta x_{[j]}}(\tilde{\boldsymbol{x}}_k, 0)$$

$\boldsymbol{V}$ is the Jacobian matrix of partial derivatives of h with respect to $\boldsymbol{v}$:

$$\boldsymbol{V}_{[i,j]} = \frac{\delta h_{[i]}}{\delta v_{[j]}}(\tilde{\boldsymbol{x}}_k, 0)$$

Note that the Jacobians $\boldsymbol{A}$, $\boldsymbol{W}$, $\boldsymbol{H}$ and $\boldsymbol{V}$ change at each time step, but for simplicity the subscript k is not used.

Now a new notation has to be introduced. The prediction error and the measurement residual are, respectively:

$$\tilde{\boldsymbol{e}}_{x_k} \triangleq \boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k$$

$$\tilde{\boldsymbol{e}}_{z_k} \triangleq \boldsymbol{z}_k - \tilde{\boldsymbol{z}}_k$$

From these two equivalences, two governing equations for an error process can be written as:

$$\tilde{\boldsymbol{e}}_{x_k} \simeq \boldsymbol{A}(\boldsymbol{x}_{k-1} - \hat{\boldsymbol{x}}_{k-1}) + \boldsymbol{\epsilon}_k \tag{3.5}$$

$$\tilde{\boldsymbol{e}}_{z_k} \simeq \boldsymbol{H}\tilde{\boldsymbol{e}}_{x_k} + \boldsymbol{\eta}_k \tag{3.6}$$

Here $\boldsymbol{\epsilon}_k$ and $\boldsymbol{\eta}_k$ represent new independent random variables having zero mean and covariance matrices $\boldsymbol{WQW}^T$ and $\boldsymbol{vRV}^T$, being Q and R the process and measurement noise covariances from the general Kalman Filter. The probability distributions are:

$$p(\tilde{\boldsymbol{e}}_{x_k}) \sim N(\mathbf{0}, E[\tilde{\boldsymbol{e}}_{x_k}\tilde{\boldsymbol{e}}_{x_k}^T])$$
$$p(\boldsymbol{\epsilon}_k) \sim N(\mathbf{0}, \boldsymbol{WQ}_k\boldsymbol{W}^T)$$
$$p(\boldsymbol{\eta}_k) \sim N(\mathbf{0}, \boldsymbol{VR}_k\boldsymbol{V}^T)$$

Equations (3.5) and (3.6) are linear and similar to the difference and measurement equations (3.1) and (3.2) from the discrete Kalman Filter. This is the reason to apply the same procedure used in the discrete Kalman Filter using equations (3.5) and (3.6).

Now, $\hat{\boldsymbol{e}}_k$ will be an estimate of $\tilde{\boldsymbol{e}}_{x_k}$, calculated using the actual measurement residual $\tilde{\boldsymbol{e}}_{z_k}$.

$$\hat{\boldsymbol{e}}_k = \boldsymbol{K}_k\tilde{\boldsymbol{e}}_{z_k} \tag{3.7}$$

The a posteriori state estimation for the original non-linear process is:

$$\hat{\boldsymbol{x}}_k = \tilde{\boldsymbol{x}}_k + \hat{\boldsymbol{e}}_k \tag{3.8}$$

And substituting (3.7) and (3.4) in (3.8), the correction/measurement update equation for calculating the a posteriori state estimate is obtained

$$\hat{\boldsymbol{x}}_k = \tilde{\boldsymbol{x}}_k + \boldsymbol{K}_k\tilde{\boldsymbol{e}}_{z_k} = \tilde{\boldsymbol{x}}_k + \boldsymbol{K}_k(\boldsymbol{z}_k - \tilde{\boldsymbol{z}}_k) = \tilde{\boldsymbol{x}}_k + \boldsymbol{K}_k(\boldsymbol{z}_k - h(\tilde{\boldsymbol{x}}_k, 0)) \tag{3.9}$$

The Kalman gain $\boldsymbol{K}_k$ comes from the same equation used in the discrete Kalman Filter, substituting the new measurement error covariance.

At this moment all the set of equations for the Extended Kalman Filter is obtained, and they are shown in Figure 3.2, where the operation of the algorithm is also presented.

The algorithm works in the same way than the discrete Kalman Filter algorithm. The approximation of the state $\tilde{\boldsymbol{x}}_k$ substitutes the a priori state estimate $\hat{\boldsymbol{x}}_k^-$ used in the discrete Kalman Filter, and the Jacobians $\boldsymbol{A}$, $\boldsymbol{W}$, $\boldsymbol{H}$ and $\boldsymbol{V}$ have the subscript k to remark that they must be recalculated at each time step.

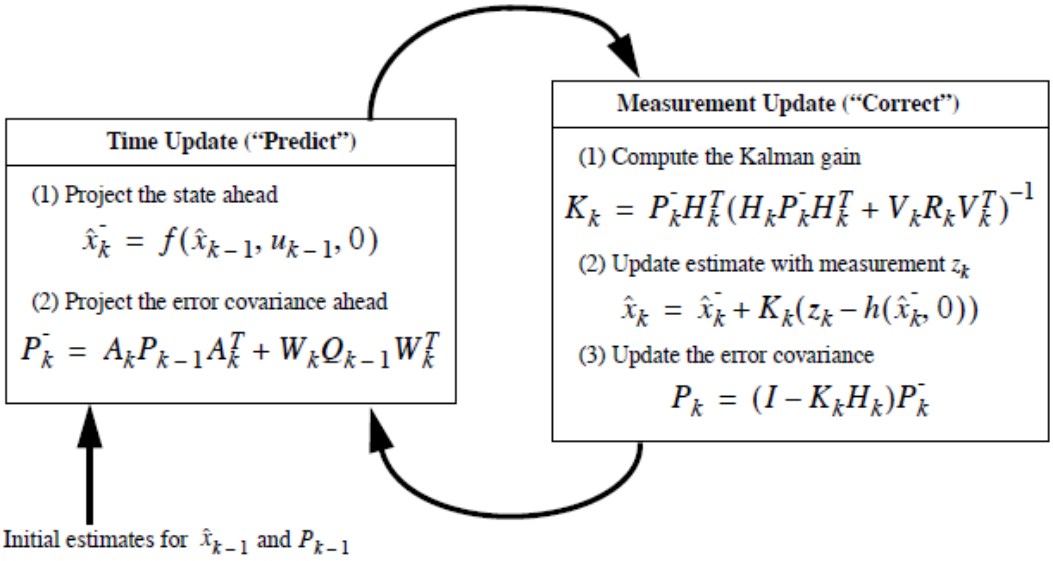

Figure 3.2: Operation of the Extended Kalman Filter showing prediction and correction phases.

3.3 Particle Filter

Another algorithm that concerns the estimation of the state $\boldsymbol{x} \in R^n$ of a discrete-time process governed by a non-linear model is the Particle Filter. Furthermore, the model might be non-Gaussian, in contrast with the Gaussian assumption of the Kalman Filter [10], [12], [13],[14].

So the general model for the process is:

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k)$$

$$\boldsymbol{z}_k = h(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{v}_k$$

The probability functions of the process noise $\boldsymbol{w}_k$ and the measurement noise $\boldsymbol{v}_k$ are arbitrary but assumed to be known.

But first of all, for better understanding of the Particle Filter, the Bayesian Filtering has to be introduced, since the Particle Filter is intended to be a numerical approximation to the Bayesian Filtering algorithm.

The Bayesian Filtering algorithm computes the posterior distribution $p(\boldsymbol{x}_k \mid \boldsymbol{Z}_k)$ of the state vector, given a set of past observations $\boldsymbol{Z}_k = \{\boldsymbol{z}_0, ..., \boldsymbol{z}_k\}$. The solution is given by the general Bayesian update recursion:

$$p(\boldsymbol{x}_k \mid \boldsymbol{Z}_k) = \frac{p(\boldsymbol{z}_k \mid \boldsymbol{x}_k)p(\boldsymbol{x}_k \mid \boldsymbol{Z}_{k-1})}{p(\boldsymbol{z}_k \mid \boldsymbol{Z}_{k-1})}$$

$$p(\boldsymbol{z}_k \mid \boldsymbol{Z}_{k-1}) = \int p(\boldsymbol{z}_k \mid \boldsymbol{x}_k)p(\boldsymbol{x}_k \mid \boldsymbol{Z}_{k-1})dx_k$$

$$p(\boldsymbol{x}_{k+1} \mid \boldsymbol{Z}_k) = \int p(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k)p(\boldsymbol{x}_k \mid \boldsymbol{Z}_k)dx_k$$

The first equation follows the Bayes' law and correspond to a measurement update. The second and the third equations come from the law of total probability and they correspond to a normalization constant and a time update, respectively.

Once the posterior distribution is obtained, a minimum mean square (MMS) estimate of the state vector $\hat{\boldsymbol{x}}_k^{MMS}$ and its covariance $\boldsymbol{P}_k^{MMS}$ can be calculated as:

$$\hat{\boldsymbol{x}}_k^{MMS} = \int \boldsymbol{x}_k p(\boldsymbol{x}_k \mid \boldsymbol{Z}_k)dx_k$$

$$\boldsymbol{P}_k^{MMS} = \int (\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k^{MMS})(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k^{MMS})^T p(\boldsymbol{x}_k \mid \boldsymbol{Z}_k)dx_k$$

In case of having a linear Gaussian model, the Kalman Filter provides the solution to this Bayesian problem. However, for non-linear or non-Gaussian models, there is in general no finite dimensional representation for the values of $(\hat{\boldsymbol{x}}_k^{MMS}, \boldsymbol{P}_k^{MMS})$. That is the reason why numerical approximations are needed, such as the one obtained with the Particle Filter.

As said, the Particle Filter provides an approximation to the posterior distribution $p(\boldsymbol{x}_k \mid \boldsymbol{Z}_k)$ of the state $\boldsymbol{x}_k$ conditioned on the set of measurements $\boldsymbol{Z}_k = \{\boldsymbol{z}_0, ..., \boldsymbol{z}_k\}$. The approximation is based on a set of N samples $\{\boldsymbol{x}^i\}_{i=1}^N$, called particles, and their associated weights $\{w^i\}_{i=1}^N$.

The recursive algorithm includes three steps. The first one is the measurement update, in which the weights are modified according to the likelihood function of the difference between the observed measurement and the predicted one. The second step is the resampling, which consists in taking N new samples of the state in substitution of the existing particles. Finally, the third step is the time update, in which a prediction of the state (the particles) in the next time step is done using the dynamic model (the proposal distribution).

Algorithm The first step in the operation of the Particle Filter is choosing a proposal distribution $q(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_{1:k}, \boldsymbol{Z}_{k+1})$, which in the general case is given by the

dynamic model. The resampling strategy has to be chosen and the number of particles N as well. The more particles used, the better approximation to the posterior distribution, but more computing time will be necessary.

In the initialization of the filter the initial particles have to be generated. Commonly, the distribution used is a uniform distribution which takes into account the possible values of the initial state. So, a set of particles $\boldsymbol{x}_0^i, i = 1, ..., N$ with a probability $p_{x_0} = w_0^i = \frac{1}{N}$ is generated.

The steps repeated in each iteration are described below, and in Figure 3.3 the operation of the Particle Filter is schematized in a flow chart.

1. *Measurement update.*

   In the measurement update phase, a prediction of the measurement has to be calculated using the particles, that can be considered as potential states.

   $$\boldsymbol{z}_k^i = h(\boldsymbol{x}_k^i)$$

   Then the observed measurement is introduced in the algorithm so as to evaluate the likelihood function of the difference between the observed and the predicted measurement. With this value, calculated for each particle, the weights are modified: the particles that lead to measurements similar to the observed one $\boldsymbol{z}_k$ will have a higher weight.

   $$w_k^i = w_{k-1}^i * p(\boldsymbol{z}_k \mid \boldsymbol{x}_k^i) = w_{k-1}^i * p_{v_k}(\boldsymbol{z}_k - h(\boldsymbol{x}_k^i))$$

   The known probability function of the measurement noise p_{v_k} is the one used for evaluating the likelihood.

   A normalization of the weights is done after computing all the values for all the particles.

   $$w_k^i = \frac{w_k^i}{\sum_i w_k^i}$$

2. *Estimation.*

   Once all the weights are modified according to the likelihood function, the posterior probability function can be approximated as well as the mean.

   $$\hat{p}(\boldsymbol{x}_k \mid \boldsymbol{Z}_k) = \sum_{i=1}^{N} w_k^i \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^i)$$

   $$\hat{\boldsymbol{x}}_k = \sum_{i=1}^{N} w_k^i \boldsymbol{x}_k^i$$

The weighted mean value $\hat{\boldsymbol{x}}_k$ is the estimation of the state. The function $\delta(u)$ is Dirac's delta.

3. *Resampling.*

   The resampling is intended to avoid the filter's depletion. This means that after some iterations, most of the particles will have negligible weights and a few of them will concentrate the probability mass. Finally, only one particle will remain with a weight equal to 1 and the filter will be doing the simulation of this particle. With the resampling, N new particles can be obtained so as to maintain the good operation of the filter.

   There are two resampling strategies. The first one is the Bayesian bootstrap and consists on taking N samples with replacement from the set $\{\boldsymbol{x}_k^i\}_{i=1}^N$ where the probability of each particle is $w_k^i = \frac{1}{N}$. This procedure is repeated in every iteration and is also known as Sampling Importance Resampling (SIR) [12], [13].

   The other option for the resampling is the Importance Sampling. In this case, the resampling is done when the effective number of samples is less than a threshold:

$$N_{eff} = (\frac{1}{\sum_i w_k^i})^2 \leq N_h$$

   the threshold can be chosen as $N_h = \frac{2N}{3}$ [14].

4. *Time update.*

   The last step is generating predictions for the particles in the next time step according to the proposal distribution, that is, according to the dynamic model.

$$\boldsymbol{x}_{k+1}^i \sim q(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_{1:k}, \boldsymbol{Z}_{k+1})$$

   Also a process noise $\boldsymbol{w}_{k+1}$ has to be added. The reason is that when resampling, the particles are, in general, replaced by the particles with a higher weight. So, the set of N particles can be formed by a few particles repeated. Adding noise to this particles provokes the N samples to be different and creates a diversity that makes the simulation richer.

