Thesis for the degree of
Master of Science in Design of Telecommunication Systems

A VEHICLE TRACKING SYSTEM BASED ON WSN

Albert Anglès Vázquez

Advisors: Jose López Vicario and Antoni Morell Pérez

Department of Telecommunications and Systems Engineering

Universitat Autònoma de Barcelona

Bellaterra, September 13, 2010

Abstract

This master thesis is covered under the XALOC project approved by the Autonomous Government of Catalonia under the INFOREGIO 2009 program. The purpose of this project is to develop a vehicle localization and tracking system based on a Wireless Sensor Network which is used to guide the driver to free parking slots. Every sensor with reference coordinates is placed in each parking slot. The development of the project consists of two parts:

1. Theoretical development and validation: this part evaluates the positioning as well as some of tracking algorithms by means of a developed Matlab simulator. The simulator is implemented with graphical user interface to allow the possibility of simulating by modifying some specified system control parameters. The simulator simulates several predefined car routes in an urban region as the centre of Barcelona. Real-time vehicle positioning as well as tracking is carried on when the car is moving. The vehicle positioning is obtained by means of the RSSI of a specified number of reference sensors and a multilateration technique (similar to the used in GPS representing the reference nodes as satellites). The vehicle tracking is realized with the following tracking algorithms: Kalman Filter that follows a uniform motion model, Extended Kalman Filter that follows the turns and a weighted combination of both for the overall route with the IMM algorithm. Theoretical analysis is realized to validate these algorithms for several scenarios.

2. Experimental development and validation: the experimental development deals with the outdoor real-time positioning and tracking of a mobile node carried inside a car. For that a Java-based navigator called **ARID Navigator** is implemented. This navigator shows the real-time position of the driver on the map of the measurement scenario. The measurement scenario is located at the Autonomous University of Barcelona's fire department parking and a measurements campaign is realized to adjust certain navigator's parameters. These parameters are mainly the Kalman parameters. In addition to the driver localization and tracking, the navigator informs the driver either with graphical or audio interface the number of free parking slots. Concerning the positioning technique for noisy environments another improved technique in terms of average position error than the applied in the theoretical study is developed.

Resum

Aquesta tesis de màster està coberta sota el projecte XALOC aprovat pel govern autònom de Catalunya sota el programa INFOREGIÓ/AJUTS 2009. L'objectiu d'aquest projecte és desenvolupar un sistema de seguiment i de localització de vehicles basat en una xarxa wireless de sensors la qual s'utilitza per tal de guiar al conductor a trobar aparcaments lliures en zones urbanes. Cada sensor amb coordenades de referència es localitza en cada plaça d'aparcament. El desenvolupament del projecte consisteix en les següents parts:

1. Desenvolupament teòric i validació: aquesta part evalua les tècniques de posicionament i seguiment de vehicles miatjançant un simulador de Matlab desenvolupat sota una interfície gràfica d'usuari per facil·litar la simulaciò amb diferents paràmetres. El simulador simula diferents rutes del coche en una regió urbana com la del centre de Barcelona. El posicionament del vehicle a temps real així com el seguiment és dut a terme. Pel que fa al posicionament, aquest s'obté a partir de les RSSI d'un nombre especificat de sensors de referència amb una tècnica de posicionament similar a la de GPS. Pel que fa al seguiment del vehicle aquest es realitza mitjançant certs algoritmes de seguiment: en concret un Kalman Filter que s'utilitza per seguir les trajectòries en moviment uniforme, un Extended Kalman Filter que segueix els girs del vehicle, i una combinació ponderada dels dos a través d'un novedós algoritme anomenat IMM (Interacting Multiple Model). Un anàlisis teòric és realitzat per tal de validar aquests algoritmes per diferents escenaris.

2. Desenvolupament experimental i validació: aquesta part té per objectiu el desenvolupament d'un navegador anomenat **ARID Navigator** basat en Java. Aquest navegador mostra la posició del conductor a temps real sobre un mapa de la zona. L'escenari de proves es troba al parking de bombers de la Universitat Autònoma de Barcelona i una campanya de mesures s'ha realitzat per tal d'ajustar certs paràmetres del navegador. Aquests paràmetres impliquen paràmetres del Filtre de Kalman ja que aquest és utilitzat per seguir la trajectòria del del vehicle en línea recta. A més a més de la localització del conductor, el navegador informa el nombre d'aparcaments lliures de manera gràfica sobre el mapa o auditiva. Pel que fa a la localització s'ha empleat un mètode millor que el que s'ha fet servir en la validació teòrica quan l'error de posicionament és gran.

Resumen

Ésta tesis de máster esta cubierta bajo el proyecto XALOC aprovado por el gobierno autónomo de Cataluña bajo el programa INFOREGIÓ/AYUDAS 2009. El objetivo de este proyecto es el desarrollo de un sistema de seguimiento y localización de vehículos basado en una red wireless de sensores la cual se utiliza para guiar al conductor a encontrar parkings libres en zonas urbanas. Cada sensor con coordenadas de referencia se localiza en cada plaza de parking. El desarrollo del proyecto consiste en las siguientes partes:

1. Desarrollo teórico y validación: esta parte evalua las técnicas de posicionamiento y de seguimiento de vehículos mediante una interfaz gráfica para facilitar la simulación con diferentes parámetros del sistema. El simulador simula diferentes rutas de un coche en una región urbana como la del centro de Barcelona. Se ha llevado a cabo el posicionamiento del vehículo a tiempo real así como el seguimiento del mismo. El posicionamiento del vehículo se obtiene a partir de las RSSI de un nombre especificado de sensores de referéncia con una técnica de posicionamiento similar a la de GPS. En cuanto al seguimiento del vehículo, este se realiza mediante algoritmos de seguimiento: en concreto un Filtro de Kalman que se usa para seguir trayectorias que siguen un modelo uniforme, un Extended Kalman Filter que sigue los giros del vehículo, así como una combinación ponderada de los dos mediante un novedoso algoritmo conocido como IMM (Interacting Multiple Model). Un análisis teórico es realizado para validar estos algoritmos para diferentes escenarios.

2. Desarrollo experimental y validación: esta parte tiene por objetivo el desarrollo de un navegador llamado **ARID Navigator** basado en el lenguage de programación Java. Este navegador muestra la posición del conductor a tiempo real sobre un mapa de la zona. El escenario de pruebas se encuentra en el parking de bomberos de la UAB y una campaña de medidas se ha realizado por tal de ajustar ciertos parámetros del navegador. Estos parámetros implican parámetros del Filtro de Kalman ya que este es utilizado para seguir la trayectória del vehículo en línea recta. Además de la localización y seguimiento del conductor, el navegador informa del número de aparcamientos libres de manera gráfica sobre el mapa o auditiva. Para la localización, se ha usado un nuevo método más robusto que el utilizado en la validación teórica cuando el error de posicionamiento es grande.

Dedicated to my dear nearest family including my 2 years nephew Damià. Also I want to dedicate this work to my beautiful Mexican friend, Saydee as well as to my flat mates Albert Arajol, Marta Alba, Roser Moreno who have allowed me to enjoy with them during the work on this project. Also I dedicate this work to other friends: Marti Manyosas, Paresh Saxena, Smrati Gupta, Laia Manyosas, Iván Aguilar, Ángel, Carlos Cisneros, Joan Aguilar, Luís Mota, Marc, Sílvia Rodríguez, Eli Pulido, Alejandro and Susana couple and of course to Sónia Cárdenas. Finally I dedicate this project to Rosa (from Llampàies), Susana (my beautiful nurse friend) and Meritxell Pedraza, Ovi, Toni, Simó and Kim who I will never forget them.

Acknowledgments

I appreciate first to my project guides: Jose López Vicario and Antoni Morell Pérez who gave me much support to carry on this Master project. Also I thank to both Albert Bel who participated in the practical section of my project and to Jose Antonio Del Peral who tolerated my presence during at least long 6 months at the UAB's Telecommunications Engineering lab.

Thanks to WorldSensing Company which has led the XALOC project.