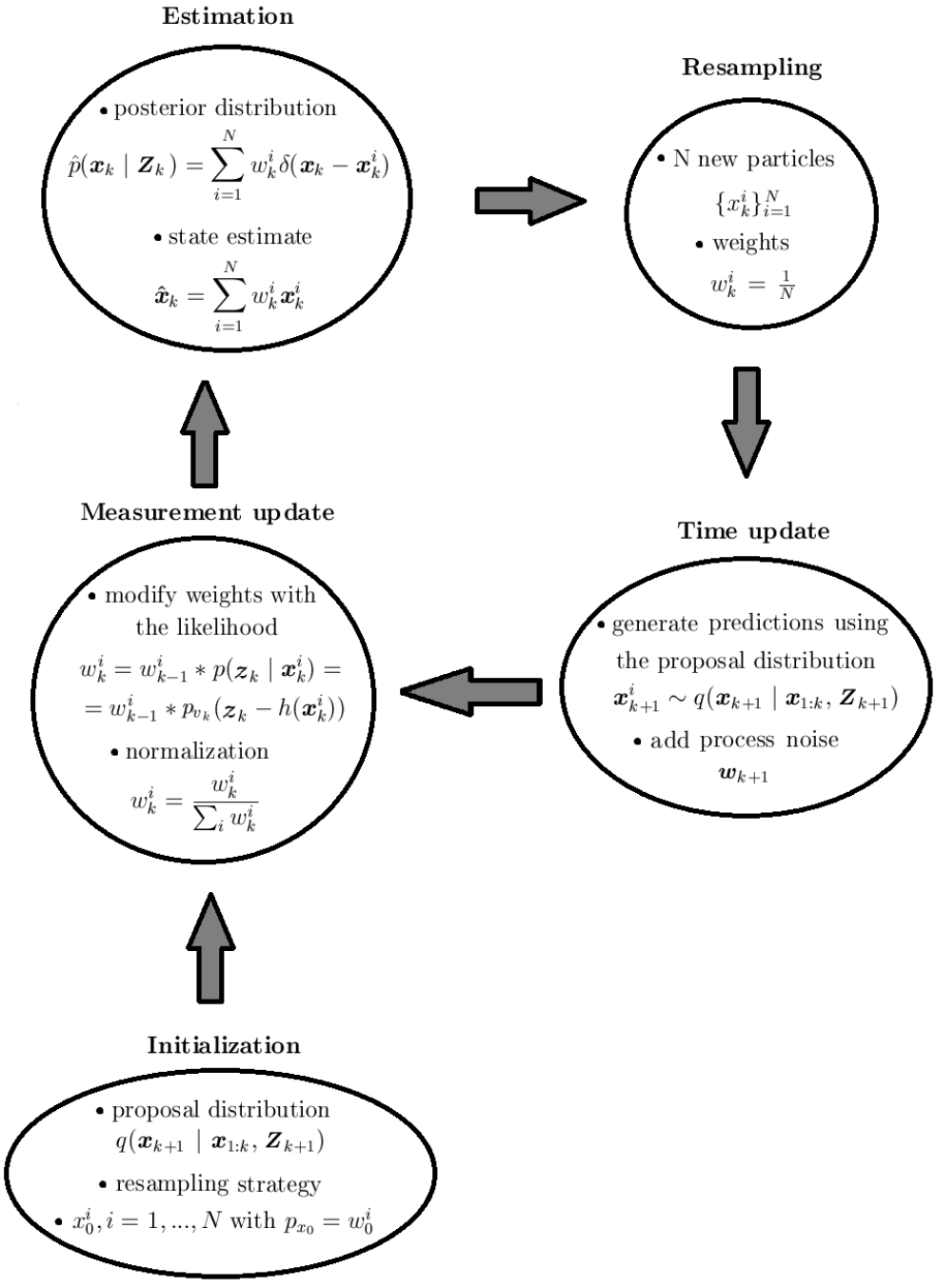

Figure 3.3: Operation of the Particle Filter.

Chapter 4

Dynamic models and designed algorithms

4.1 Dynamic models

In applications such as positioning, tracking or navigation, a good model for describing the motion of the object is fundamental [10].

There are two types of model for state estimation. The first type are the kinematic models that consider unknown inputs as process noise. The other type incorporates known inputs to the system.

The name of the dynamic model depends on which navigation parameters are included in the state vector. The reason for including a navigation parameter in the state vector is obtaining an estimation of it. If only the position is included in the state vector the name for the model is Constant Position model. If also the velocity is included, then the name is Constant Velocity model. Finally, if the state vector also contains the acceleration, then the name is Constant Acceleration model.

The motion can be considered in one dimension (X), two dimensions (X,Y) or three dimensions (X,Y,Z). As said before, in this project the application is intended to work in only two dimensions.

In Table 4.1, the state vectors and the dynamic equations for calculating the state in the next time step are presented. As said before, the noise $\boldsymbol{w}_k$ can be interpreted as a process noise for a pure kinematic model, or as an input in case of being a sensed motion. So, each dynamic model has two possibilities taking into account if it uses a measured navigation parameter as an input or not. If a measurement of the velocity is taken, it can be used as an input in a Constant Position model, and then the measurement noise has the role of the process noise. In the other hand, it might not be of interest to use the velocity measurement as an input (for example, if it is used

in the measurement update in a Kalman Filter), and it is preferred to estimate its value. Then, the chosen dynamic model has to be the Constant Velocity one. The same happens if an acceleration measurement is taken. It can be used as an input in a Constant Velocity model or it can be estimated in a Constant Acceleration model.

Model name	State	Dynamic equation
C. P.	$\boldsymbol{x}_k = \begin{bmatrix} x \\ y \end{bmatrix}$	$\boldsymbol{x}_{k+1} = \boldsymbol{I}_{2\times2}\boldsymbol{x}_k + T\boldsymbol{I}_{2\times2}\boldsymbol{w}_k$
C. V.	$\boldsymbol{x}_k = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}$	$\boldsymbol{x}_{k+1} = \begin{bmatrix} \boldsymbol{I}_{2\times2} & T\boldsymbol{I}_{2\times2} \\ \boldsymbol{0}_{2\times2} & \boldsymbol{I}_{2\times2} \end{bmatrix} \boldsymbol{x}_k + \begin{bmatrix} \frac{T^2}{2}\boldsymbol{I}_{2\times2} \\ T\boldsymbol{I}_{2\times2} \end{bmatrix} \boldsymbol{w}_k$
C. A.	$\boldsymbol{x}_k = \begin{bmatrix} x \\ y \\ v_x \\ v_y \\ a_x \\ a_y \end{bmatrix}$	$\boldsymbol{x}_{k+1} = \begin{bmatrix} \boldsymbol{I}_{2\times2} & T\boldsymbol{I}_{2\times2} & \frac{T^2}{2}\boldsymbol{I}_{2\times2} \\ \boldsymbol{0}_{2\times2} & \boldsymbol{I}_{2\times2} & T\boldsymbol{I}_{2\times2} \\ \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times2} & \boldsymbol{I}_{2\times2} \end{bmatrix} \boldsymbol{x}_k + \begin{bmatrix} \frac{T^3}{6}\boldsymbol{I}_{2\times2} \\ \frac{T^2}{2}\boldsymbol{I}_{2\times2} \\ T\boldsymbol{I}_{2\times2} \end{bmatrix} \boldsymbol{w}_k$

Table 4.1: Translational kinematics models in 2D. $\boldsymbol{I}_{2\times2}$ is the two-dimensional diagonal matrix and $\boldsymbol{0}_{2\times2}$ is a 2×2 matrix of zeros.

4.2 Designed algorithms

In this section, the designed algorithms for the application are presented. They will be divided in four groups. The first group consists of the methods that use the Extended Kalman Filter (EKF) algorithm for fusing the UWB range measurements with the DR system measurements, that is, the velocity and the orientation of the master. The second group includes the methods that also use the EKF algorithm, but now the information fused will come from the UWB range measurements and the INS which provides measurements of acceleration and angular rate. The third group consists only of one algorithm. It is the Complementary Kalman Filter (CKF) which uses the EKF algorithm for estimating the errors rather than the states. The CKF uses the information from the UWB system and the INS. The last group is also compound of one algorithm. In this case it uses the Particle Filter (PF) recursion for fusing the measurements that come from the UWB system and the DR system.

4.2.1 Extended Kalman Filter with Dead Reckoning System

As said, in this section the fusion algorithms that use the EKF and the information coming from the UWB system and the DR system are presented. Concretely, three

algorithms have been developed with this technique.

With velocity and orientation measurements coming from the DR system, the dynamic models that can be used are the Constant Position and the Constant Velocity. In the first case, the velocity and orientation measurements are used as an input to the EKF. In the second case, the measurements are introduced in the filter during the measurement update, so their function is correcting the a priori estimate of the state. In both algorithms, the UWB range measurements are used in the correction phase.

A special algorithm is included in this section. It follows the Constant Position model, but it does not use the DR measurements. Only the UWB range measurements are introduced in the correction phase of the filter.

Extended Kalman Filter for Constant Position model (CP)

This algorithm uses the EKF algorithm and the UWB range measurements. So, the velocity and the orientation are not needed. As it follows the Constant Position dynamic model the state vector only contains the position in two dimensions which has to be estimated.

$$\boldsymbol{x}_k = \begin{bmatrix} x_k \\ y_k \end{bmatrix}$$

The equation that characterizes the Constant Position dynamic model, as shown in Table 4.1, is:

$$\boldsymbol{x}_{k+1} = \boldsymbol{I}_{2\times 2}\boldsymbol{x}_k + T\boldsymbol{I}_{2\times 2}\boldsymbol{w}_k$$

And the measurement has the following expression:

$$\boldsymbol{z}_k = h(\hat{\boldsymbol{x}}_k^-) + \boldsymbol{v}_k$$

As said in Section 3.2 about the Extended Kalman Filter, $\boldsymbol{w}_K$ and $\boldsymbol{v}_k$ are the process noise and the measurement noise, respectively. It is important to note that the dynamic equation is linear and the non-linearity is in the measurement estimation equation, and that the measurement noise is additive and is not included in the non-linear function. So, only the Jacobian $\boldsymbol{H}$ will be necessary.

At this moment, the two groups of equations used in the EKF can be defined. First of all the time update equations, also known as prediction equations. The a priori estimate is obtained from:

$$\hat{\boldsymbol{x}}_k^- = \boldsymbol{I}_{2\times 2}\hat{\boldsymbol{x}}_{k-1}$$

and the a priori error covariance from:

$$\boldsymbol{P}_k^- = \boldsymbol{I}_{2\times2}\boldsymbol{P}_{k-1}\boldsymbol{I}_{2\times2}^T + T^2\boldsymbol{I}_{2\times2}\boldsymbol{Q}_k\boldsymbol{I}_{2\times2}^T$$

Before presenting the measurement update equations, the measurement function $h(\hat{\boldsymbol{x}}_k^-)$ has to be defined. As the only measurements used are the UWB range measurements (distance from the master to the four slaves), the $h(\hat{\boldsymbol{x}}_k^-)$ has to calculate the distance from the a priori master's position $\hat{\boldsymbol{x}}_k^- = \begin{bmatrix}\hat{x}_k^- \\ \hat{y}_k^-\end{bmatrix}$ to the known position of the slaves, denoted as $(S_{x,n}, S_{y,n})$ for slave n. As said before, the euclidean distance is the non-linear function that relates the a priori estimate of the state with the measurement:

$$h(\hat{\boldsymbol{x}}_k^-) = \begin{bmatrix} \sqrt{(\hat{x}_k^- - S_{x,1})^2 + (\hat{y}_k^- - S_{y,1})^2} \\ . \\ . \\ . \\ \sqrt{(\hat{x}_k^- - S_{x,4})^2 + (\hat{y}_k^- - S_{y,4})^2} \end{bmatrix}$$

Note that the measurement noise $\boldsymbol{v}_k$ is not added to the equation as well as the process noise $\boldsymbol{w}_k$ in the equation for calculating the a priori state estimate. Now, the Jacobian $\boldsymbol{H}_k$ can be directly derived as:

$$\mathbf{H}_k = \frac{\delta(h(\hat{\boldsymbol{x}}_k^-))}{\delta \boldsymbol{x}_k} = \begin{bmatrix} \frac{(\hat{x}_k^- - S_{x,1})}{\sqrt{(\hat{x}_k^- - S_{x,1})^2 + (\hat{y}_k^- - S_{y,1})^2}} & \frac{(\hat{y}_k^- - S_{y,1})}{\sqrt{(\hat{x}_k^- - S_{x,1})^2 + (\hat{y}_k^- - S_{y,1})^2}} \\ . & . \\ . & . \\ . & . \\ \frac{(\hat{x}_k^- - S_{x,4})}{\sqrt{(\hat{x}_k^- - S_{x,4})^2 + (\hat{y}_k^- - S_{y,4})^2}} & \frac{(\hat{y}_k^- - S_{y,4})}{\sqrt{(\hat{x}_k^- - S_{x,4})^2 + (\hat{y}_k^- - S_{y,4})^2}} \end{bmatrix}$$

Finally, the three equations for the measurement update step or correction step are the same that are used in the section about the EKF algorithm. So they will be recalled now, but in the following sections they will not be written, since they are always the same. The first one computes the Kalman gain. The second one corrects the a priori estimate using the innovation and the Kalman gain, obtaining the a posteriori state estimate. The last one updates the error covariance so as to obtain the a posteriori error covariance:

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}_k^T [\boldsymbol{H}_k \boldsymbol{P}_k^- \boldsymbol{H}_k^T + \boldsymbol{R}_k]^{-1}$$
$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(\boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_k^-))$$
$$\boldsymbol{P}_k = [\boldsymbol{I}_{2\times2} - \boldsymbol{K}_k \boldsymbol{H}_k]\boldsymbol{P}_k^-$$

The noise matrices $\boldsymbol{R}_k$ and $\boldsymbol{Q}_k$ can change in every iteration, and their value will depend on the scenario where the application works. They will have to be tuned to improve the operation of the algorithm and to get a better accuracy in the results.

Extended Kalman Filter for Constant Position model with velocity as an input (CPCV)

This algorithm is very similar to the one explained in the previous section (CP). It uses the EKF equations and the UWB range measurements, but also the velocity and the orientation provided by the DR system. The position is the only navigation state included in the state vector, therefore the dynamic model used is the Constant Position.

$$\boldsymbol{x}_k = \begin{bmatrix} x_k \\ y_k \end{bmatrix}$$

Once the state vector is defined, and knowing that the characterization of the model is the same that in the CP algorithm, the time update equations are presented.

$$\hat{\boldsymbol{x}}_k^- = \boldsymbol{A}\hat{\boldsymbol{x}}_{k-1} + \boldsymbol{B}\boldsymbol{u}$$

This is the equation that provides the a priori estimate, where:

$$\boldsymbol{A} = \boldsymbol{I}_{2\times 2}$$

$$\boldsymbol{B} = T\boldsymbol{I}_{2\times 2} = \begin{bmatrix} T & 0 \\ 0 & T \end{bmatrix}$$

$$\boldsymbol{u} = \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \end{bmatrix}$$

The vector $\boldsymbol{u}$ is the input that incorporates the velocity v and orientation θ measurements to the a priori state estimate. Due to that fact, the process noise covariance $\boldsymbol{Q}_k$ contains the measurement noise from the DR system. Then, the a priori error covariance is given by:

$$\boldsymbol{P}_k^- = \boldsymbol{A}\boldsymbol{P}_{k-1}\boldsymbol{A}^T + \boldsymbol{B}\boldsymbol{Q}_k\boldsymbol{B}^T$$

The measurement estimation $h(\hat{\boldsymbol{x}}_k^-)$ and the Jacobian $\boldsymbol{H}_k$ are the same than in the CP algorithm, since the UWB range measurements are the only ones used in the EKF correction phase.

$$h(\hat{\boldsymbol{x}}_k^-) = \begin{bmatrix} \sqrt{(\hat{x}_k^- - S_{x,1})^2 + (\hat{y}_k^- - S_{y,1})^2} \\ . \\ . \\ . \\ \sqrt{(\hat{x}_k^- - S_{x,4})^2 + (\hat{y}_k^- - S_{y,4})^2} \end{bmatrix}$$

$$\mathbf{H}_k = \frac{\delta(h(\hat{\boldsymbol{x}}_k^-))}{\delta \boldsymbol{x}_k} = \begin{bmatrix} \frac{(\hat{x}_k^- - S_{x,1})}{\sqrt{(\hat{x}_k^- - S_{x,1})^2 + (\hat{y}_k^- - S_{y,1})^2}} & \frac{(\hat{y}_k^- - S_{y,1})}{\sqrt{(\hat{x}_k^- - S_{x,1})^2 + (\hat{y}_k^- - S_{y,1})^2}} \\ . & . \\ . & . \\ . & . \\ \frac{(\hat{x}_k^- - S_{x,4})}{\sqrt{(\hat{x}_k^- - S_{x,4})^2 + (\hat{y}_k^- - S_{y,4})^2}} & \frac{(\hat{y}_k^- - S_{y,4})}{\sqrt{(\hat{x}_k^- - S_{x,4})^2 + (\hat{y}_k^- - S_{y,4})^2}} \end{bmatrix}$$

Lastly, the measurement update equation are the same than in the general case of the EKF algorithm.