Contents

List of Figures

List of Tables

Notation

In the sequel, matrices are indicated by uppercase boldface letters, vectors are indicated by lowercase boldface letters, and scalars are indicated by italics letters. Other specific notation has been introduced as follows:

$\sim \mathcal{N}(\mu_s, \sigma_s^2)$	A certain random value distributed with a Gaussian probability density function with mean μ_s and variance σ_s^2
$\mathcal{N}(\bar{\mathbf{x}}, P)$	A certain random vector $\mathbf{x}$ distributed with a Gaussian probability density function with mean $\bar{\mathbf{x}}$ and covariance P
$'$	Transposition (of a matrix or a vector)
$\sim$	Distributed as
$\underset{x,y}{\operatorname{argmin}}$	Arguments (x, y) that minimizes the respective cost function
$E[\cdot]$	Expectation of $\cdot$
$\mathbf{x}$	State vector
$\hat{x}$	Estimate of x
$\mathbf{F_{KF}}, \mathbf{H_{KF}}$	State Kalman Filter Transition Matrix and Kalman Filter Measurement Matrix
$\mathbf{F_{EKF}}, \mathbf{H_{EKF}}$	Jacobian of the Extended Kalman Filter transition matrix and the Extended Kalman Filter Observation Matrix
$\mathbf{S}$	Innovation covariance, Residual covariance, error covariance
Ω	Turn rate in $/sg$ applied in the coordinated turn model
$\sigma_z, \sigma_u, \sigma_\Omega$	Standards deviation of the position measurements, process acceleration noise and turn rate
$\mathbf{z}$	Obtained measurements which are given to the KF, EKF or IMM filters
$\mathbf{u}[n] = \mathbf{\Gamma v}, \mathbf{w}[n]$	State or process noise vector and Observation noise vector

$\mathbf{Q}, \mathbf{R}$	Process (state) noise covariance and Observation noise covariance
$\mathbf{U}$	Process acceleration noise covariance matrix
$P(k\|j)$	Conditional covariance matrix of state at time k given observations through time j
T	Sampling interval
∇_x	Gradient with respect to the vector $\mathbf{x}$
$\pi_{\mathbf{CT}}$	Mode transition probability matrix
Λ	likelihood function
σ^2_{shad}	Shadowing noise power
γ_{rssi}	Path loss exponent used to obtain values of RSSI from the received power model
γ_d	Path loss exponent used in the distance computation from the received power model
T_a	Sensor activation time in s
$\mathbf{I}$	Identity matrix

Acronyms

3G	Third Generation
WSN	Wireless Sensor Networks
GPS	Global Positioning System
BS	Base Station
IP	Internet Protocol
GN	Gathering Node
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
GSM	Global System for Mobile
RSSI	Received Signal Strength Indicator
RMSE	Root Mean Square Error
DV	Distance Vector
LOS	Line Of Sight
NLOS	Non Line Of Sight
ToF	Time Of Flight
UWB	Ultra-WideBand
WNLS	Weighted Non Linear Least Squares
MDS	Multilateration Scaling
MLE	Maximum Likelihood Estimator
ISM	Industrial,Scientific and Medical
MAC	Medium Access Control
FSSS	Frequency Hopping Spread Spectrum
DSSS	Direct Sequence Spread Spectrum

GFSK	Gaussian Frequency Shift Keying
DQPSK	Diferential Offset Phase shift Keying
LLE	Local Linear Embedding
IM	Iterative Multilateration
KF	Kalman Filter
EKF	Extended Kalman Filter
UKF	Unscented Kalman Filter
IMM	Interacting Multiple Model
WGN	White Gaussian Noise
AR	Auto Regressive
WSS	Wight Sense Stationary
MMSE	Minimum Mean Square Estimator
LMMSE	Linear Minimum Mean Square Estimator
CT-Model	Coordinated Turn Model
ATC	Air Traffic Control
VTC	Vehicle Traffic Control
GUI	Graphical User Interface
IDE	Integrated Development Environment
UTM	Universal Traverse Mercator
s.t.d	standard deviation
WAPM	Weighted Average Power Method

Chapter 1

Introduction

1.1 Motivation

The actual tendency of wireless communications focuses on the design and construction of small sensors that are able to sense and communicate with each other via wireless. This is a very hot topic of research since hundreds of small sensors can be exploited in a large field to obtain remotely real time information as well as monitoring. Generally a sensor has the capacity to sense one or several physical magnitudes (i.e. temperature, pressure, force, magnetism). As a sensor is aware of the physical magnitude several sensors forming a sensor network can be useful in several fields. For example some applications of WSN (*Wireless Sensor Networks*) can be useful in the following fields:

1. In geology: the monitoring of tectonic plates,the prediction of either a volcanic eruption or an avalanche.
2. In medicine: the monitoring of patient's heartbeat or the localization of a patient in a large hospital
3. In logistics: the tracking and monitoring of the transport packages.

Another application where the WSN can be useful is the localization of free parking slots in urban areas. Currently in a big town it is quite difficult for the users to find available outdoor parking places because there is not information to guide the user to find an available parking slot. The consequence of this fact is the high amount of CO2 pollution launched to the atmosphere and therefore contributing to an increase of the global world contamination. The idea to built an efficient car parking management system leads the following advantages:

1. Less time for the users to find an available parking lot.

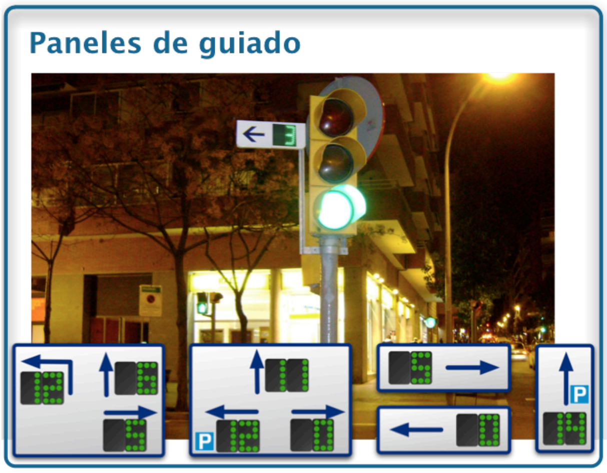


Figure 1.1: Guiding panels.

2. Reduction of the fuel consumption and a decrease of the CO2 pollution.

3. Automatic management in non-free car parking (user must pay). In this application the price associated to the parking time of each car can be automatically computed from a central site.

4. An efficient car parking management system can provide a reduction of the traffic in a certain area as the traffic is distributed and it avoids bottlenecks.

5. As the technology used in these applications is a Wireless Sensor Network, positioning of the parked cars and also real-time positioning or tracking of users that are looking for free parking places can be carried on to drive the user to those known coordinates stored in the central site. Therefore geographical routing known as navigation in GPS can be exploited in real-time.

The motivation of this work is to get into the interesting world of WSN with a new application useful in outdoor car management systems that uses a WSN to track and to guide the driver to find a free parking slot. This application allows a user to find a free parking spots in a certain urban area through a set of indications shown in panels on the street. In order to guide the driver its real time position is needed. This application uses some positioning strategies which are introduced briefly in chapter 2.

As an example, figure 1.1 shows a way to guide the users and the information of available parking slots in each direction. It is useful for the users to find free parking slots if information is provided to the driver.

1.2 Objectives

The objectives of this Master thesis are majority covered by the objectives of the XALOC project which is an e-infrastructure project in the framework of the INFOREGIO program, funded by the Autonomous Government of Catalonia. The mission of this project which is led by the company WorldSensing is to develop a platform based on a WSN capable of detecting outdoor free parking spaces and locating vehicles that carry an on board sensor. The localization information as well as tracking makes possible to guide the drivers to free parking spaces in the area of interest by means of panels on the road or notifications to mobile terminals. The goal of XALOC project is to reduce pollutant emissions thus contributing to a more sustainable development of cities. The work for the Master project has two parts: theoretical and practical:

- The theoretical part consists on the study and analysis of tracking algorithms with Matlab simulations that have been carried on with a developed graphical user interface.
- The practical part consists of a development of a navigator based in Java. Several outdoor measurements are performed to adjust certain parameters in order to track the car that moves along a road surrounded by sensors. The tracking of the car is done with a simple tracking algorithm.

Below the objectives of this work are summarized:

1. Introduction to the Wireless Sensor Networks.
2. Introduction to the current positioning techniques for WSN.
3. Introduction to tracking algorithms.
4. Implementation of tracking algorithms for the target tracking.
5. Simulations and discussions for different system parameters.
6. Implementation of an experimental test-bed consisting of a WSN deployed along a road and a car that is moving through the road. Furthermore, a Java-based navigator is developed in order to show the measured real time position of the driver over a map by means of a positioning strategy based on RSSI (*Received Signal Strength Indicator*) and a tracking algorithm.

1.3 Document Organization

This Master thesis is organized as follows:

1. Chapter 2 is a state of the art in the field of positioning techniques and tracking algorithms for WSN.

2. Chapter 3 contains the implementation of the set of algorithms with their corresponding models.

3. Chapter 4 deals with several simulations to analyse the performance of the tracking algorithms for different cases. To make easier the management of the software a GUI (Graphical User Interface) is implemented in Matlab.

4. Chapter 5 introduces the design of an implemented Java-based navigator. From the received power of a number of specified sensors the navigator computes the real-time coordinates and draws these over map of the measurement geographical area. In addition to the vehicle localization, the navigator gives information about the free and non free parking places which are obtained from an external database server. Several outdoor measurements are carried on to find the optimal system parameters that allows to obtain an accurate and valid navigator for the measured scenario.

5. Chapter 6 gives the conclusions of all the carried work as well as the future work with the intention to follow.

Chapter 2

State of Art

The purpose of this chapter is to give to the reader, a brief introduction about the amazing world of WSN. These networks can be large with hundreds or even thousands of event-driven sensors that are placed in remotely regions. Therefore it is needed to localize those sensor nodes automatically activated by a certain detected event. Hence it means that the coordinates of that sensor must be discovered. An application example can be the detection and tracking of a target that enters the sensing range and moves through the sensor field. This chapter is devoted to provide the reader some of the current localization techniques and tracking algorithms.

2.1 Introduction to Wireless Sensor Networks

Nowadays WSN have become a hot topic for many research entities focused in many real-time applications such as detection, mobile sensors tracking, localization of events, etc. A WSN is a kind of Ad-Hoc network with a set of autonomous nodes which are energy-constrained and interconnected through wireless links. Many research studies are carried on to discover energy-efficient algorithms that might run in a WSN [17], [18], [19] since sensor nodes are able to carry some processing. This philosophy is seen in distributed WSN where the processing is shared among several sensor nodes. Thus it avoids the need of having a single processing unit usually known as FC (*Fusion Center*). Figure 2.1 shows an example of a centralized WSN with the sink acting as a FC and the sensor nodes reaching the FC through multihop. Figure 2.2 shows another example of a distributed WSN. In this figure several clouds known as nodes clusters can be seen. In each cluster a cluster head (or leader node) is chosen to carry some processing. The results computed by each cluster head are sent to the application node through the gateway node connected to an IP(*Internet Protocol*)-network.

The architecture of a WSN can be described by following elements [2] [20]:

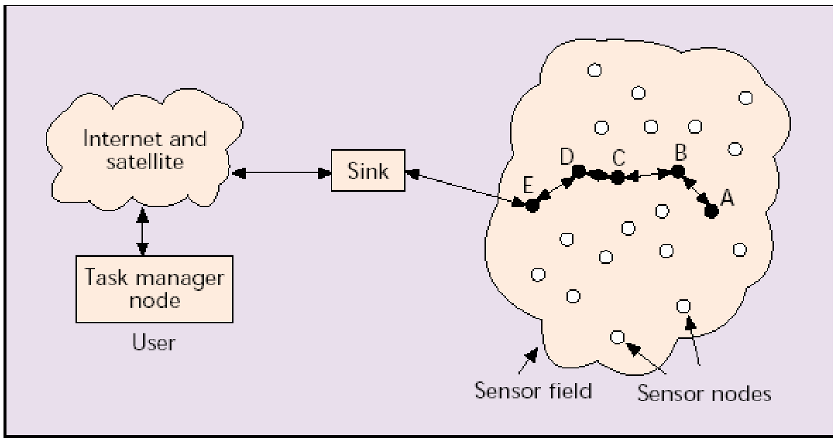


Figure 2.1: A centralized WSN architecture [1].

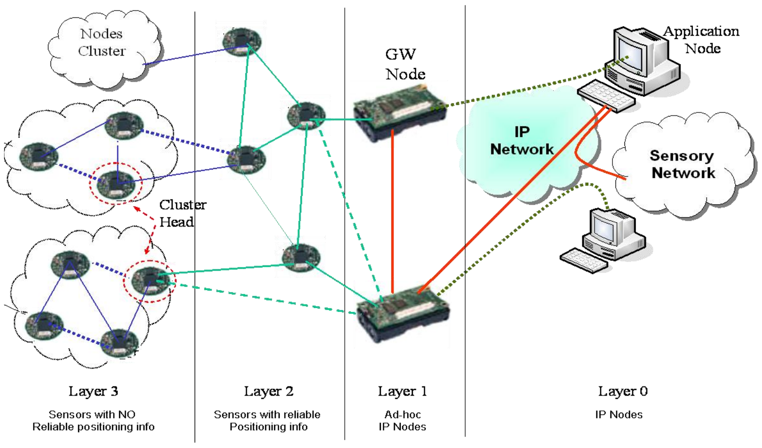


Figure 2.2: A distributed WSN architecture [2].

- **Sink nodes** are ad-hoc IP nodes with enough computational skills and enough energy to allow a wired/wireless communication interface to other TCP(Transport Control Protocol)/IP data networks.

- **Sensor nodes** as the shown in figure 2.3 consist by a set of sensing, processing, communication, actuation, and power units integrated on a single or multiple boards and packaged in a few cubic inches. These nodes are energy constrained because they are powered by small batteries such as AA,AAA or watch batteries. However they provide few kilobytes of memory and a low speed processor unit. These kind of nodes can be classified in two categories depending on its function:

  – Sensors that provide reliable positioning information by the use of a GPS (*Global Positioning system*) or already known their coordinates. In the literature these sensors are referred as anchor nodes, beacon nodes, known nodes or reference nodes. If these sensors have a GPS receiver then they must be powered by more powerful batteries due to the fact that a GPS receiver requires a lot of energy consumption for the processing. In the example of figure 2.2 these sensors appear at Layer 2.

– Sensors that do not have reliable geographic location information and they are not equipped with a GPS receiver or a similar positioning device. Due to that precise geographic coordinates are not available, these nodes must be able to estimate their relative position by the use of some appropriate localization technique. In the literature these sensors are called unknown sensors, blind sensors or just sensor nodes. In the example of figure 2.2 these sensors appear at Layer 3.

Figure 2.3: Crossbow IRIS sensor node [3].

- **Base nodes** which are responsible to gather information from other sensors and to send it to a central processing unit for the processing tasks. The gathered data can be stored either in a local or an external database. These kinds of nodes are needed in a centralized architecture as the shown in figure 2.4.

The drawback of a centralized approach is the communication bottleneck at and near the central processor [5]. A mitigation of this issue is the use of a distributed or decentralized scheme where the processing tasks are not only performed at the BS node but the processing is shared to other nodes. In the distributed approach the whole sensor network is divided into sets of sensors. Each set forming a cluster contains a *cluster head* or *leader node* which is able to perform processing tasks for its cluster in addition to the processing tasks performed at the FC [21].
On the other hand, as each sensor node has a processing unit it can be able to perform some small processing. If all the sensor nodes of the network participate in the processing a cluster-based architecture may not be needed. However most of the found literature consider the cluster-approach a good choice [22] [14] [23], specially for distributed target tracking. A comparative

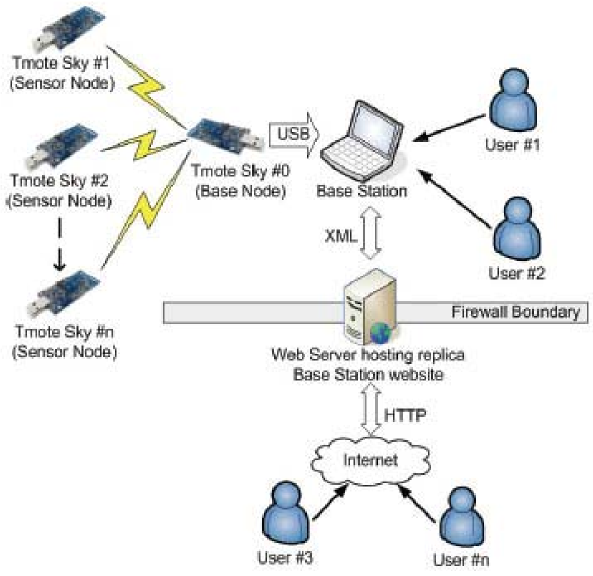

Tmote Sky #1 (Sensor Node)
Tmote Sky #2 (Sensor Node)
Tmote Sky #n (Sensor Node)
Tmote Sky #0 (Base Node)
USB
Base Station
User #1
User #2
XML
Firewall Boundary
Web Server hosting replica Base Station website
HTTP
Internet
User #3
User #n

Figure 2.4: Centralized WSN architecture.

between centralized and distributed architectures is found in table 2.1. Also a comparison is shown in figure 2.5 in terms of computation operations.