Extended Kalman Filter for Constant Velocity model (CV)

This is the first algorithm designed to estimate the velocity as well as the position. So, corresponding to the Constant Velocity model, the state vector will be:

$$\boldsymbol{x}_k = \begin{bmatrix} x_k \\ y_k \\ v_{x,k} \\ v_{y,k} \end{bmatrix}$$

As seen in Table 4.1, the dynamic model is governed by the following equation:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{W}\boldsymbol{w}_k$$

where:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I}_{2\times 2} & T\boldsymbol{I}_{2\times 2} \\ \boldsymbol{0}_{2\times 2} & \boldsymbol{I}_{2\times 2} \end{bmatrix} =$$

$$\boldsymbol{W} = \begin{bmatrix} \frac{T^2}{2}\boldsymbol{I}_{2\times 2} \\ T\boldsymbol{I}_{2\times 2} \end{bmatrix}$$

Once the model is known, the time update equations for projecting the state and the error covariance ahead can be defined respectively as:

$$\hat{\boldsymbol{x}}_k^- = \boldsymbol{A}\hat{\boldsymbol{x}}_{k-1}$$

$$\boldsymbol{P}_k^- = \boldsymbol{A}\boldsymbol{P}_{k-1}\boldsymbol{A}^T + \boldsymbol{W}\boldsymbol{Q}_k\boldsymbol{W}^T$$

Since this algorithm is not using the measurements as inputs, the matrix $\boldsymbol{Q}_k$ is the process noise covariance.

At this moment, the measurement estimate $h(\hat{\boldsymbol{x}}_k^-)$ is defined. It has to predict the range values (distance from master to the four slaves) and also the DR velocity and orientation measurements:

$$h(\hat{\boldsymbol{x}}_k^-) = \begin{bmatrix} \sqrt{(\hat{x}_k^- - S_{x,1})^2 + (\hat{y}_k^- - S_{y,1})^2} \\ . \\ . \\ . \\ \sqrt{(\hat{x}_k^- - S_{x,4})^2 + (\hat{y}_k^- - S_{y,4})^2} \\ \sqrt{(\hat{v}_{x,k}^-)^2 + (\hat{v}_{y,k}^-)^2} \\ arctan(\frac{\hat{v}_{y,k}^-}{\hat{v}_{x,k}^-}) \end{bmatrix}$$

Then the Jacobian $\boldsymbol{H}_k = \frac{\delta(h(\hat{\boldsymbol{x}}_k^-))}{\delta \boldsymbol{x}_k}$ has the following expression:

$$\mathbf{H}_k = \begin{bmatrix} \frac{(\hat{x}_k^- - S_{x,1})}{\sqrt{(\hat{x}_k^- - S_{x,1})^2 + (\hat{y}_k^- - S_{y,1})^2}} & \frac{(\hat{y}_k^- - S_{y,1})}{\sqrt{(\hat{x}_k^- - S_{x,1})^2 + (\hat{y}_k^- - S_{y,1})^2}} & 0 & 0 \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ \frac{(\hat{x}_k^- - S_{x,4})}{\sqrt{(\hat{x}_k^- - S_{x,4})^2 + (\hat{y}_k^- - S_{y,4})^2}} & \frac{(\hat{y}_k^- - S_{y,4})}{\sqrt{(\hat{x}_k^- - S_{x,4})^2 + (\hat{y}_k^- - S_{y,4})^2}} & 0 & 0 \\ 0 & 0 & \frac{\hat{v}_{x,k}^-}{\sqrt{(\hat{v}_{x,k}^-)^2 + (\hat{v}_{y,k}^-)^2}} & \frac{\hat{v}_{y,k}^-}{\sqrt{(\hat{v}_{x,k}^-)^2 + (\hat{v}_{y,k}^-)^2}} \\ 0 & 0 & \frac{-\frac{\hat{v}_{y,k}^-}{\hat{v}_{x,k}^-}}{1+(\frac{\hat{v}_{y,k}^-}{\hat{v}_{x,k}^-})^2} & \frac{\frac{1}{\hat{v}_{x,k}^-}}{1+(\frac{\hat{v}_{y,k}^-}{\hat{v}_{x,k}^-})^2} \end{bmatrix}$$

The measurement update equations are the ones from the general EKF case, noting that the innovation will be the difference between the observed measurement $\boldsymbol{z}_k$ (including the UWB range measurements, and the velocity and orientation from the DR system) and the estimate one $h(\hat{\boldsymbol{x}}_k^-)$.

4.2.2 Extended Kalman Filter with Inertial Navigation System

Two algorithms are developed in this section. Both use the EKF equations for fusing the UWB range information with the acceleration and angular rate measurements provided by the INS. The INS acceleration measurements are in body frame coordinates, so the rotation matrix $\boldsymbol{T}_b^l$ has to be applied for having the acceleration in the local frame coordinates, as described in Section 2.2.3.

The first method uses the Constant Velocity model, so the estimated navigation states are the position and the velocity. The INS measurements are used as an input in the EKF, and the UWB ranges are used in the correction phase. The second algorithm includes the acceleration in the state vector so that it can be estimated, due to that fact, the Constant Acceleration dynamics characterize the model. The UWB and INS measurements are both used for correcting in the EKF.

Extended Kalman Filter for Constant Velocity model using acceleration as an input (CVCA)

As commented before, the state vector includes the position and the velocity of the master, being the general state vector for the Constant Velocity model.

$$\boldsymbol{x}_k = \begin{bmatrix} x_k \\ y_k \\ v_{x,k} \\ v_{y,k} \end{bmatrix}$$

The dynamic equation that characterizes the model is the same that governs the CV algorithm explained before, but taking into account that the process noise $\boldsymbol{w}_k$ is substituted by the acceleration measurements in the local frame and the matrix $\boldsymbol{W}$ now is $\boldsymbol{B}$ (referring to the matrix that relates the input to the state), so the time update equation for the state vector is:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}$$

where:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I}_{2\times 2} & T\boldsymbol{I}_{2\times 2} \\ \boldsymbol{0}_{2\times 2} & \boldsymbol{I}_{2\times 2} \end{bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix} \frac{T^2}{2}\boldsymbol{I}_{2\times 2} \\ T\boldsymbol{I}_{2\times 2} \end{bmatrix}$$

$$\boldsymbol{u} = \boldsymbol{a}^l = \begin{bmatrix} a_x \\ a_y \end{bmatrix} = \boldsymbol{T}_b^l \boldsymbol{a}^b$$

The input vector $\boldsymbol{u}$ incorporates the acceleration a and the angular rate ω to the measurement. One consideration is taken: the two components of the acceleration a_x and a_y are calculated using the master's orientation obtained by propagating the navigation equation that relates the orientation θ with the angular rate ω, see Figure 2.4.

The a priori error covariance is given by:

$$\boldsymbol{P}_k^- = \boldsymbol{A}\boldsymbol{P}_{k-1}\boldsymbol{A}^T + \boldsymbol{B}\boldsymbol{Q}_k\boldsymbol{B}^T$$

Since the INS measurements are used as inputs, the UWB ranges are the only ones that are used in the correction phase of the EKF, including them in the measurement vector $\boldsymbol{z}_k$. So the measurement estimate $h(\hat{\boldsymbol{x}}_k^-)$ has to predict the ranges in the same way that the CP or the CPCV algorithm do.

$$h(\hat{\boldsymbol{x}}_k^-) = \begin{bmatrix} \sqrt{(\hat{x}_k^- - S_{x,1})^2 + (\hat{y}_k^- - S_{y,1})^2} \\ . \\ . \\ . \\ \sqrt{(\hat{x}_k^- - S_{x,4})^2 + (\hat{y}_k^- - S_{y,4})^2} \end{bmatrix}$$

Then the Jacobian $\boldsymbol{H}_k$ is:

$$\mathbf{H}_k = \frac{\delta(h(\hat{\boldsymbol{x}}_k^-))}{\delta \boldsymbol{x}_k} = \begin{bmatrix} \frac{(\hat{x}_k^- - S_{x,1})}{\sqrt{(\hat{x}_k^- - S_{x,1})^2 + (\hat{y}_k^- - S_{y,1})^2}} & \frac{(\hat{y}_k^- - S_{y,1})}{\sqrt{(\hat{x}_k^- - S_{x,1})^2 + (\hat{y}_k^- - S_{y,1})^2}} \\ . & . \\ . & . \\ . & . \\ \frac{(\hat{x}_k^- - S_{x,4})}{\sqrt{(\hat{x}_k^- - S_{x,4})^2 + (\hat{y}_k^- - S_{y,4})^2}} & \frac{(\hat{y}_k^- - S_{y,4})}{\sqrt{(\hat{x}_k^- - S_{x,4})^2 + (\hat{y}_k^- - S_{y,4})^2}} \end{bmatrix}$$

The measurement update phase follows the general case for the EKF, using the defined $h(\hat{\boldsymbol{x}}_k^-)$ and $\boldsymbol{H}_k$.

Extended Kalman Filter for Constant Acceleration model (CA)

In this algorithm the position, velocity and acceleration of the master are estimated. This is the reason for using the Constant Acceleration model. Then the state vector is:

$$\boldsymbol{x}_k = \begin{bmatrix} x_k \\ y_k \\ v_{x,k} \\ v_{y,k} \\ a_{x,k} \\ a_{y,k} \end{bmatrix}$$

The UWB range measurements and the INS measurements are both used in the correction phase of the EKF, so in the dynamic equation for the Constant Acceleration model from Table 4.1, the vector $\boldsymbol{w}_k$ is the process noise.

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{W}\boldsymbol{w}_k$$

where:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I}_{2\times2} & T\boldsymbol{I}_{2\times2} & \frac{T^2}{2}\boldsymbol{I}_{2\times2} \\ \boldsymbol{0}_{2\times2} & \boldsymbol{I}_{2\times2} & T\boldsymbol{I}_{2\times2} \\ \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times2} & \boldsymbol{I}_{2\times2} \end{bmatrix}$$

$$\boldsymbol{W} = \begin{bmatrix} \frac{T^3}{6}\boldsymbol{I}_{2\times2} \\ \frac{T^2}{2}\boldsymbol{I}_{2\times2} \\ T\boldsymbol{I}_{2\times2} \end{bmatrix}$$

The a priori state estimate and the a priori error covariance are obtained from the time update equations:

$$\hat{\boldsymbol{x}}_k^- = \boldsymbol{A}\hat{\boldsymbol{x}}_{k-1}$$

$$\boldsymbol{P}_k^- = \boldsymbol{A}\boldsymbol{P}_{k-1}\boldsymbol{A}^T + \boldsymbol{W}\boldsymbol{Q}_k\boldsymbol{W}^T$$

For simplicity, the estimated measurements in $h(\hat{\boldsymbol{x}}_k^-)$ are the ranges and the accelerations in local frame. The angular rate w is not estimated since it is used for calculating the orientation of the master (propagating the navigation equation). With the master's orientation, the rotation matrix $\boldsymbol{T}_b^l$ can be calculated and the INS acceleration measurements can be rotated from body to local frame and they can be included in the measurement vector $\boldsymbol{z}_k$.

$$h(\hat{\boldsymbol{x}}_k^-) = \begin{bmatrix} \sqrt{(\hat{x}_k^- - S_{x,1})^2 + (\hat{y}_k^- - S_{y,1})^2} \\ . \\ . \\ . \\ \sqrt{(\hat{x}_k^- - S_{x,4})^2 + (\hat{y}_k^- - S_{y,4})^2} \\ \hat{a}_{x,k}^- \\ \hat{a}_{y,k}^- \end{bmatrix}$$

The Jacobian $\boldsymbol{H}_k$ necessary for the measurement update equations, which follow the general case for the EKF as the other algorithms do, is:

$$\mathbf{H}_k = \frac{\delta(h(\hat{\boldsymbol{x}}_k^-))}{\delta \boldsymbol{x}_k} = \begin{bmatrix} \frac{(\hat{x}_k^- - S_{x,1})}{\sqrt{(\hat{x}_k^- - S_{x,1})^2 + (\hat{y}_k^- - S_{y,1})^2}} & \frac{(\hat{y}_k^- - S_{y,1})}{\sqrt{(\hat{x}_k^- - S_{x,1})^2 + (\hat{y}_k^- - S_{y,1})^2}} & 0 & 0 & 0 & 0 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \frac{(\hat{x}_k^- - S_{x,4})}{\sqrt{(\hat{x}_k^- - S_{x,4})^2 + (\hat{y}_k^- - S_{y,4})^2}} & \frac{(\hat{y}_k^- - S_{y,4})}{\sqrt{(\hat{x}_k^- - S_{x,4})^2 + (\hat{y}_k^- - S_{y,4})^2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

4.2.3 Complementary Kalman Filter (CKF)

The Complementary Kalman Filter is the name given to an algorithm that uses the EKF equations for estimating the errors in the navigations states rather than the states [1],[3],[15]. In Figure 4.1 the architecture of the algorithm is presented.

As said before, the measurements to be fused in this algorithm are the UWB ranges and the acceleration and the angular rate of the master measured by the accelerometer and the gyroscope installed in the IMU. With the IMU data, the navigation equations are propagated and the state vector is estimated for the current time step. This estimation is considered the navigation solution, that is, the result for the positioning application. But not only the position is estimated, also the velocity and the orientation of the master. So, the state vector is:

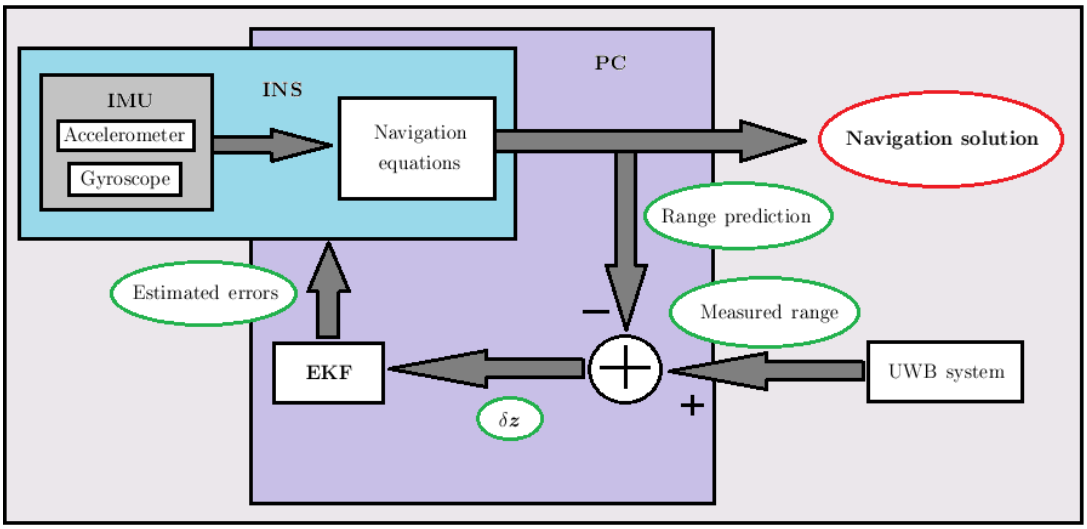

Figure 4.1: Fusion algorithm based on the architecture of the Complementary Kalman filter.

$$\boldsymbol{x}_k = \begin{bmatrix} x_k \\ y_k \\ z_k \\ v_{x,k} \\ v_{y,k} \\ v_{z,k} \\ \phi_k \\ \theta_k \\ \psi_k \end{bmatrix}$$

This algorithm has been developed in a three dimensional model. But as the application is working in two dimensions, the parameters that are not used are set to 0. In this case, only the position and velocity in X and Y, and the yaw angle ψ (orientation of the master) are estimated. The other two angles are the roll ϕ and the pitch θ.