Scheme	**Base Node**	**Failure Risk**	**Cost**	**Energy efficient**	**Convergence**
Centralized	Needed	High	High	Low	Slow
Distributed	Unneeded	Low	Low	High	fast

Table 2.1: Centralized vs. distributed scheme comparison

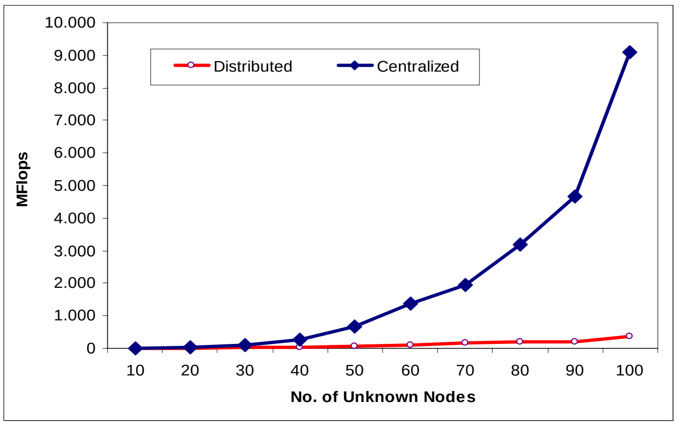


Figure 2.5: Comparison between centralized and distributed architectures with respect to the number of computational operations for an increasing of the unknown nodes [4].

As a distributed WSN allows the nodes to exchange data when required, it is common to use this architecture in collaborative or cooperative WSN. The main advantage of cooperative WSN is the cooperation between sensor nodes to carry a distributed localization algorithm. Section 2.2.1.2 gives an introduction to localization in WSN.

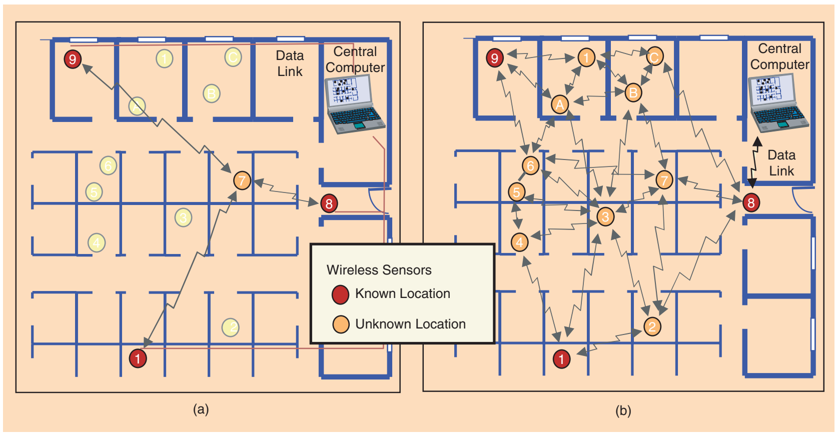


Figure 2.6: a) Non-cooperative localization: traditional trilateration is a special case in which measurements are made only between an unknown sensor node and three location aware nodes. b) Cooperative localization: the measurements made between any pair of sensors can be used to aid in the location estimate of the unknown sensor[5].

2.2 Introduction to Localization in WSN

Localization of a node in a network appear actually in different environments using wireless technologies such as WLAN (*Wireless Local Area Network*), WPAN (*Wireless Personal Area Network*), GSM (*Global System for Mobile*) and more recently WSN. The localization of a node can be done by means of the following choices:

- Cooperative localization allows the cooperation between unknown sensor nodes to obtain their positions. For that every unknown sensor must be within the coverage range of their neighbours. Cooperative localization techniques are useful in scenarios where GPS can not be used for several reasons. Some of these reasons can be for instance the high deployment cost or the low coverage from the reference nodes in severe environments such as urban areas or tunnels. The problem of cooperative localization generally is a 2-D (*two-dimensional*) problem as the shown in both figures 2.7 and 2.6. This problem consists to find the following $2n$ unknown-location node coordinates [5]:

$$\boldsymbol{\theta}_x = [x_1, \ldots, x_n], \quad \boldsymbol{\theta}_y = [y_1, \ldots, y_n] \tag{2.1}$$

given the known reference coordinates $[x_{n+1}, \ldots, x_{n+m}, y_{n+1}, \ldots, y_{n+m}]$, and pair-wise measurements $(X_{i,j})$ between the nodes i and j that can be any physical reading (such as ToA(*Time of Arrival*), AoA(*Angle Of Arrival*) or RSSI) which indicates a distance or a relative position.

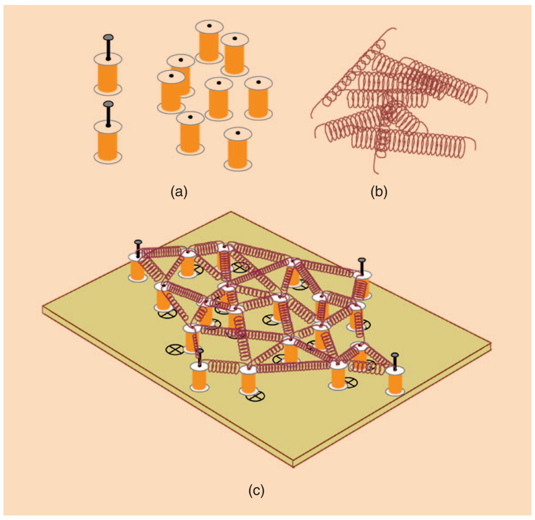


Figure 2.7: Cooperative localization is analogous to finding the location of (a) masses connected by a network of (b) springs. First, reference nodes are nailed on a board to their fixed coordinates. Springs with a length equal to the measured ranges can be compressed or stretched. The equilibrium point of all the masses (c) indicated by $\otimes$ represent the minimum-energy localization estimation [5].

- Non-cooperative whenever the unknown sensors only communicates directly to at least three reference nodes with known coordinates. For example, GPS use non-cooperative localization since a GPS receiver computes its coordinates with three satellites.

Localization can be performed by means of of the following requirements:

- Localization techniques
- Communication Technology
- Position Strategy

2.2.1 Localization Techniques

Many existing localization algorithms attempt to solve the problem of determining a node's location within a region. These algorithms are based on a localization technique. Different localization techniques differ in the way how this location is obtained. The choice of a localization technique depends on certain factors such as the network characteristics, device restrictions due to its hardware complexity, nature of the environment (indoor/outdoor), communication costs, error requirements and device mobility. These localization techniques are classified as:

- **range-free** schemes estimate the unknown node positions without direct distance information [24].
- **range-based** schemes estimate metrics in the received signal coming from the neighbours in order obtains the relative distances.

This section discusses both range-free and range-based solutions [25] [5] [2].

2.2.1.1 Range-free localization techniques

This technique assumes that an unknown node cannot obtain distances by direct measurements but it suppose that exists some physical parameter related somehow with a distance (this relationship may be obscure or unknown) such as the typical RSSI parameter. This kind of technique does not estimate directly the node position from the RSSI values but it uses a different approach. The idea is the following: each anchor node knows the RSSI (related to the distance) from the other anchor nodes and broadcasts them to the network. The set of RSSI values determine a set of concentric coverage circles centred at each anchor node with radius the distance corresponding to the set of RSSI values. The number of coverage circles is the same as the number of anchor nodes. Then the unknown node compares the RSSI from every anchor node with the known RSSI between all the other anchor nodes. This comparison results in a bounded region covered between those anchors with nearest RSSI from that anchor node which RSSI is compared. For every anchor a bounded region is obtained and the intersection of all these regions gives an area where the unknown node is located [2] [6]. It may be better to explain this with an example. Consider a network composed by three anchor nodes $AN_{i=1,2,3}$ and the unknown node called UN. It is assumed that the i_{th} anchor node broadcasts its coordinates (X_i, Y_i) periodically as well as the RSSI values of the received signals from other anchor nodes. Furthermore, for every anchor node AN_i there are three concentric coverage circles with radius the distance obtained with the RSSI from the other anchor nodes $AN_{j\, j \neq i}$. The notation RSSI(A, B) used following is the received power value by node B for a signal transmitted by node A. Therefore, lower the distance between nodes A and B higher the RSSI value as it is inversely proportional to distance. Each of the regions is determined as follows:

- First the UN compares its RSSI from the anchor AN_1 with the RSSI from the other anchor nodes of the network. If the hypothesis RSSI(AN1, AN2) > RSSI(AN1, UN) >> RSSI(AN1, AN3) is true then it implies that the unknown node is located in the red ring (a bounded region between the anchors AN_2 and AN_3).