Once the navigation solution is obtained, and hence the position, a prediction of the range (distance from master to slaves) can be made. Then, when a UWB range measurement is taken, the difference between the observed ranges and the predicted ones is calculated, and called $\delta \boldsymbol{z}$. This difference in the ranges will be the innovation for the correction phase of the EKF.

The originality of this algorithm is that the EKF is estimating the error in the states. The reason for doing this is inserting the errors in the calibration points of the navigation equations (see Figure 2.2) in order to obtain a corrected estimate of the state in the next time step. The procedure of calculating the state with the navigation equations, running the EKF to estimate the errors in the navigation states, and inserting

the errors as a feedback in the navigation equations is the base of the CKF algorithm.

Now the parameters of the algorithm will be discussed with more detail. First of all, the navigation equations that relates the previous state and the IMU measurements to the current state are presented:

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{I}_{3\times3} & T\boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} \end{bmatrix} \boldsymbol{x}_{k-1} + \begin{bmatrix} \frac{T^2}{2}\boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ T\boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & T\boldsymbol{I}_{3\times3} \end{bmatrix} \boldsymbol{T}_b^l \begin{bmatrix} \tilde{\boldsymbol{a}}^b \\ \tilde{\boldsymbol{\omega}}^b \end{bmatrix}$$

Note that the IMU measurements are rotated from body to local coordinates in order to calculate the states in the local frame.

The state vector used in the EKF will be called error state vector since it contains the error in the navigation states. The errors to be estimated are the error in the position, the error in the velocity, the error in the orientation, the bias in the acceleration and the bias in the angular rate:

$$\delta\boldsymbol{x}_k = \begin{bmatrix} \delta\boldsymbol{p}_k^l \\ \delta\boldsymbol{v}_k^l \\ \delta\boldsymbol{\theta}_k^l \\ b\boldsymbol{a}_k^b \\ b\boldsymbol{w}_k^b \end{bmatrix}$$

Each vector in the error state vector has three components, so there are fifteen parameters to be estimated. It is important to see that the errors in the navigation states are estimated directly in local coordinates, but the bias in the IMU measurements are in body coordinates. The reason is that the navigation states are corrected after the propagation of the navigation equations and then the states are in local coordinates, and the IMU measurements are corrected before propagating the navigation equations, when they are still in body coordinates. The rotation matrix $\boldsymbol{T}_b^l$ also uses the yaw angle ψ corrected with the estimated error.

At this moment the equations for the EKF can be defined. The first one in the time update phase projects the previous error state vector ahead, so as to obtain the a priori error state estimate:

$$\delta\hat{\boldsymbol{x}}_k^- = \boldsymbol{E}_k\delta\hat{\boldsymbol{x}}_{k-1}$$

The matrix $\boldsymbol{E}_k$ is the error function matrix that relates the error states at the previous time step $k-1$ to the error states at the current step k [3],[6]. It is defined as:

$$\boldsymbol{E}_k = \begin{bmatrix} \boldsymbol{I}_{3\times3} & T\boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} & -T\hat{\boldsymbol{\Omega}}_{a,k} & T\hat{\boldsymbol{T}}^l_{b,k} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} & T\hat{\boldsymbol{T}}^l_{b,k} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} \end{bmatrix}$$

Where $\hat{\boldsymbol{\Omega}}_{a,k}$ is the skew symmetric matrix of the acceleration measurements in local coordinates:

$$\hat{\boldsymbol{\Omega}}_{a,k} = \begin{bmatrix} 0 & \tilde{a}^l_{z,k} & -\tilde{a}^l_{y,k} \\ -\tilde{a}^l_{z,k} & 0 & \tilde{a}^l_{x,k} \\ \tilde{a}^l_{y,k} & -\tilde{a}^l_{x,k} & 0 \end{bmatrix}$$

Now the a priori error covariance can be defined:

$$\boldsymbol{P}^-_k = \boldsymbol{E}_k \boldsymbol{P}_{k-1} \boldsymbol{E}^T_k + \boldsymbol{Q}_k$$

The matrix $\boldsymbol{Q}_k$ is the process noise covariance, and has to be tuned before the initialization of the algorithm, and it can vary during the operation.

To use the EKF structure, the range measurements need to be related to the error states. With the known position of the slaves $\boldsymbol{S}_n = (S_{x,n}, S_{y,n}, S_{z,n})$ for slave n (assuming that $S_{z,n} = 0$), and the estimated position of the master provided by the INS, a range estimate can be calculated. However, the error in the position still has to be introduced. Then the measurement function $h(\delta\hat{\boldsymbol{x}}^-_k)$ will be the difference between the range estimates taking into account the error in the master position and the range estimate without the error, a kind of estimation of the error in the ranges:

$$h(\delta\hat{\boldsymbol{x}}^-_k) = \begin{bmatrix} ||\boldsymbol{S}_1 - (\delta\hat{\boldsymbol{p}}^-_k + \hat{\boldsymbol{x}}_{INS,k})|| - ||\boldsymbol{S}_1 - \hat{\boldsymbol{x}}_{INS,k}|| \\ \cdot \\ \cdot \\ \cdot \\ ||\boldsymbol{S}_4 - (\delta\hat{\boldsymbol{p}}^-_k + \hat{\boldsymbol{x}}_{INS,k})|| - ||\boldsymbol{S}_4 - \hat{\boldsymbol{x}}_{INS,k}|| \end{bmatrix}$$

Where:

$$||\boldsymbol{S}_n - (\delta\hat{\boldsymbol{p}}^-_k + \hat{\boldsymbol{x}}_{INS,k})|| = \sqrt{[S_{x,n} - (\hat{x}_k + \delta\hat{x}^-_k)]^2 + [S_{y,n} - (\hat{y}_k + \delta\hat{y}^-_k)]^2 + [S_{z,n} - (\hat{z}_k + \delta\hat{z}^-_k)]^2}$$

$$||\boldsymbol{S}_n - \hat{\boldsymbol{x}}_{INS,k}|| = \sqrt{[S_{x,n} - \hat{x}_k]^2 + [S_{y,n} - \hat{y}_k]^2 + [S_{z,n} - \hat{z}_k]^2}$$

The a priori error estimate in the master position is indicated in the vector $\delta\hat{\boldsymbol{p}}_k^- = (\delta\hat{x}_k^-, \delta\hat{y}_k^-, \delta\hat{z}_k^-)$. The INS estimated position of the master is $\hat{\boldsymbol{x}}_{INS,k} = (\hat{x}_k, \hat{y}_k, \hat{z}_k)$. The $||.||$ is indicating the euclidean distance.

Now the Jacobian $\boldsymbol{H}_k$ can be defined:

$$\boldsymbol{H}_k = \frac{\delta(h(\delta\hat{\boldsymbol{x}}_k^-))}{\delta(\delta\boldsymbol{x}_k)} = \begin{bmatrix} \frac{S_{x,1}-(\hat{x}_k+\delta\hat{x}_k^-)}{||\boldsymbol{S}_1-(\delta\hat{\boldsymbol{p}}_k^-+\hat{\boldsymbol{x}}_{INS,k})||} & \frac{S_{y,1}-(\hat{y}_k+\delta\hat{y}_k^-)}{||\boldsymbol{S}_1-(\delta\hat{\boldsymbol{p}}_k^-+\hat{\boldsymbol{x}}_{INS,k})||} & \frac{S_{z,1}-(\hat{z}_k+\delta\hat{z}_k^-)}{||\boldsymbol{S}_1-(\delta\hat{\boldsymbol{p}}_k^-+\hat{\boldsymbol{x}}_{INS,k})||} & \boldsymbol{0}_{1\times 12} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \frac{S_{x,4}-(\hat{x}_k+\delta\hat{x}_k^-)}{||\boldsymbol{S}_4-(\delta\hat{\boldsymbol{p}}_k^-+\hat{\boldsymbol{x}}_{INS,k})||} & \frac{S_{y,4}-(\hat{y}_k+\delta\hat{y}_k^-)}{||\boldsymbol{S}_4-(\delta\hat{\boldsymbol{p}}_k^-+\hat{\boldsymbol{x}}_{INS,k})||} & \frac{S_{z,4}-(\hat{z}_k+\delta\hat{z}_k^-)}{||\boldsymbol{S}_4-(\delta\hat{\boldsymbol{p}}_k^-+\hat{\boldsymbol{x}}_{INS,k})||} & \boldsymbol{0}_{1\times 12} \end{bmatrix}$$

The $\boldsymbol{0}_{1\times 12}$ is a row vector of twelve zeros.

The measurement update phase follows the general case of the EKF algorithm. First the Kalman gain is calculated, and then the a posteriori error estimates and the a posteriori error covariance are computed:

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}_k^T [\boldsymbol{H}_k \boldsymbol{P}_k^- \boldsymbol{H}_k^T + \boldsymbol{R}_k]^{-1}$$

$$\delta\hat{\boldsymbol{x}}_k = \delta\hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(\delta\boldsymbol{z}_k - h(\delta\hat{\boldsymbol{x}}_k^-))$$

$$\boldsymbol{P}_k = [\boldsymbol{I}_{2\times 2} - \boldsymbol{K}_k \boldsymbol{H}_k]\boldsymbol{P}_k^-$$

As explained before, the vector $\delta\boldsymbol{z}_k$ is the difference between the UWB range measurements and the predicted ranges using the estimated master position $\hat{\boldsymbol{x}}_{INS,k}$ at step k. The innovation is the difference between $\delta\boldsymbol{z}_k$ and the measurement function $h(\delta\hat{\boldsymbol{x}}_k^-)$.

Luckily, the insertion of the estimated errors in the navigation equations permits to simplify the prediction and correction phases of the EKF. The reason is that once the errors are inserted, it is assumed that now the states are correct and there is no error. Then the a priori error estimate $\delta\hat{\boldsymbol{x}}_k^-$ is set to zero. And the time update phase (or prediction phase) is only based on calculating the a priori error covariance $\boldsymbol{P}_k^-$.

In the measurement update phase (or correction phase), the measurement function $h(\delta\hat{\boldsymbol{x}}_k^-)$ is zero due to it is assumed that there is no a priori error. In the Jacobian $\boldsymbol{H}_k$ the values of the a priori error estimate that are used are zero. So the expression is simplified.

When calculating the posterior error estimate, only the Kalman gain $\boldsymbol{K}_k$ and the vector $\delta\boldsymbol{z}_k$ are needed. Then the a posteriori error estimates are the ones to be inserted in the next iteration in the navigation equations and the algorithm is repeated, setting again the a priori error estimates to zero.

One last thing about the algorithm is that the values of the accelerometer bias and the gyroscope bias, are saved in every iteration, and the new values from the a posteriori error estimate are added to the saved values. This saved values are the ones used to correct the IMU biases. The reason for doing this is that the mean of the biases is not zero, and the changes in the biases are characterized as a random walk. Saving the values of the biases and incorporating the new changes from the a posteriori estimate, is the way to estimate the real values of the biases.

4.2.4 Particle Filter with Dead Reckoning System (PF)

This algorithm is designed to estimate the position and velocity of the master fusing the UWB ranging measurements and the velocity and orientation measurements from the IMU using the Particle Filter. Then, the state vector is as follows:

$$\boldsymbol{x}_k = \begin{bmatrix} x_k \\ y_k \\ v_{x,k} \\ v_{y,k} \end{bmatrix}$$

This is the Constant Velocity state vector, so the dynamic model is governed by the following equation:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{W}\boldsymbol{w}_k$$

where:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I}_{2\times 2} & T\boldsymbol{I}_{2\times 2} \\ \boldsymbol{0}_{2\times 2} & \boldsymbol{I}_{2\times 2} \end{bmatrix} =$$

$$\boldsymbol{W} = \begin{bmatrix} \frac{T^2}{2}\boldsymbol{I}_{2\times 2} \\ T\boldsymbol{I}_{2\times 2} \end{bmatrix}$$

The proposal distribution $q(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_{1:k}, \boldsymbol{Z}_{k+1})$ will follow this model, but the process noise term $\boldsymbol{W}\boldsymbol{w}_k$ will be added apart. Concretely a white Gaussian noise with distribution $p_{w_k} \sim N(\boldsymbol{0}, \boldsymbol{Q})$, and its covariance $\boldsymbol{Q}$ will have to be tuned for obtaining the best results.

Also the measurement has an equation that relates the state with the own measurement:

$$\boldsymbol{z}_k = h(\boldsymbol{x}_k) + \boldsymbol{v}_k$$

The measurement noise also has a white Gaussian distribution $p_{v_k} \sim N(\mathbf{0}, \boldsymbol{R})$, with a known covariance $\boldsymbol{R}$. This distribution will be used as the likelihood function during the measurement update phase. A tuning of the measurement covariance $\boldsymbol{R}$ is also possible.

In the initialization of the algorithm, the number of particles $\boldsymbol{x}_0^i, i = 1, ..., N$ is set to N=200. For their generation, four uniform distributions are chosen. The two first are for generating the particles dedicated to the position. The limits of the uniform distribution are, assuming that the master will not leave the area delimited by the slaves, the minimum and maximum values in the X component for x_k and the minimum and maximum values in the Y component for y_k. The more reduced the range of position, the better results in the initialization, because the particles will cover the range better. The other two uniform distributions will generate the particles dedicated to the initial velocity in X, $v_{x,k}$, and in Y, $v_{y,k}$. The range of the uniform distributions will be reduced if the knowledge about the possible velocities is high. The weights are the same for all the particles and equal to $p_{x_0} = w_0^i = \frac{1}{N} = \frac{1}{200}$.

Once the initialization is done, the iterations of the Particle Filter algorithm can start. The first phase is the measurement update, in which a prediction of the measurement is calculated. Specifically, the function $h(\boldsymbol{x}_k)$ has to predict the UWB range values, and the velocity and orientation measurements from the IMU. This calculation has to be made for all the particles (the prediction for the i-th particle is indicated with the superscript i):

$$\boldsymbol{z}_k^i = h(\boldsymbol{x}_k^i) = \begin{bmatrix} \sqrt{(x_k^i - S_{x,1})^2 + (y_k^i - S_{y,1})^2} \\ . \\ . \\ . \\ \sqrt{(x_k^i - S_{x,4})^2 + (y_k^i - S_{y,4})^2} \\ \sqrt{(v_{x,k}^i)^2 + (v_{y,k}^i)^2} \\ arctan(\frac{v_{y,k}^i}{v_{x,k}^i}) \end{bmatrix}$$

When all the predicted measurements are calculated, it is time to evaluate them in order to modify the weight of the particles. An observed measurement is introduced to the filter, and the difference between the real measurement and the predicted one is evaluated in the likelihood function p_{e_k} and multiplied by the previous weight:

$$w_k^i = w_{k-1}^i * p(\boldsymbol{z}_k \mid \boldsymbol{x}_k^i) = w_{k-1}^i * p_{v_k}(\boldsymbol{z}_k - h(\boldsymbol{x}_k^i))$$

Now the particles with a higher similarity to the real state, have a higher weight. Lastly, all the weights have to be normalized.

Once the weights are modified, it is possible to calculate an approximation to the state

estimate:

$$\hat{\boldsymbol{x}}_k = \begin{bmatrix} \hat{x}_k \\ \hat{y}_k \\ \hat{v}_{x,k} \\ \hat{v}_{y,k} \end{bmatrix} = \sum_{i=1}^{N} w_k^i \boldsymbol{x}_k^i$$

In order to avoid the impoverishment of the simulation, a resampling has to be done. If not, a few particles will concentrate the probability mass, even one of them can reach a probability of one, leading to the simulation of the trajectory of a unique particle. The resampling strategy that has been chosen is commonly referred to as Ripley's method (it can be seen in [10]). The resampling is repeated in every iteration, so it is a SIR procedure. After the resampling, all the weights become equiprobable again.

The last step is the time update, in which the particles are changed according to the proposal distribution (the dynamic model) so as to generate the state predictions for the next time step $\boldsymbol{x}_{k+1}^i, i = 1, ..., N$. The process noise $\boldsymbol{w}_{k+1}$ has to be added to each particle according to the noise probability function $p_{w_{k+1}}$. At this moment, the new particles obtained are the ones that will be evaluated in the next iteration.

These are the steps for the Particle Filter recursion. Only note that the noise distributions p_{w_k} and p_{v_k}, can be changed in every iteration if it is necessary.

Chapter 5

Simulation of algorithms and comparison

This chapter is dedicated to show the results of testing the algorithms described in previous chapters with simulated trajectories using Matlab.

5.1 Simulation setup

The trajectories are generated following the translational kinematics models shown in Table 4.1. So, one trajectory will follow the Constant Position model, another the Constant Velocity model and, the last one, the Constant Acceleration model. In all cases, the noise term $\boldsymbol{w}_k$ is treated as white Gaussian process noise. Furthermore, another trajectory is tested. It is called spline trajectory, since it is created with the "spline" function of Matlab, which connects the points given as input to the function with paths built using piecewise linear accelerations. As this trajectory is not following none of the dynamic models used in the developed algorithms, it will be a good test for them.

The procedure for obtaining the true trajectories and the noisy measurements begins with the propagation of the 2D translational kinematics equations of the dynamic models during some time steps. The period of time between two samples is T and it is set to 100 ms. In every iteration, a sample of the state vector is saved. Once all the necessary samples are obtained, the calculation of the measurements is done. Only the position information of the states vector is necessary. The true trajectory is constituted by the position itself in X and Y components. The velocity is obtained by making the difference between two consecutive position samples and dividing by T, first for one component and then for the other. The acceleration is calculated by making the difference of two consecutive velocity samples and dividing by T, again for the two components. The ranges are computed by calculating the euclidean distance

between the position of the master in each iteration and the position of the four slaves. The slaves are located in the vertices of a $10m \times 10m$ square. Concretely, their coordinates are (0,0), (10,0), (10,10) and (0,10) all of them in meters. The orientation is obtained calculating the *arctan* of the two components of the velocity. Finally, the angular rate is the difference between two consecutive samples of the orientation divided by the sampling time T. By this way, the true trajectory of the master and all the possible true measurements can be obtained.

In each algorithm only the necessary measurements will be used, according to their description in Chapter 4. In order to acquire the noisy measurements, white Gaussian noise is added to all of them as well as a bias term in the acceleration and angular rate. The variances used in the additive white Gaussian measurement noise are indicated in Section 2.1 for the range measurements, in Section 2.2.1 for the acceleration and angular rate, and in Section 2.3 for the velocity and orientation. The biases added to the acceleration and the angular rate are indicated in Section 2.2.

At this moment, all the necessary parameters are calculated. The last thing to do is providing the initial state vector and error covariance matrix. Also the process noise matrix $\boldsymbol{Q}$ and the measurement noise matrix $\boldsymbol{R}$ has to be given, and since these trajectories have a controlled generation, the process noise and the measurement noise can be set with the same variance values used in the generation (in principle, the optimal values), but they can be tuned for better performance. For the Particle Filter the number of particles has been set to 200.

Once the simulations are finished, the comparison between the algorithms is done by looking at two parameters. The first one, and the most important, is the Root-Mean-Square Error (RMSE) of the difference between the true positions of the trajectory and the estimated ones. The expression is written as follows:

$$PositionRMSE = \sqrt{\frac{\sum_{i=1}^{N^oSamples}(||\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k||)2}{N^oSamples}}$$

where:

$$||\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k|| = \sqrt{(x_k - \hat{x}_k)^2 + (y_k - \hat{y}_k)^2}$$

This error can also be evaluated in other navigational states such as the velocity or the orientation. But in this project, only the position error is evaluated since it is intended to develop indoor positioning algorithms.

The other parameter that will be compared between the algorithms is the computing time necessary for running the simulations. The less computing time, the better the algorithm. Obviously, for real-time operation, it has to be less than:

$$MaxComputingTime = (NumberOfSamples) \times 100ms$$

Finally, note that the obtained results will be the mean of 50 different simulations.

5.2 Simulation with a Constant Position trajectory

The translational kinematic model followed for creating the trajectory is:

$$\boldsymbol{x}_{k+1} = \boldsymbol{I}_{2\times 2}\boldsymbol{x}_k + T\boldsymbol{I}_{2\times 2}\boldsymbol{w}_k$$

The initial position is located at $\boldsymbol{x}_0 = (5, 5)$. The process noise $\boldsymbol{w}_k$ has units of velocity, and its effects are the change of the master position. Its standard deviation in each component is $\sigma_x = \sigma_y = \sqrt{0.1}\frac{m}{s}$.

The number of samples taken are 1000. Once the trajectory is generated and the measurements calculated, the simulations can be done. The initial state vectors introduced in the algorithms that used the EKF recursion, contain the true initial position $(5, 5)$. In the PF algorithm, which uses the Particle Filter recursion, the uniform distributions for generating the first particles have a very short range close to the true initial position and velocity (supposed to be zero). All the other parameters ($\boldsymbol{Q}$,$\boldsymbol{R}$ and $\boldsymbol{P}_0$) have been tuned for obtaining the best accuracies in the positioning. The resulting trajectories for each algorithm are shown in Figure 5.1 a) (the trajectories are from one of the 50 different simulations).

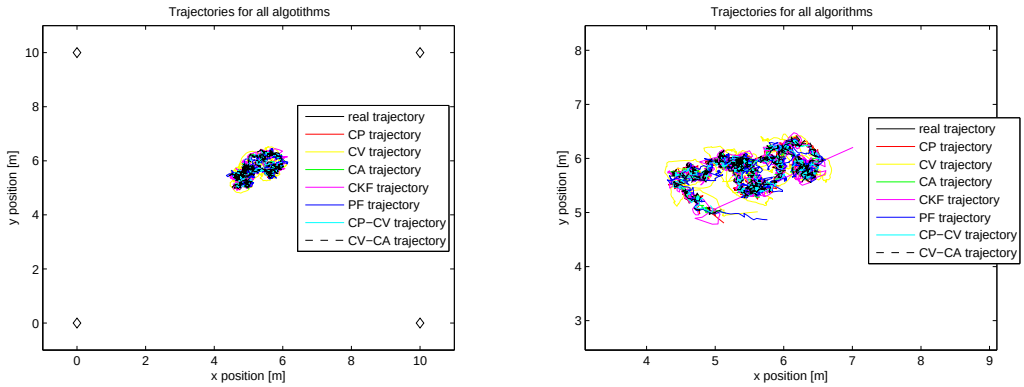

Figure 5.1: Trajectories obtained applying the algorithms in a Constant Position trajectory. a) On the left, correct initialization. b) On the right, random initialization.

All the algorithms seem to have a good performance, since no one of them is diverging. The graphics with the RMSE error in the position and the time spent for the simulation will help in the discussion about which algorithm is better. They can be seen in Figure 5.2 a).

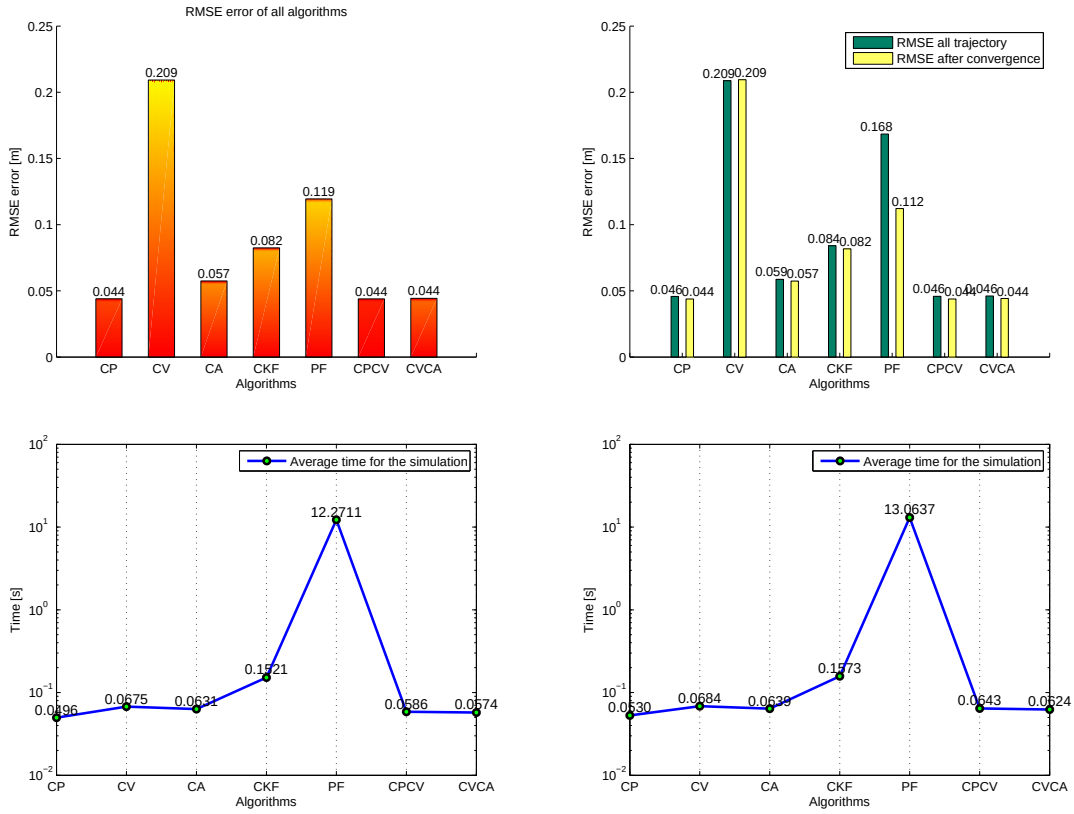

Figure 5.2: a) On the left, above is presented the bar graph of the RMSE errors and below the computing times for correct initialization. b) On the right, RMSE errors and computing times for random initialization.

At first sight, the CP, CPCV, CVCA and CA algorithms seem to obtain the better accuracies, in the order of 4-5 cm, while the CV has an RMSE error of 20 cm. Taking into account the computing time, all of them have similar results, in the order of 50-100 ms. However, the PF algorithm is the slowest, needing 12 s to run the simulation. Luckily, it is not exceeding the maximum computing time permitted which is 100 s.

Before choosing the best algorithms for this dynamic model, a simulation with random initial position will be done. In Figure 5.1 b) the trajectories are shown, and in Figure 5.2 b) the errors and the computing times.

In the plot of the trajectories it can be seen that the filters are converging in a few iterations. Besides, it is easy to see that the CV algorithm has some problems to follow the trajectory, probably due to the model used in this algorithm. In the RMSE bar graph, the errors are similar to the values obtained with the correct initialization. The computing times are similar too.

In conclusion, if the master is moving describing a Constant Position trajectory, the best algorithms are the CP, CPCV, CA and CVCA, with accuracies in the order of

4-5 cm. The computing time is not a problem for them. Nonetheless, the particle filter is two orders of magnitude slower than the others.

5.3 Simulation with a Constant Velocity trajectory

The trajectory is generated propagating the following translational kinematic equation:

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} \boldsymbol{I}_{2\times 2} & T\boldsymbol{I}_{2\times 2} \\ \boldsymbol{0}_{2\times 2} & \boldsymbol{I}_{2\times 2} \end{bmatrix} \boldsymbol{x}_k + \begin{bmatrix} \frac{T^2}{2}\boldsymbol{I}_{2\times 2} \\ T\boldsymbol{I}_{2\times 2} \end{bmatrix} \boldsymbol{w}_k$$

The process noise $\boldsymbol{w}_k$ has units of acceleration, so it changes the velocity values. The standard deviations for the process noise in each component are the same and equal to $\sigma_x = \sigma_y = \sqrt{0.001}\frac{m}{s^2}$.

Two different simulations are performed. In the first one, the algorithms are initialized with the correct initial state vector. On the other hand, the second simulation has random initializations for the algorithms. Both simulations are generated with an initial module of the velocity equal to $0.1\frac{m}{s}$, and with the master oriented at $\frac{\pi}{4} rad$. The resulting trajectories, the RMSE errors in the position, and the computing time are presented in Figure 5.3.

It can be seen that all the algorithms reach the convergence in a few iterations, though their initial state vector is wrong. In terms of accuracy in the positioning, the best algorithms are the CV, CA and CPCV with only 2-3 cm of error. But the other algorithms also have a good performance since their error is between 4 and 5 cm. It is important to note that the computing time are approximately equal to the values obtained in the previous simulations.

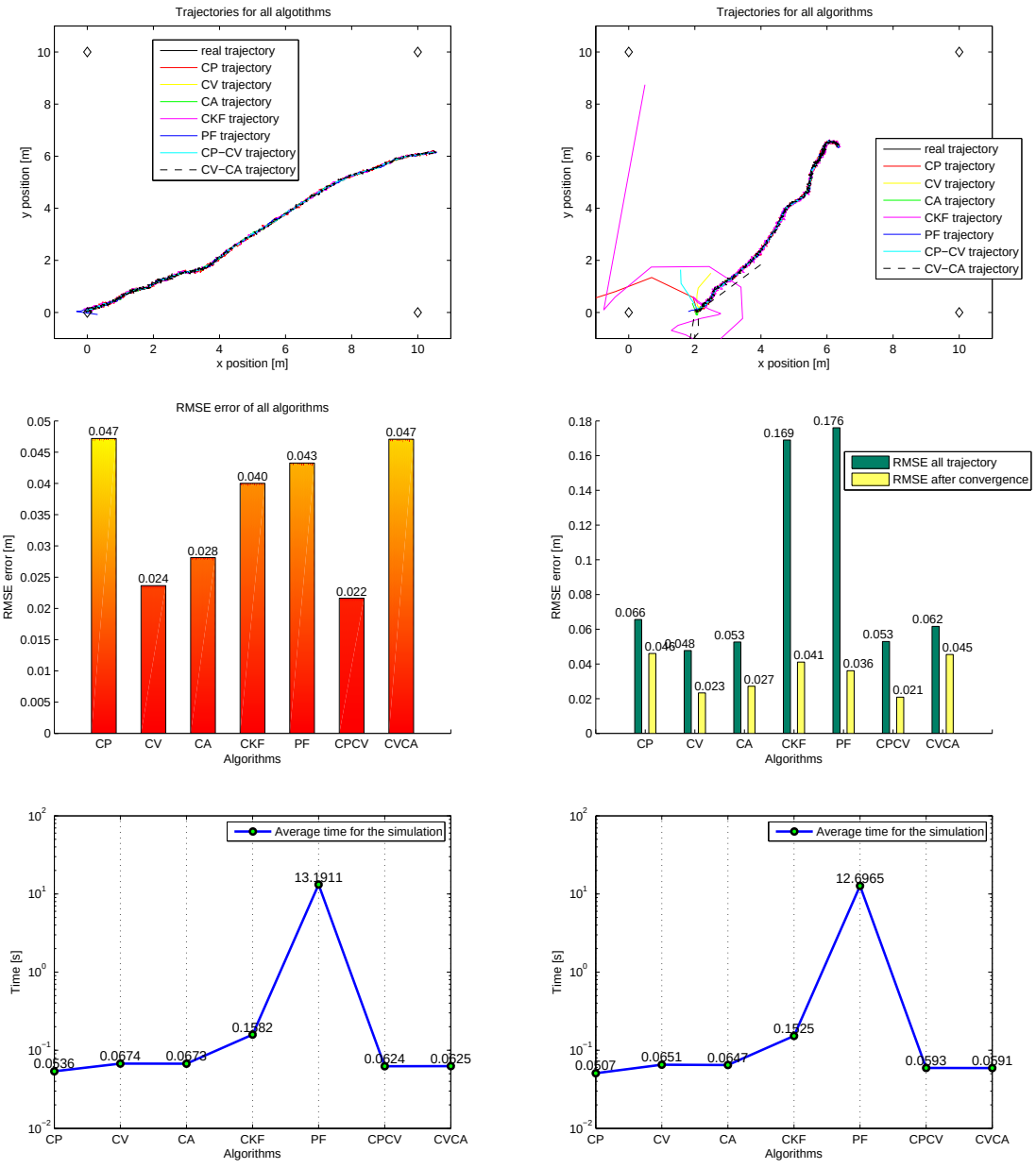

Figure 5.3: On the left, the results for the simulation with correct initialization are presented, and , on the right, with random initialization.