- Following the UN compares its RSSI from the anchor AN_2 with the RSSI from the other anchor nodes of the network. If this comparison RSSI(AN2, AN1) > RSSI(AN2, UN) >> RSSI(AN2, AN3) is true it means that the UN lies in the green ring which is another bounded region between the anchors.
- The resolve the ambiguity of the two possible regions a third anchor node AN_3 is needed. Then the UN compares its RSSI from the anchor AN_3 with the RSSI from one of the other anchors. If RSSI(AN3, UN > RSSI(AN3, AN1) is true it implies that the unknown node is found in the lower intersection marked in brown.

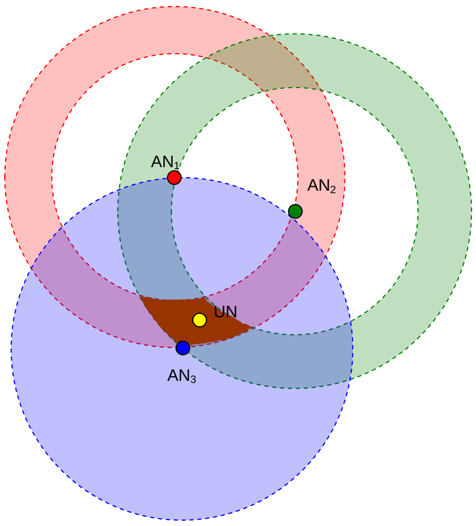


Figure 2.8: Range-free localization [2]

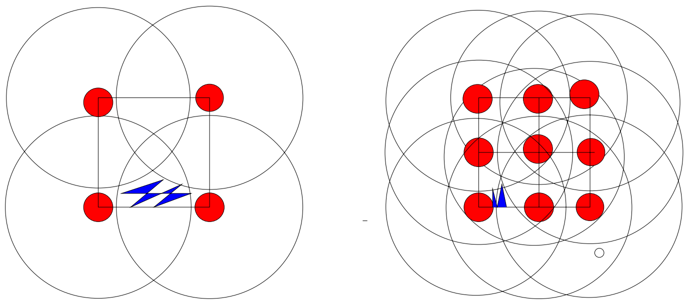


Figure 2.9: Granularity of localization regions vs. Range overlap. a)2x2 Grid of reference nodes. Fewer and Larger localization regions. b)3x3 Grid of reference nodes. More and smaller localization regions.[6]

Every anchor AN_i contributes to make the UN position bounds tighter, due to the intersection of the set of bounds for every anchor. On the other hand increasing the number of reference nodes, the range overlap of them also increase meaning more bounded regions. Therefore the granularity of the localization regions becomes smaller, and hence the accuracy of the localization estimate improves. This explanation can be seen in figure 2.9 [6]. The UN localizes itself to the connectivity region of this set of K reference points, which is defined by the *centroid* of these reference points [6].

$$(X_{est}, Y_{est}) = \left(\frac{X_{i1} + \ldots + X_{ik}}{k}, \frac{Y_{i1} + \ldots + Y_{ik}}{k} \right) \tag{2.2}$$

A metric used to know the accuracy of this estimate is the localization error LE(*Least Squares*) or RMSE(*Root Mean Square Error*):

$$LE = \sqrt{(X_{est} - X_a)^2 + (Y_{est} - Y_a)^2} \tag{2.3}$$

Therefore as more anchor nodes are within the range of the unknown sensor, greater accuracy in the positioning because more bounding regions will be obtained and hence the final intersecting area will be smaller. This approach is known as Centroid scheme [6].

Another range-free localization technique is the DV-Hop (*Distance-Vector-Hop*)[7][25]. The main idea of this technique differs in the way that the distances from the unknown nodes to the reference nodes is obtained. This technique uses a similar approach to the classical distance vector routing algorithms so that all nodes (both unknown and reference nodes) in the network find the minimum number of hops as well as the direction (related to the shortest path) to reach a reference node . In other words a routing metric (hop count) is used to measure the distance (in hops) between a source node and a destination node. Each hop in a path from source to destination is assigned a hop count value, which is typically 1. The number of hops is incremented by one if a certain node receives a packet with another destination. Every node in the network needs to know the 2-D coordinates (X_i, Y_i) as well as the minimum number of hops h_i to reach the anchor node i.
The first stage of the DV-Hop algorithm is to find the number of hops from every node to each anchor node (or reference node). Once an anchor node obtains the number of hops to other anchor nodes, it estimates an average size for one hop with the use of the following correction and broadcasts it to the nearby nodes:

$$HopSize_i = \frac{\sum \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\sum h_i}, i \neq j, \text{all reference nodes j}, \tag{2.4}$$

being (x_j, y_j) the 2-D coordinates of the anchor j, and h_i the distance in hops from anchor j to anchor i. When an arbitrary unknown node receives the correction from a nearby anchor then the unknown node computes the distance in meters to the every anchor node with the received

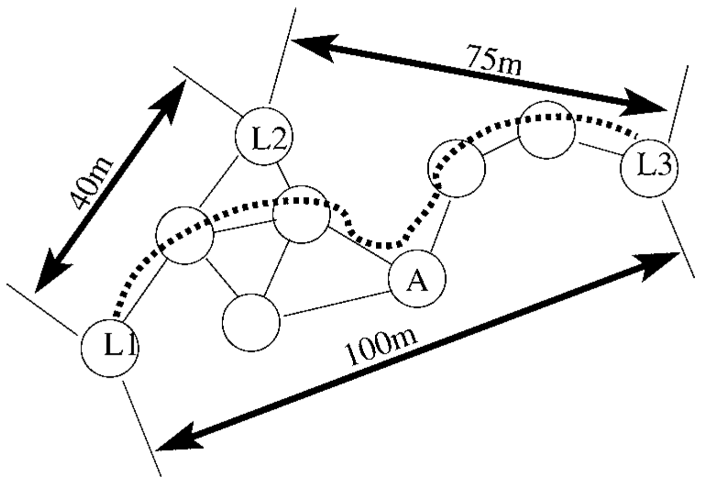


Figure 2.10: DV-Hop correction example [7].

HopSize correction and the known number of hops obtained at the first stage. At the last stage the unknown nodes perform trilateration to estimate its position. Following an example is used to explain the functionality of the DV-Hop algorithm. Consider the network graph shown in figure 2.10. L1, L2 and L3 represent the reference nodes whereas A is the unknown node that needs to find its coordinates. L1 has both the Euclidean distance and the number of hops to L2 and L3,which is 2 and 6 respectively. L1 computes the HopeSize correction with (2.4) which in fact is the average size of one hope, in meters: (100+40)/(6+2) = 17.5 m. The same does L2: (40+75) / (2+5) = 16.42 m and L3:(75+100)/(6+5) = 15.90 m. The unknown node gets the average size per hop from one of closest anchor nodes. With this approach, the node A chooses the correction computed by L2. Once it has the average size of one hop it computes the distance to L1, L2 and L3 by using the minimum number of hops stored in its database. Finally the unknown node starts a localization procedure to obtain its own coordinates with the obtained distances to the anchor nodes. This localization procedure is called trilateration and it is explained in section 2.2.1.2.

One of the drawbacks of DV-hop is that the network graph must not change,it must be static in order to compute the number of hops as well as the euclidean distances between anchor nodes. The advantages are its simplicity and the fact that it does not depend on range measurement error due to the RSSI variation. There are other algorithms that follow the same approach but instead of using the number of hops as a metric they uses other metrics such as the RSSI-based distance or the true Euclidean distance. Some of these algorithms found in [7] are known as DV-distance or DV-Euclidean.

The accuracy of the DV-Hop with trilateration is dependent on the number of anchor nodes used to compute the average distance of a single hop. Simulations shown in [26] demonstrate that the best accuracy is around 50 m when the anchor ratio is 5%. This accuracy is improved when the number of anchor nodes increases.