5.4 Simulation with a Constant Acceleration trajectory

The trajectory is governed by the following dynamic equation:

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} \boldsymbol{I}_{2\times2} & T\boldsymbol{I}_{2\times2} & \frac{T^2}{2}\boldsymbol{I}_{2\times2} \\ \boldsymbol{0}_{2\times2} & \boldsymbol{I}_{2\times2} & T\boldsymbol{I}_{2\times2} \\ \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times2} & \boldsymbol{I}_{2\times2} \end{bmatrix} \boldsymbol{x}_k + \begin{bmatrix} \frac{T^3}{6}\boldsymbol{I}_{2\times2} \\ \frac{T^2}{2}\boldsymbol{I}_{2\times2} \\ T\boldsymbol{I}_{2\times2} \end{bmatrix} \boldsymbol{w}_k$$

In this case, the process noise $\boldsymbol{w}_k$ has units of jerk (the derivative of the acceleration), and its effects are changing the acceleration values. The standard deviations of the process noise in each component are $\sigma_x = \sigma_y = \sqrt{0.00001}\frac{m}{s^3}$.

Again two simulations are performed, one with the correct initialization in the state vector and another with random initializations. The master initial velocity is $0.1\frac{m}{s}$, oriented at $\frac{3\pi}{4} rad$. The acceleration in the X component is $0.01\frac{m}{s^2}$ in the first simulation and $0.05\frac{m}{s^2}$ in the second. The accelerations in the Y component are $0.005\frac{m}{s^2}$ and $0.003\frac{m}{s^2}$ respectively. Figure 5.4 presents the results obtained in Matlab.

The results show the convergence of all the algorithms with both initializations, even in the case of the random initial state vector, the convergence is reached in very fem iterations. The obtained errors are a bit lower than the previous simulations. Again the best algorithms are the CV, CA and CPCV with errors between 1.5 and 2.5 cm. The others also present good values between 3 and 4 cm.

The time spent for running the simulations seem to be independent of the analyzed trajectory, since the results are very similar to the values obtained before.

5.5 Simulation with a Spline trajectory

The generation of this trajectory is carried out by a Matlab code which uses the "spline" function. The inputs to the Matlab program are some points of the desired trajectory, the sampling time T (0.1 s), and the time that is necessary for going from one point to the following one (4 s in this case). The "spline" function joins the points with paths characterized by having piecewise linear accelerations.

Once the trajectory is created, and the noisy measurements calculated, two simulations can be performed in the same way that the previous ones: one with a correct initialization and the other with a random initialization. The results are presented in Figure 5.5.

The simulations show the quick convergence of the algorithms. Regarding the RMSE error, the best algorithms are the CA, the CPCV and the PF, however all the errors are small, in the order of 3-4 cm, so it seems that the algorithms have the power to estimate correctly the master position even if the dynamic model of the trajectory

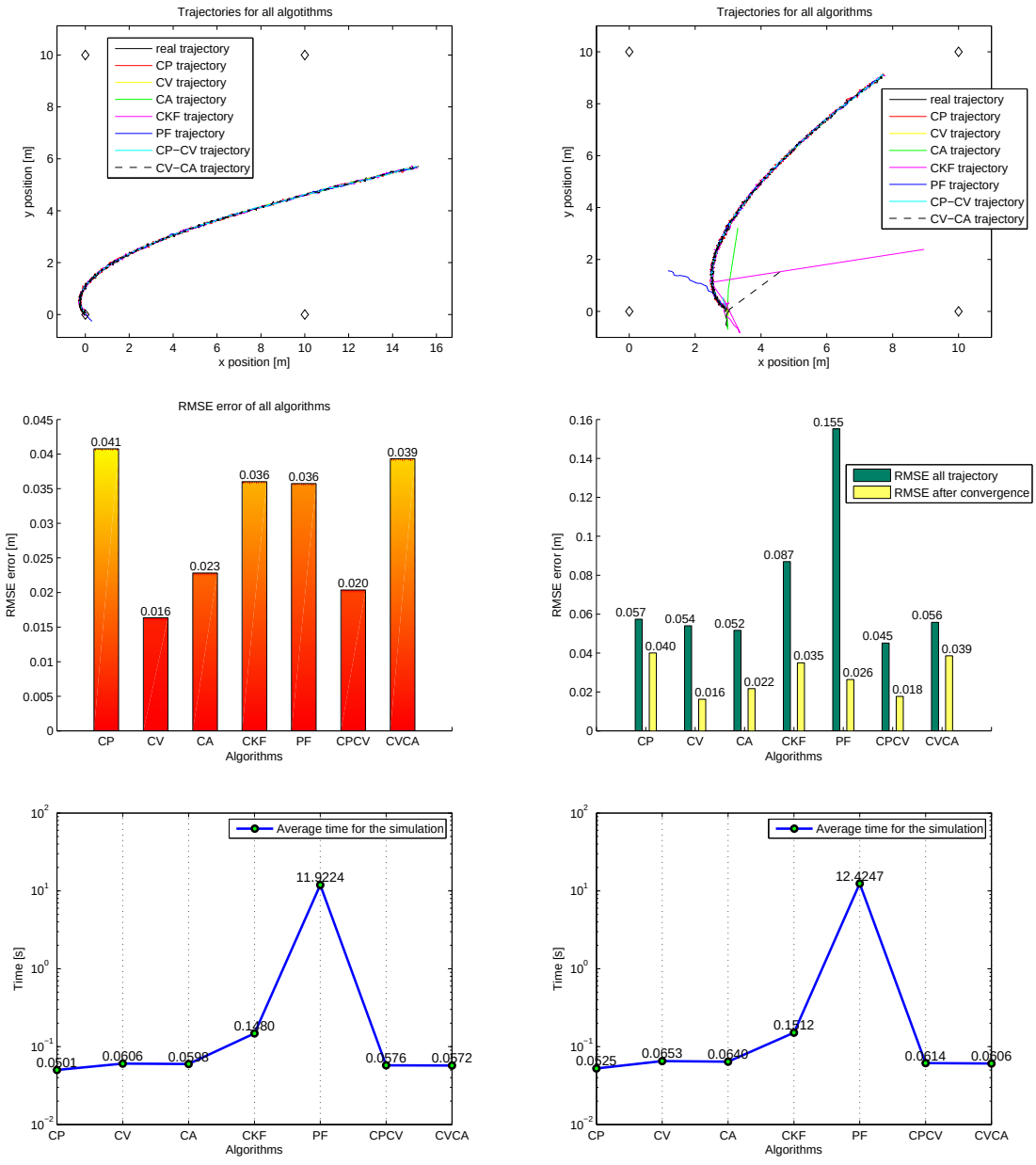

Figure 5.4: Results of simulating the Constant Acceleration trajectories. On the left, correct initialization. On the right, random initialization.

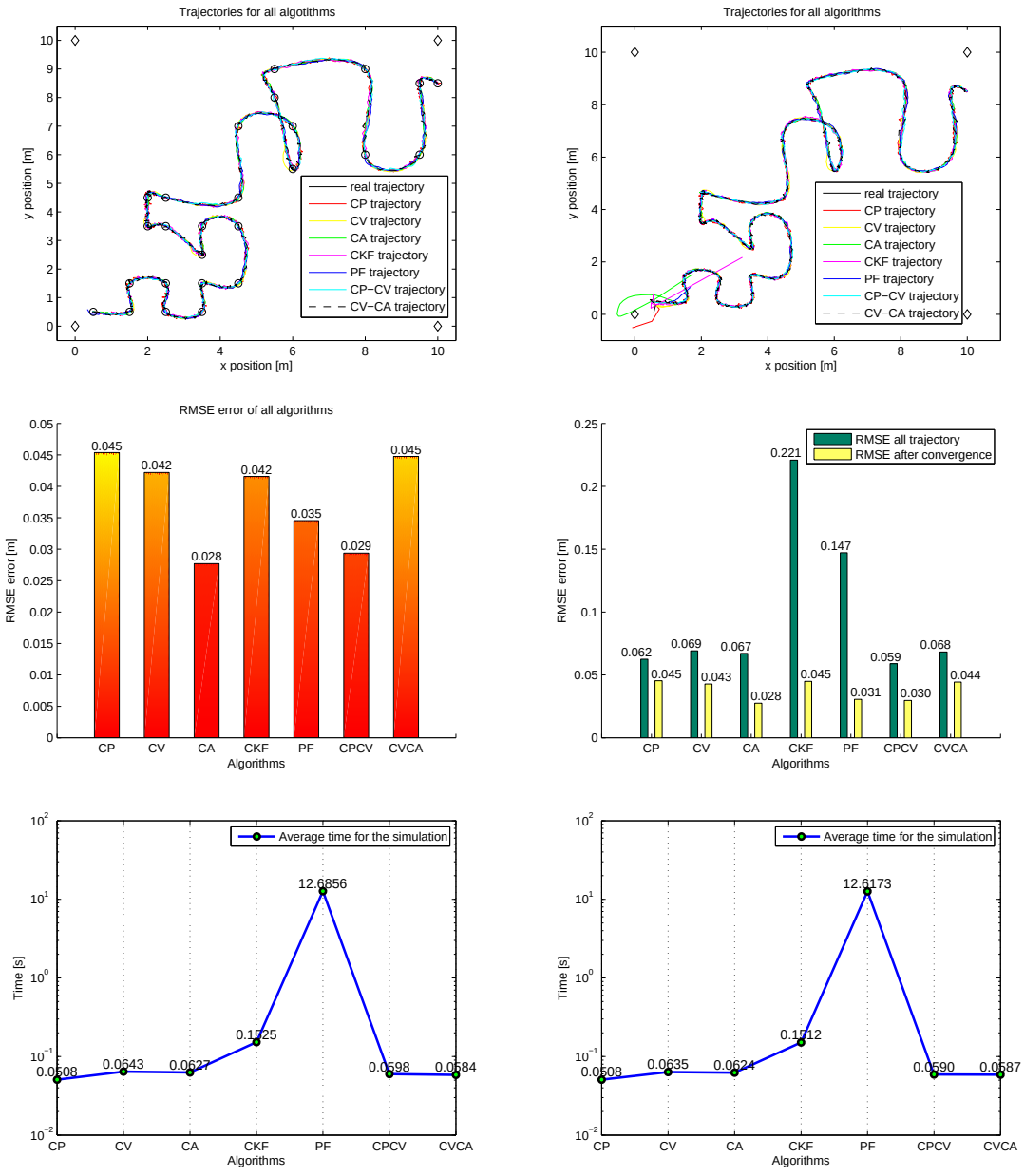

Figure 5.5: Results of simulating the Spline trajectories. On the left, correct initialization. On the right, random initialization.

is none of the models that the algorithms are using. The computing time graphs confirm that the algorithms have no dependence on the trajectory model, since they spend similar times in all the simulations.

5.5.1 Simulation with stand-alone systems

It has been seen that the algorithms are working well with the simulated trajectories. Nevertheless, they have not been compared with the trajectories that can be obtained using the UWB system, the DR system or the INS in a stand-alone configuration. These cases are analyzed with the spline trajectory.

For obtaining the position estimates using only the noisy ranging measurements, the Linear Least Squares (LLS) approach is used [16]. It gives an approximated solution to an overdetermined system of linear equations. The approximation is defined as that which minimizes the sum of squared differences between the measured range values and the estimated ones. The procedure is done for all the time steps, and the resulting trajectory is the one presented in Figure 5.6.

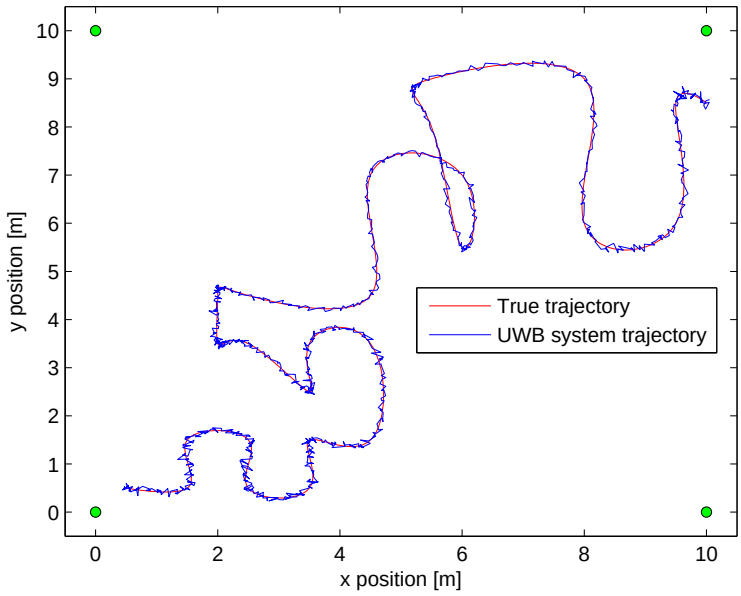

Figure 5.6: Trajectory obtained with the UWB system in stand-alone configuration.

The RMSE error obtained is 6.2 cm. It's a low value because the scenario of the simulations is only taking into account the general errors, but in real scenarios, other sources of error such as interferences can be present. However, all the algorithms has less error than the stand-alone UWB system.

The INS and the DR system can work as autonomous positioning systems, only by

propagating the navigation equations (see Section 2.2.2 for the INS and Section 2.3 for the DR system). The measurements have the same errors and biases used in the previous simulations. The trajectories resulting from the propagation of the navigation equations are shown in Figure 5.7.

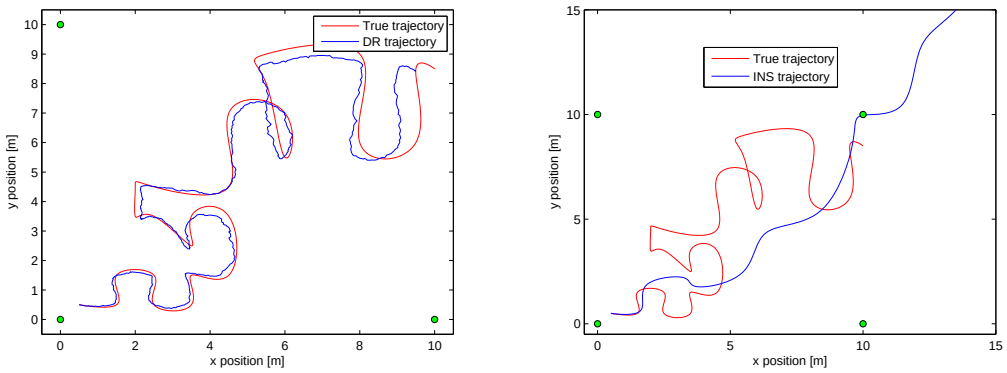

Figure 5.7: The trajectory on the left is the one obtained with the DR system. On the right, there is the INS trajectory.

It is easy to see that the results are not good. The DR system is following the true trajectory, but is starting to diverge. It is logical since the error grows with time. The RMSE error is 60 cm. On the other hand, the INS trajectory is diverging from the beginning, due to the biases and the unbounded growth of the accumulated error. The RMSE error is 31 m.

In conclusion, the developed algorithms are outperforming the results that can be obtained with the stand-alone systems. This results are confirming the advantages of the information fusion described in Section 2.4.

5.5.2 Other navigational states

As noted in Chapter 2, not only the master position can be estimated using these developed fusion algorithms. Other navigational states of the master can be predicted such as the velocity, the acceleration or the orientation. Even the bias in the accelerometer and in the gyroscope can be estimated with the CKF algorithm.

In Figure 5.8, the true values of the velocity in the two components of the local frame are shown. The velocity estimates of some algorithms are superposed, and it can be seen that all the estimates are following the true values, with very little error.

In Figure 5.9, the true values of the acceleration in the local frame are compared to the estimates obtained with the CA algorithm (it is the only one that has the acceleration in the state vector). Again the estimates are very accurate, only the biases are causing some errors. Note that the accelerations are linear due to the use of the "spline" function.

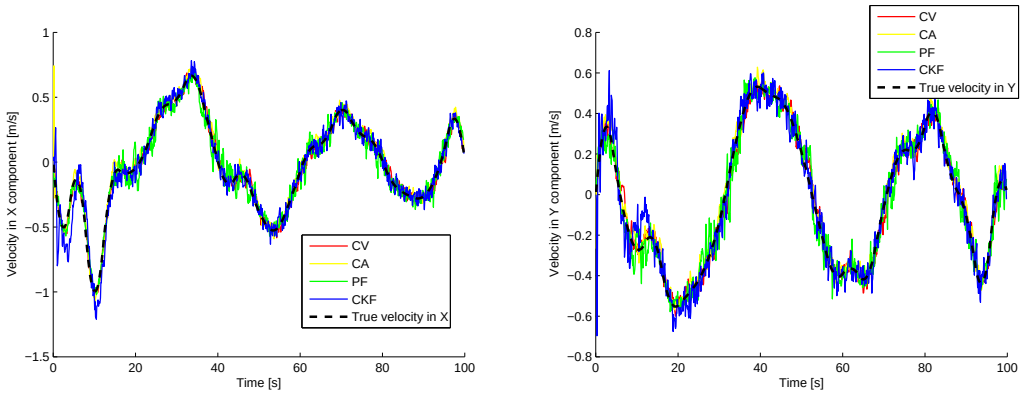

Figure 5.8: True velocities and the estimates obtained by some algorithms. On the left the X component, on the right the Y component.

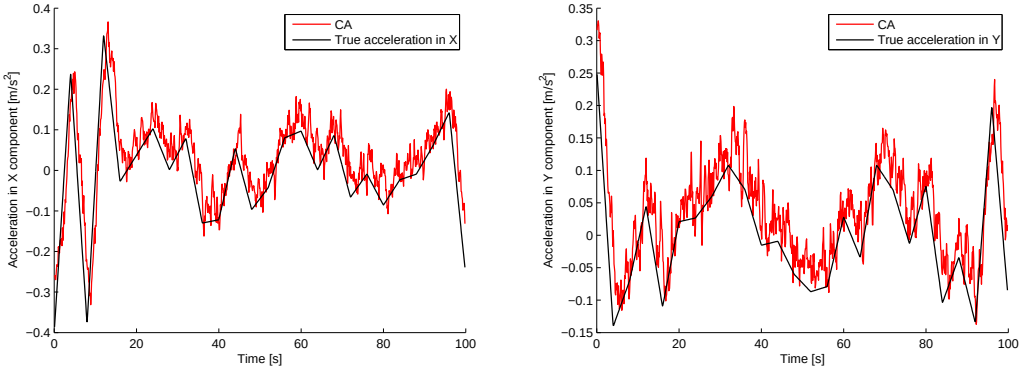

Figure 5.9: True accelerations and the estimate obtained by the CA algorithm. On the left the X component, on the right the Y component.

5.5.3 Shadow areas and sensors with different acquisition rates

As said in Chapter 2, one of the benefits of the sensor fusion algorithms is that when the mobile node goes through a shadow area, in which there are no UWB range measurements available, the INS or DR measurements can be used to track the trajectory. Also the dynamic models which govern the algorithms can help in tracking the position during shadow areas.

Note that in this project, the acquisition rates of the individual sensors are supposed to be equal in all the simulations. Nevertheless, the INS and DR systems usually have a higher rate. In this case the situation is similar to going through a shadow area: the UWB measurements will not be available in all iterations.

The solution in these two situations depend on the algorithm. If the algorithm is using the INS or DR measurements as an input $\boldsymbol{u}$ during the prediction phase, then in the

correction phase only the UWB measurements are needed. When no UWB measurements are available, the correction phase is ignored, letting the a priori estimate (the prediction) be the a posteriori estimate.

The other case occurs in the algorithms that do not use the input $\boldsymbol{u}$, and use all the measurements during the correction phase. One possible solution is to ignore the correction phase, as done in the previous case, but then the information from the measurements is totally lost, while in the previous case, it is used during the prediction phase. In order to avoid the loss of information, another solution can be implemented. It consists on only using the INS or DR measurements during the correction phase, by setting to zero the elements corresponding to the UWB measurements in the innovation.

These solutions have been simulated with the Spline trajectory, and using the CV algorithm (without input) and the CPCV algorithm (with input). Both of them use the UWB range measurements and the DR velocity measurements. In particular, two shadow areas have been introduced in the trajectory. The results are shown in Figure 5.10.

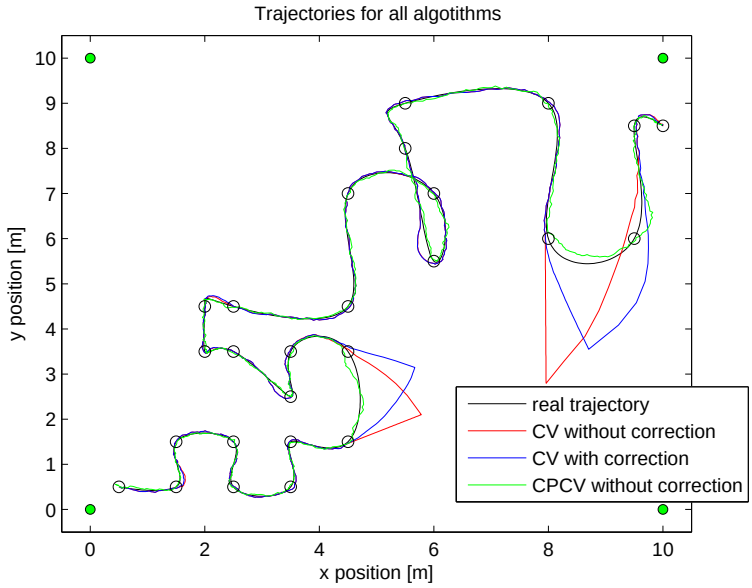

Figure 5.10: Trajectories obtained simulating shadow areas or different sampling rates in the sensors.

Looking at the trajectories, it can be seen that the best results are obtained with the CPCV algorithm, which use the input. Concretely, the RMSE error is 5.2 cm. The CV algorithm has an error of 57 cm without the correction phase, and an error of 42 cm, using the velocity measurements in the corrections.

So the conclusion is that when going through a shadow area or when using information sources which acquire the data at different rates, the best results are obtained with the

algorithms that use the measurements with a higher acquisition rate as inputs, and perform the correction phase only when a UWB range measurement is available. The reason is that this is the best way to take profit of all the available information. Finally, note that the accuracy of the results will depend on the quality of the measurements.

5.6 Results discussion

The RMSE errors and the computing times obtained in the simulations are summarized in Table 5.1.

	C.P. model	C.V. model	C.A. model	Spline
Algorithms	RMSE– Time	RMSE– Time	RMSE– Time	RMSE– Time
CP	4.4 – 0.053	4.6 – 0.051	4.0 – 0.052	4.5 – 0.051
CV	20.9 – 0.068	2.3 – 0.065	1.6 – 0.065	4.3 – 0.063
CA	5.7 – 0.063	2.7 – 0.064	2.2 – 0.064	2.8 – 0.062
CKF	8.2 – 0.157	4.1 – 0.152	3.5 – 0.151	4.5 – 0.151
PF	11.2 –13.063	3.6 –12.696	2.6 –12.423	3.1 –12.617
CPCV	4.4 – 0.064	2.1 – 0.059	1.8 – 0.061	3.0 – 0.059
CVCA	4.4 –0.0624	4.5 – 0.059	3.9 – 0.061	4.4 – 0.058
Range	–	–	–	6.2 –
DR	–	–	–	60 –
INS	–	–	–	31000 –

Table 5.1: Summary of the RMSE errors (cm) and computing times (s) obtained with all the algorithms.

Regarding the RMSE errors, the Constant Position trajectory, which is characterized by a very static master, is giving the worst results. In the others simulated trajectories, all the algorithms are performing rather well. The most constant of them are the CP, CA, CPCV and CVCA, which maintain a low error in all the cases.

The computing times are below the maximum time permitted (they can make all the necessary calculation before new measurements are taken). All of them but the PF algorithm, only need between 50 and 150 ms to perform a simulation of a 100 s trajectory. The PF algorithm, using the Particle Filter recursion, is an special case that needs much more calculation, and spends in the order of 12-13 s, which is a hundred times the duration of the other simulations. In terms of power consumption and algorithm speed, it might be a disadvantage.

On the other hand, it is important to note that all the developed algorithms are outperforming the accuracies obtained with the UWB, DR and INS systems working in a stand-alone configuration. This is one of the benefits of the information fusion. Furthermore, other navigational states are estimated with high precision, as seen in

Section 5.4.2. These good estimates cannot be obtained with the navigation equations used in the INS and DR systems, because errors grow with time, leading to divergence in the estimates.

In conclusion, in a simulated scenario with known and controlled errors, all the algorithms have a good performance. Now the interesting thing is testing the algorithms in a real and uncontrolled scenario, which has more sources of errors such as vibrations or interferences, possible shadow areas, outliers in the measurements, and other unexpected circumstances that may worsen the results. Chapter 6 is dedicated to this testing with real datasets.

Chapter 6

Test with experimental datasets

In this chapter, the developed algorithms are tested with experimental datasets acquired with a mobile robot which performs the master functions, using a commercial UWB positioning system.

6.1 Experimental setup

The aim of this chapter is to evaluate if the fusion of heterogeneous information is useful for positioning applications in two dimensions in real scenarios. In particular, the information integrated comes from the measurements obtained from the odometer and the INS installed on the mobile robot and a Ubisense commercial UWB positioning system [5]. These measurement sets were acquired in a different experimental research project, at the Signal Processing Department of KTH, prior to this Master Thesis project.

The test has been performed in an indoor environment, the R1 reactor hall in KTH (www.r1.kth.se), which can be seen in Figure 6.1. It is a 12×30 m hall, in which a Ubisense commercial UWB positioning system has been installed and calibrated. It is composed of 8 wired sensors (the slaves) mounted in fixed and known positions along the walls of the hall, and several wireless tags which can be placed on mobile nodes (the masters). The sensors measure the time-difference of arrival and angle of arrival of UWB pulses transmitted by the tags. Using these data, the system provides a measure of the position of a tag. The experiments were performed in such a way that the tags were in line of sight from the wired sensors. Note that, although this commercial UWB positioning system gives measures of the position, the developed algorithms use ranging measurements, so the distances between the measured position and the slaves position are calculated in order to emulate the direct range measurements. Besides, only four of the eight range measurements are used in the algorithms.

The ground truth reference (the true trajectory) was provided by a SICK lidar scanner

Figure 6.1: Image of the R1 reactor hall from above.

(LMS200), attached atop the mobile robot, via simultaneous localization and mapping, SLAM. The velocity measurements were acquired using the wheel encoder odometer. The configuration of the SLAM system gives an accuracy of 1 to 10 mm when locked on to two walls (not parallel). During the test, several walls were visible to the robot at all times. Furthermore, the SLAM system enables to make corrections in the odometer.

The acceleration and angular rate measurements were acquired by a 3DM-GX2 IMU of Microstrain® [8], installed on the mobile robot. A picture of it can be seen in Figure 2.3. The IMU is able to provide measurements at a rate of 250 Hz, and in the three axes of the instrument frame which is supposed to be aligned with the body frame of the robot.

The odometer readings are subject to two main sources of error, systematic errors and disturbances. Any bias was not significant since a careful calibration was made. There was a small inherent systematic error that is due to misalignment of the wheels, and the resolution rate of the encoders. These small errors along with the small roughness of the floor can be modeled as Gaussian noise. The larger disturbances were due to wheel slippage, but this problem was avoided by carefully driving the robot.

The IMU measurements are also subject to some sources of errors, as explained in Section 2.2.1. The roughness of the floor provoked constant vibrations that were adding a noise component to the readings. And not only this, the higher bumps caused outliers in the readings. Also an offset in the platform where the IMU was attached caused that the gravity, which in principle only affected the Z component of the readings (which is not used in this 2D positioning), affected the X and Y components of the accelerometer measurements, adding a bias to them. Finally, the inherent sources of error that depend on the quality of the sensor were present: systematic errors and bias. Some of the errors could be attenuated processing the data. The outliers were removed comparing the measurements with a threshold, the gravity components were eliminated by subtracting the bias term which could be observed with the IMU in

a static position (without motion), and the noise component due to vibrations was attenuated with a low-pass filtering.

Before presenting the results of testing the algorithms developed in the project with these datasets, note that the robot setup and the acquisition of the measurements were performed before this project was started. So, the author was not involved in the design and posterior performance. The configuration of the system explained in Chapter 2 was based in this experimental setup. Also the algorithms have been developed taking into account how to take profit of these datasets.

6.2 Testing of the algorithms using UWB and DR measurements

In this section, the results of testing the algorithms which fuse the UWB ranging and DR measurements are presented. In particular, the emulated range measurements are the distances from the estimated position provided by the Ubisense UWB positioning system to the four selected slaves. The velocity and orientation readings are acquired from the wheel encoder odometer.

The algorithms tested are the CP, the CPCV, the CV and the PF. The first three use the EKF recursion, and the last one is based on the Particle Filter method. All of them have been tuned so as to get the best performances. The results are displayed in Figure 6.2. The trajectories obtained from the UWB system and the DR system working in stand-alone configurations have also been calculated, in order to compare them with the trajectories from the fusion algorithms. They are presented in Figure 6.3.