There exist other methods based on the same distance vector approach focused to measure ranges from an unknown node to neighbours that previously have successfully obtained its position. Once an unknown node computes its position it broadcasts its computed coordinates to its neighbours. These distributed algorithms known as iterative or successive refinement are used to improve the accuracy. The accuracy of refinement algorithms is dependent on:

- the accuracy of the initial position estimates
- the magnitude of errors in the range estimates
- the average number of neighbours
- the fraction of anchor nodes

According to [27] the position error is less than 33% in a scenario with 5% range measurement error, 5% anchor population and an average connectivity of 7 nodes.

Another range-free technique is the Amorphous Positioning based on DV-Hop [25]. It has a similar approach than the DV-hop algorithm but only differs in the way how the HopSize metric is computed at the second stage. The first stage of this algorithm is the same as DV-Hop consisting to find the minimum number of hops to reach the anchors in the received beacons. In the second stage each node computes locally an estimation of the single hop distance whereas in the case of DV-Hop the correction is computed only by the anchor nodes. At the third stage, once the estimated hop distance is obtained the nodes computes the distances to a minimum of three anchors to obtain its estimated coordinates. The localization procedure at this stage is trilateration. A more detailed explanation of the algorithm can be found in [28].

This section has covered some of range-free algorithms found in the literature. All of the range-free algorithms are called as coarse grained localization algorithms because the accuracy is not good in comparison with the range-based localization methods discussed next. The best accuracy with range-free algorithms is in the order of meters whereas the best accuracy obtained using range-based methods is in the order of few centimetres.

2.2.1.2 Range-based localization Schemes

As the name mentions these schemes estimates the distances between every pair of connected nodes (sensors-to-sensors and sensors-to-anchors) by analysing a metric or any property of the received signal that depends on the relative positions of the nodes. These ranging metrics are described following:

- RSSI: It is defined as the square of the voltage obtained by a receiver's circuit that measures the received signal strength. Also the RSSI is equivalent to the squared magnitude of the signal amplitude which is the received power. The received power is a function of the distance between the transmitter and the receiver that is proportional to $d^{-\gamma}$, where γ is the known as path-loss exponent. The minimum value of γ is 2 for LOS (*Line-Of-Sight*) environments. The path-loss model at a distance d_{ij} from the transmitter j is shown in (2.5) [5]:

$$\begin{gathered} P_L(d_{ij}) = P_0 - 10\gamma log_{10} d_{ij} - v_{ij} \\ d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i + y_j)^2}), \end{gathered} \tag{2.5}$$

  where P_0 is the received power (dBm) at 1 m distance and v_{ij} represents log-normal shadow fading due to the multipath effect in wireless channels. The multipath effect appears when several signals travel from the transmitter to the receiver through different spatial paths. These signals arrive at the receiver with a certain time delay $\nabla\tau$ and a certain phase delay $\nabla\phi$. At the receiver the incoming signals replica are added thus causing either a constructive or a destructive interference. Despite the simplicity of this ranging metric it has the main drawback that the RSSI is random due to multipath. The effect of multipath can be diminished by using a spread-spectrum method (e.g,direct-sequence or frequency hopping) which averages the received power over a wide bandwidth reducing then the interference in the unlicensed bands [5].

- ToA: Also named Time of Flight is another kind of ranging metric that obtains a distance estimate by means of the direct signal propagation delay from the transmitter to the receiver. This metric provides more accuracy than RSSI (in the order of few cm) because it does not suffers variations due to multipath. However the provided high accuracy implies a complex hardware and synchronized devices with a very good clock precision. Inaccuracies in the clocks synchronizations translates directly to an imprecise location. When synchronization is not available or can not be used because of the device hardware constraints other alternatives can be used. An alternative is the combination of two signals with different frequencies and hence different propagation speed as shown in figure 2.11 (usually both a RF signal and an ultrasound signal are used). The receiver obtains the distance by multiplying the difference between the propagation time delay of each incoming signal with the light speed.

  Another alternative is the compensation of clock phase differences, a common practice known as two-way ranging or round-trip ToA discussed below. Despite the high accuracy that ToA provides it requires to measure the propagation time delay of the direct signal, that is, it requires LOS conditions. If the wireless channel is NLOS(*Non Line Of sight*) its performance is severely decreased. On the other hand ToA has the inconvenient of

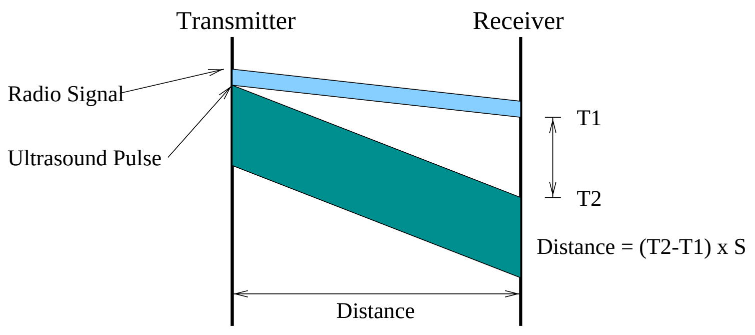


Figure 2.11: ToA measurement using ultrasound and radio signals [8].

multipath in the sense that many received multipath signals can mask the peak of the LOS signal. This drawback can be combat by using wider band signals. UWB(*Ultra WideBand*) is high-bandwidth technology used common in ToA measurements.

- TDoA (*Time Difference Of Arrival*): this ranging metric obtains the unknown coordinates (x, y) by resolving a $2\times$ 2 equations system that takes into account the separation distance between every pair of nodes with known coordinates. The basic idea is the following: the unknown device sends a signal at the time instant $t_0 = t_0' + \varepsilon$ being ε the synchronization time error. Then the signal is received to at least three reference nodes at the time instants $T_A = t_0 + \tau_A$, $T_B = t_0 + \tau_B$ and $T_C = t_0 + \tau_C$. Finally the time difference of arrival is computed to obtain the unknown coordinates. Mathematically it is formulated in (2.6).

$$\begin{gathered} T_i - t_0 = \tfrac{1}{c}\sqrt{(x-x_i)^2 + (y-y_i)^2} \\ T_i - T_j = \tfrac{1}{c}\left(\sqrt{(x-x_i)^2 + (y-y_i)^2} - \sqrt{x^2+y^2}\right) \quad i = B, C \,\text{and}\, j = A, \end{gathered} \tag{2.6}$$

where (x_i, y_i) is the location of the receiver i, c is the light speed, A, B, C are the reference nodes and (x, y) are the coordinates of the unknown node.
This ranging metric is widely used in GPS or in cellular networks in downlink as shown in the figure 2.12. For this case the user's terminal (the unknown node) computes the time difference between the received signals from each synchronized base station. In other cases it can be that unknown mobile itself sends the synchronized signals to each of the reference nodes (uplink). As synchronization is needed TDoA presents the same disadvantage than the ToA technique.

- RToA(Round-trip Time Of Arrival): This metric is also known as two-way ranging metric because it measures the round trip time between the transmitter and the receiver. Thus RToA avoids the need to have synchronized clocks which is required by ToA and TDoA ranging metrics. With RToA, the original sensor transmits a signal at a time t_0 to a

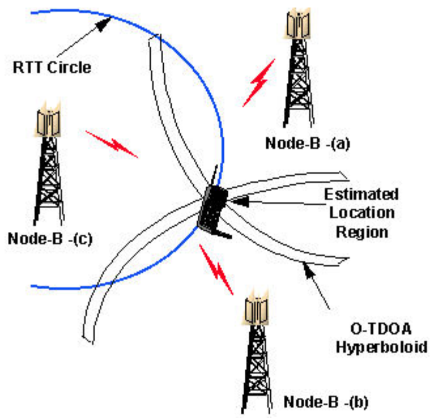


Figure 2.12: TDoA measurement in celular networks [9].

second receiver sensor, which immediately replies with its own signal. The reply reaches the original sensor at time t_r which is the initial time t_0 plus twice the propagation delay and an internal processing time of the receiver sensor. The receiver parameter usually is provided by the manufacturer:

$$t_r = t_0 + 2\frac{d}{c} + \Delta \tag{2.7}$$

A graphical example of this metric is shown in figure 2.13. The time measurement starts at node A by sending a package to node B at time t_0. When node B receives the package it starts to processing which takes a time Δ. After processing a reply is sent to node A which receives the package at time t_r. Then the distance from node A to node B is obtained from the difference between t_r and t_0. Double-sided two-way is used to avoid the inconvenient of the clock drift when the processing time is much higher than the propagation time.