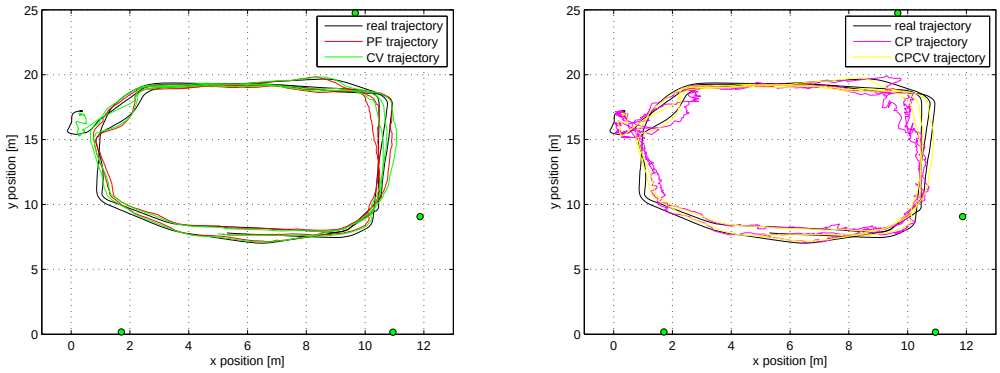

Figure 6.2: Trajectories obtained with the PF and the CV algorithms are presented on the left. Trajectories from the CP and the CPCV algorithms are on the right.

At first sight, it seems that the PF, CV and CPCV algorithms are following the true trajectory more accurately than the CP, which has a behavior very similar to

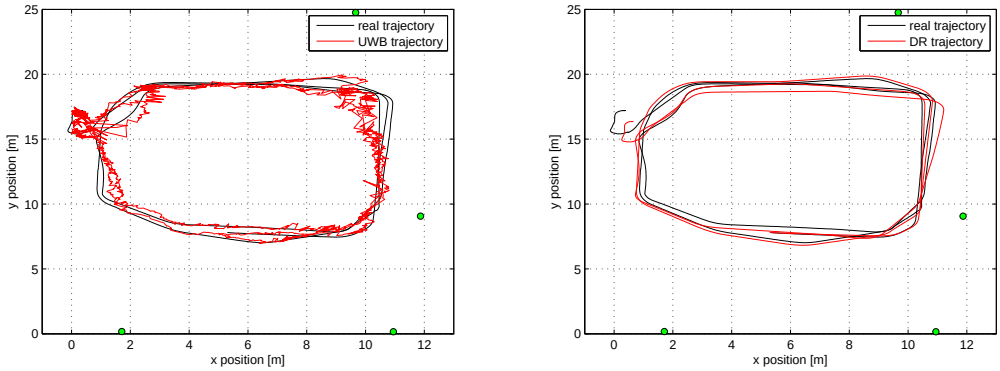

Figure 6.3: The trajectory from the UWB system is shown on the left, and the one from the DR system is shown on the right.

the stand-alone UWB system. On the other hand, the trajectory resulting from the stand-alone DR system worsens as time passes due to the accumulation of errors that cause a rotation of the estimated trajectory. It can be seen also that the UWB range measurements have more error in two of the corners of the trajectory. The possible reason is that the big metallic pipes installed in the R1 reactor hall, are causing very powerful reflections of the signals, and in these two corners the effects are worse. A new calibration of the UWB system would be enough to solve this problem, since the pipes were installed after the installation of the Ubisense UWB positioning system and the first calibration is not taking into account the effects of these pipes. In Figure 6.4, the RMSE errors are presented, and will help to discuss the performance of the algorithms.

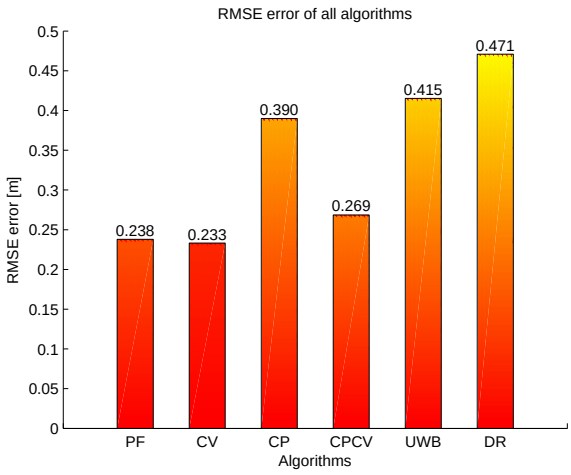

Figure 6.4: RMSE errors for the tested algorithms.

The PF and CV are the best ones, with an error of 23 cm, followed by the CPCV with 27 cm. They are outperforming the stand-alone systems in more than 15 cm,

confirming the good performance that can be achieved with the information fusion algorithms.

6.3 Testing of the algorithms using UWB and INS measurements

In this section, the CA, CACV and CKF algorithms, which fuse the IMU measurements and the UWB ranges, are tested. As said before, the accelerations and angular rates acquired by the IMU have been processed in order to attenuate the effects of vibrations, drifts and outliers. Also the data has been re-sampled at 10 Hz, since the acquisition rate of the IMU is 250 Hz, and the UWB system gives the measurements at 10 HZ. In this way, both measurements can be introduced in the algorithm at a time. However, working with different acquisition rates is possible (as shown in Section 5.??). One option is only perform the prediction phase of the algorithms, where the dynamic model is used, and make no corrections. The other option, which can be performed in the algorithms with inputs $\boldsymbol{u}$, is making the predictions using the dynamic model and the inputs, and only make the corrections when a UWB measurement is acquired.

The resulting trajectories of applying the algorithms to the datasets,and the RMSE errors are presented in Figure 6.5. In both figures, there are also the results from the UWB system in stand-alone configuration. The INS in autonomous configuration has not been represented, since the divergence of the results was so high that it makes no sense to show them.

It can be seen that the errors obtained are in the same order of magnitude as the error from the UWB system. Looking at the trajectories, they are following the UWB system trajectory. The reason is that the filters have been tuned for obtaining the best results, and in this case, the best performance is obtained by trusting the UWB range measurements rather than the acceleration and angular rate measurements or the model.

In conclusion, the IMU measurements are not sufficiently accurate to be trusted in this particular experimental setup. The possible reason is that the effects of the sources of errors (vibrations, drifts...) are too pronounced.

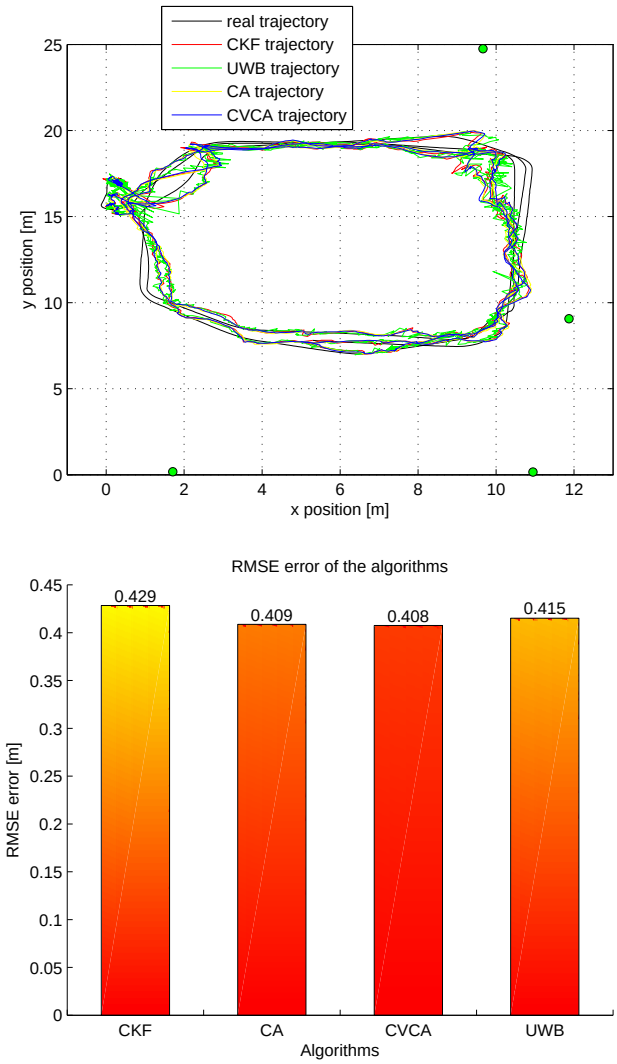

Figure 6.5: Results of testing the CA, CVCA and CKF algorithms together with the results from the UWB system in stand-alone configuration. In the upper graphic the trajectories are shown. In the lower graphic the RMSE errors are shown.

6.4 Results discussion

As seen in the results, only the information fusion algorithms using measurements from the UWB system and the DR system are able to outperform the performances of the stand-alone systems, in more than 15 cm, reaching accuracies of 23-25 cm. In contrast, the fusion of data coming from the UWB system and the INS has not been satisfying. The best performance is achieved by trusting the UWB measurements, as the IMU measurements are not accurate enough.

One reason may be that the designed setup for the mobile robot is more favorable for the use of the wheel encoder odometer and not for the IMU. The vibrations caused by the floor roughness, the drifts and other sources of error have less effect in the case of the odometer.

In conclusion, some of the algorithms can achieve accuracies in the order of 23-25 cm, concretely the PF, CV and CPCV, by means of fusing velocity and orientation measurements with UWB range measurements. Better accuracies could be obtained with a new calibration of the Ubisense UWB positioning system (for solving the problems caused by the reflections), but the mentioned algorithms have been able to follow trajectories that were affected by these errors. The good performance of the fusion algorithms has been demonstrated in case of having reliable sources of information. In contrast, if one of the sources of information is not good enough, the algorithms can only achieve the best performance of the individual systems, which can be taken as a proof of the robustness of these algorithms, since they are able to estimate the position of the master even if some measurements are not reliable.

Chapter 7

Conclusions

The aim of this project was to provide positioning algorithms for indoor navigation applications. For such applications, the requirements were to accurately localize a mobile node in real-time. The solution has been focused on the development, implementation and evaluation of different sensor fusion algorithms. Such algorithms are able to take profit of the benefits of the individual sensors or systems which cooperate in the application, by means of fusing the different nature of the information acquired by each sensor.

In total, seven algorithms have been implemented. Three of them integrate UWB ranging measurements with velocity and orientation measurements from a DR system. Another three, fuse the ranging measurements with acceleration and angular rate measurements from an INS. The last one, is a special case that only uses ranging measurements. The transition equations which model the predictions of the algorithms have been adapted to different dynamic models, depending on the navigational states that are intended to be estimated. Furthermore, this diversity increases the possibilities to find a model which fits properly with the motion of a mobile node.

The results of evaluating the behavior of the algorithms in Matlab simulations with diverse trajectories have shown that all the algorithms, in the general case, outperformed the performances of the individual systems working in stand-alone configuration, in terms of RMSE error. Besides, not only the position could be estimated in an accurate way, also other navigational states such as the velocity or the acceleration have been estimated with good results.

Concerning the specification of being algorithms which can be used in real time applications, the results have shown that all of them could make all the required calculations within the time between two consecutive acquisitions of measurements. Moreover, the simulations with diverse trajectories have also indicated that the computing time does not depend on the data, only on the algorithm. These results confirm that the use of the developed algorithms in real time applications is feasible.

After that, a new challenge was addressed. Testing the algorithms with real datasets taken from an experimental setup, in which the sources of errors could not be known and the algorithms had to be tuned in order to obtain the best performances. These datasets were acquired by a mobile robot in a different experimental research prior to the development of this Master Thesis project. So, the algorithms were developed taking into account the posterior tests with this real data.

The tests results have shown that the algorithms which fused measurements from the UWB system and the DR system, were able to outperform by more than 15 cm the accuracies obtained with the individual systems. Concretely, accuracies in the order of 23-25 cm were achieved. On the other hand, the algorithms which integrated measurements from the UWB system and the INS, obtained results similar to the stand-alone UWB system. The reason was that the INS measurements were not sufficiently reliable due to the errors produced by vibrations, bumps in the floor, and biases. Surely the reason was that the experimental setup of the mobile robot was more favorable for the use of a DR system and not for an INS.

However, as said, the algorithms have been able to match the results of the UWB system, which was the best of the individual systems. This was the confirmation of the robustness of the developed sensor fusion algorithms, which are capable of discarding the non-reliable measurements and estimate the position and the other navigational states only using the useful information.

7.1 Further work

The possibility of testing the algorithms in a real time experiment, as the one performed with the mobile robot, would be a very interesting way of proving the capabilities of the work developed in this project.

A further development related to this project would be to improve the algorithms including the adaptability to different circumstances and requirements. This means that the algorithms would be able to adapt their dynamic models to the motion of the mobile node, and tune their parameters automatically in order to achieve the best performances.

These algorithms have been developed to be used in 2D indoor positioning applications. Nevertheless, increasing the dimensionality to three dimensions would provide a solution to all classes of positioning applications. This would be another important improvement for the work done in this project.

List of Figures

List of Tables

Bibliography

[1] A. De Angelis, J. Nilsson, I. Skog, P. Händel, and P. Carbone, "Indoor Positioning by Ultrawide Band Radio Aided Inertial Navigation," *Metrology and Measurement Systems*, vol. 17, no. 3, pp. 447–460, 2010.

[2] A. De Angelis and C. Fischione, "A distributed information fusion method for localization based on pareto optimization," *Proceedings of the IEEE International Conference on Distributed Computing in Sensor Systems*, 2011.

[3] J. O. Nilsson, *Navigation System for a Micro-UAV*. Master thesis, KTH, Stockholm, Sweden, 2008.

[4] "Revision of part 15 of the commission's rules regarding ultra-wideband transmission systems," 2002. Report and order, in FCC 02 48, Apr. 2002.

[5] Ubisense Ltd., "The Ubisense Precise Real-time Location System - Series 7000 Sensor." [Online]. Available: http://www.ubisense.net/, 2011.

[6] J. A. Farrell, *Aided Navigation. GPS with High Rate Sensors*. McGraw-Hill, 2008.

[7] K. J. Walchko and P. A. C. Mason, "Inertial navigation," *Florida Conference on Recent Advances in Robotics*, 2002.

[8] "Microstrain 3dm-gx2 gyro enhanced orientation sensor data sheet," July 2007.

[9] G. Welch and G. Bishop, "An introduction to the kalman filter," *UNC-Chapel Hill, Technical Report 95-041*, July 24 2006.

[10] F. Gustafsson, *Statistical sensor fusion*. Studentliteratur, 2010.

[11] R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering, Second Edition*. John Wiley & Sons, 1992.

[12] M. Sanjeev Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, February 2002.

[13] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," *IEE PROCEEDINGS-F*, vol. 140, April 1993.

[14] F. Gustaffson, F. Gunnarsson, N. Bergman, U. Forssell, R. Jansson, J.and Karlsson, and P. Nordlund, "Particle filters for positioning, navigation and tracking," *IEEE Transactions on Signal Processing, Technical Report 2333*, February 2001.

[15] J. Nilsson, A. De Angelis, I. Skog, P. Carbone, and P. Händel, "Signal Processing Issues in Indoor Positioning by Ultra Wideband Radio Aided Inertial Navigation," in *17th European Signal Processing Conference, EUSIPCO*, (Glasgow, Scotland), August 2009.

[16] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks, Chapter 9 – Localization and positioning.* Wiley, 2005.