- AoA: this metric analyse the direction of the arrival signal, rather than the distance to neighbouring sensors as done in the previous discussed metrics. There are two ways to measure the angle of arrival as shown in figure 2.14. In any way AoA requires an array of antennas embedded in the sensor. The first way is based on measuring the phase delay $\phi = 2\pi f_c \tau$ of every received signal, being f_c the narrowband signal frequency and τ the difference in the arrival times for a received signal at each of the sensor antennas. The other way uses the RSS ratio between two or more directional antennas to find the direction of arrival that is to find the antenna where the signal is coming from.

  This ranging metric requires several antennas embedded at the sensor implying an increase of both the device dimension and therefore the cost. Hence AoA is not appropriated in a WSN.

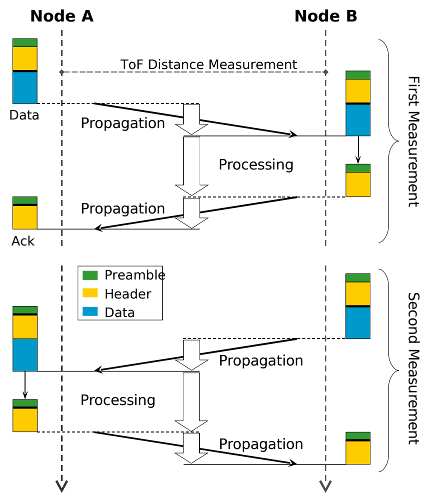


Figure 2.13: Double-sided two-way ranging technique [10].

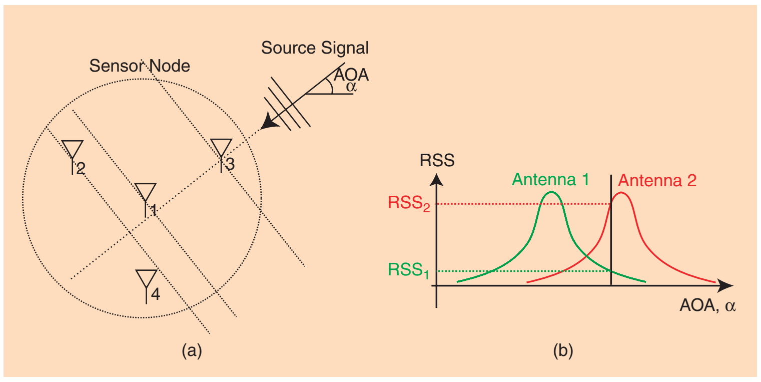


Figure 2.14: AoA estimation methods.(a) AoA is estimated from the ToA differences among antennas elements embedded in the sensor node. (b) AoA can also be estimated from the RSS ratio RSS_1 / RSS_2 between directional antennas[5].

All of the previous discussed techniques tries to estimate the distances to the reference nodes (except AoA) at the first stage. At the second stage, the unknown node estimates its relative position by means of the following procedures known as trilateration or multilateration:

- Trilateration/multilateration: this technique shown in figure 2.15 is valid only in direct communications between unknown sensors to anchors. The main idea of this technique is to find those coordinates (x, y) where a set of n corresponding circles to n anchors intersect. If $n = 3$ the procedure is known as trilateration whereas if $n = 3$ the procedure is known as multilateration. The unknown node obtains the distances d_1 and d_2 from the anchors AN_1 and AN_2 with the RSSI. A third AN_3 anchor node is needed to solve the ambiguity of the two possible locations determined by the two circle's intersections.

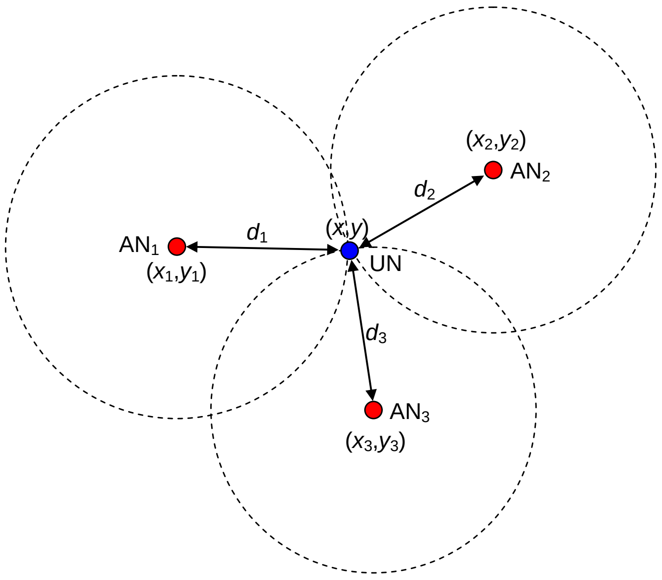


Figure 2.15: Trilateration using distance measurements. The circles represent the coverage area of the sensor nodes [2].

According to 2.15 the unknown coordinates (x, y) can be find by solving a set of linear equations that represent the circle equations shown in (2.8).

$$\sqrt{(x - x_i)^2 + (y - y_i)^2} = d_i \quad \text{for} \quad i = 1, 2, 3, \tag{2.8}$$

where d_i is the measurement distance between the unknown sensor and the anchor i with coordinates (x_i, y_i). Notice that the only parameter in (2.8) that can affect the accuracy of the location error is the measured distance. It is known that the accuracy of the estimated distance is dependent on the used ranging metric. The accuracy of the measured distance by the RSSI being the worst ranging metric can be increased if the same approach in (2.8) is realized with more than three anchor nodes. It known as multilateration and the problem which is generalized to N anchor nodes focuses to solve the following [2]:

$$C(x, y) = \sum_{i=1}^{N} w_i \left(\sqrt{(x - x_i)^2 + (y - y_i)^2} - d_i \right)^2 \quad \text{for} \quad i = 1, \ldots, N, \tag{2.9}$$

where $w_1, w_2, \ldots, N$ are positive weights that emphasize the most reliable measurements (e.g, they can be 1/0 or any other quantity directly related to the quality of the radio link). The minimization of (2.9) is an unconstrained optimization problem in the unknowns (x, y).

$$(\hat{x}, \hat{y}) = \underset{x,y}{\operatorname{argmin}}\, C(x, y) \tag{2.10}$$

Finding the solution of (2.10) is a difficult nonlinear optimization problem. Therefore some suboptimal approximations can be used to transform the nonlinear problem in a linear problem [2]. Thus, squaring (2.8) and defining the following variables:

$$R = x^2 + y^2$$

$$R_i = x_i^2 + y_i^2 \quad ,1 \leqslant i \leqslant N \tag{2.11}$$

results in the following system of equations:

$$-2x_i x - 2y_i y + R = d_i^2 - R_i \quad ,1 \leqslant i \leqslant N \tag{2.12}$$

which are linear in (x, y, R) and thus can be written in a matrix form:

$$\mathbf{Az} = \mathbf{b}, \tag{2.13}$$

where:

$$\mathbf{z} = [x, y, R]^T$$

$$\mathbf{A} = \begin{bmatrix} -2x_1 & -2y_1 & 1 \\ -2x_2 & -2y_2 & 1 \\ \vdots & \vdots & \vdots \\ -2x_N & -2y_N & 1 \end{bmatrix} \tag{2.14}$$

$$\mathbf{b} = \left[d_1^2 - R_1, d_2^2 - R_2, \ldots, d_N^2 - R_N\right]^T$$

Now the solution of (2.13)is computed below by WLS(*Weighted Least Squares*):

$$\hat{\mathbf{z}} = \underset{z}{\operatorname{argmin}} (\mathbf{Az} - \mathbf{b})^T \mathbf{W} (\mathbf{Az} - \mathbf{b})$$

$$\mathbf{W} = \begin{bmatrix} w_1 & 0 & \ldots & 0 \\ 0 & w_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & w_N \end{bmatrix} \tag{2.15}$$

The solution of (2.15) can be seen in (2.16) which gives the estimation $(\hat{x}, \hat{y})$ as the first and second components of the vector $\hat{\mathbf{z}}$:

$$\hat{\mathbf{z}} = (\mathbf{A^T W A})^{-1} \mathbf{A^T W b} \tag{2.16}$$

One the one hand notice that this method is fully centralized because all the variables (distance measurements) are combined in a central processing unit to perform localization. On the other hand multilateration can provide better accuracy than other range-based techniques since more anchors nodes with known positions and optimal weights can be used together to find the unknown coordinates. However this technique is valid when the sensor nodes are in the coverage area of the anchor nodes.

2.2.2 Communication Technology

The accuracy in the location error is dependent on the other factors in addition to the used localization localization techniques. The communication technology is also a very important requirement for localization in WSN. This section discusses some of the current wireless communication technologies and their impact in WSN: WLAN, Bluetooth, UWB,and ZigBee [11]. In principle they can be applied in WSN because their range is short.

- WLAN: this wireless technology was developed to provide wireless communication in the unlicensed ISM(*Industrial, Scientific and Medical*) band of both 2.4 GHz and 5 GHz to the original LAN(*Local Area Networks*) networks which computers communicated by means of wires. The WLAN specifications are found in its standard IEEE 802.11. This standard is kept by the WiFi Alliance organization providing interoperability between devices from different manufacturers. These specifications adapts the physical and MAC(*Medium Access Control*) layer of the original Ethernet protocol to provide communication through air. The following family of WLAN standards are extended in a lot of places to provide wireless communication to hundreds of users:
  - IEEE 802.11g: this standard is an amendment of its predecessor, IEEE 802.11b which was based on either on DSSS(*Direct Sequence Spread Spectrum*) with a gross throughput upto 11 *Mbps*. The allowed bandwidth in both 800.11g and 802.11b is $20MHz$ and the non-overlapping channels in the 2.4 GHz band are 1, 6 and 11. The physical layer of 802.11g is based on multiple carrier or OFDM(*Orthogonal Frequency Division Multiplexing*) with digital constellations BPSK(*Binary Phase Shift Keying*), QPSK(*Quadrature Phase Shift Keying*), 16QAM(*Quadrature Amplitude Modulation*) and 64QAM. This improvement in the IEEE 802.11g physical layer leads to a gross throughput upto 54 *Mbps* when using the 64QAM constellation. The coverage is up to 300 m which can be improved by means of directive antennas and powerful transmitters.
  - IEEE 802.11n: it is a recent amendment which improves both the throughput and the coverage of the previous 802.11 standards by adding more than one antenna at the transmitter and at the receiver. The last is called multiple-input multiple-output

(MIMO) with notation $M \times N$ which means that the input to the wireless channel is through M antennas whereas the output is from N antennas. Several antenna configurations cane be considered: 2x2,2x3,3x3. Another improvement with respect to the previous standards is its ability to operate under the both both bandwidths 20 MHz and 40 MHz. This standards specifies a maximum gross throughput of 600 *Mbps* under the assumption of having 4 spatial streams, 64QAM constellation, 40 MHz channel bandwidth and GI(*Guard Interval*) of 800 ns.

Several positioning/tracking experiments with WLAN technology have been carried on, although most of them in indoor scenarios [29][30][31][32]. The set of ToF and RSSI ranging metrics have been proved for localization in WLAN [33][34][35]. Taking the advantage that ToF performs better than RSSI in terms of a reduced location error (in order of cm), its inherent hardware complexity can be exploited in WLAN networks as they are not energy-constrained as happens in WSN.

- Bluetooth known with its standard IEEE 802.15.1 is the set of the physical and MAC layer specifications for a WPAN. A comparison with respect to WiFI is the following:
  - Less coverage: this parameter depends on the bluetooth chipset class. Class-1 with a range of 100 m and a maximum allowed power of 20 dBm, Class-2 with a range of 10 m and a maximum allowed power of 4 dBm and Class-3 with 1 m of coverage and 0 dBm of transmit power.
  - Less throughput: the nominal transmission speed can reach upto 1 *Mbps* (in practice around 720 *Kbps*) with the Bluetooth Version 1.2 and upto 3 *Mbps* (in practice around 2.1 *Mbps*) with the enhancement Bluetooth 2.0+EDR (Enhancement Data Rate).
  - Pysical layer was based on FHSSS(*Fast-frequency Hopping Spread Spectrum*) and the modulation GFSK(*Gaussian Frequency Shift Keying*) in older versions. Newer versions use a combination between the modulation GFSK and either of the modulations $\pi\backslash 4$-DQPSK (at its lower rate of 2 *Mbps*) and 8DPSK (at its higher rate of 3 *Mbps*). The same ISM frequency band is used: 2.4 GHz.
  - Lower energy consumption (in the order of mA even in stand-by mode), cheaper and smaller devices.

Bluetooth has the main advantage that can be used to transfer data between small devices such as mobile phones, PDAs, game consoles in areas without WiFI coverage. Currently most of the personal devices carry bluetooth chipsets. Bluetooth technology is a better choice than WiFI for WSN since it provides less energy consumption and less throughput (a high amount of resources is not needed in WSN). Several applications using bluetooth

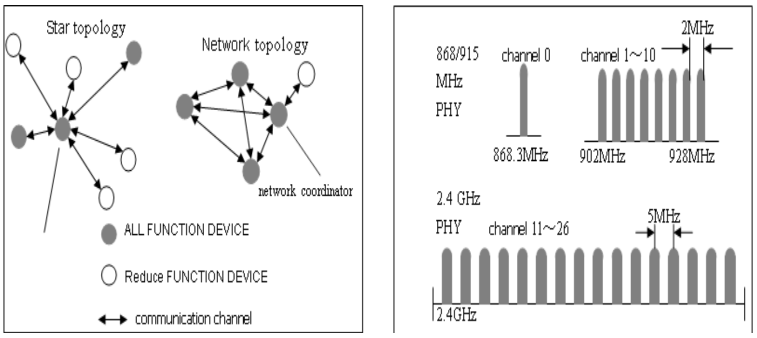


Figure 2.16: a)Zigbee networking topology, b)Zigbee operating frequency bands [11].

with RSSI in WSN have been carried on to perform localization as it does not require extra hardware. [36][37][38].

- UWB [5]: this wireless technology employs radio signals composed by narrow pulses of very short duration (the order of nanoseconds). In the frequency domain it corresponds to a bandwidth greater than 500 MHz. Some of the standards that cover UWB are IEEE 802.15.3a and IEEE 802.15.4a. Furthermore, UWB provides high data rates (at least 480), low power consumption and low cost but at the expenses of very low range. Digital home applications as well as other applications requiring short range can take advantage of this broadband wireless technology. The very high bandwidth of UWB in comparison to the other wireless technologies leads to a very high temporal resolution making it ideal for high-precision radiolocation applications. Therefore the use of UWB with the ranging metric ToA leads to a very high accuracy (the order of few cm). Several localization applications with UWB in WSN are reported in [39][40][41].

- Zigbee technology provides low data rate, low power consumption, low transmit rate, low cost and mainly it is addressed to remote control applications. The standardization of Zigbee is IEEE 802.15.4 specifying several network topologies such as mesh, star and P2P(*Point to Point*). The physical layer is based on DSSS and the range per node can be upto 500 m in LOS conditions [42] depending on the transmit power and the used antenna. Another interesting feature is the use of IPv6 (*IP version 6*, that is a 128-bit IP address with the least significant 64-bit for host addressing) allowing to form a network of thousands or even million of devices. The adopted standard defines different frequencies supported by Zigbee each of them corresponding to a certain number of channels and a throughput: 16 channels at 2.4 GHz offering 250 $Kbps$, 10 channels at 915 MHz offering 40 $Kbps$ and 1 channel at 868 MHz offering 20 $Kbps$. Figure 2.16 shows two kinds of network topologies as well as the supported operating frequency bands.

  Since May 2003 Zigbee is under the tutelage of Zigbee Alliance upon which Zigbee devices and applications are constructed. Furthermore Zigbee is more attractive than Bluetooth for WSN because the energy consumption is much lower (the order of μA in standby mode). Following it shows the general Zigbee/804.15.4 features:

– Dual PHY (2.4 GHz and 868/915 MHz)

– Data rates of 250 $Kbps$ (2.4 GHz), 40 $Kbps$ (915 MHz), and 20 $Kbps$ (2.4 GHz)

– Optimized for low energy consumption ($< mA$)

Zigbee is widely used in several WSN applications. In the context of positioning zigbee has the drawback of a bad clock resolution being inappropriate the use of ToA ranging techniques. Taking the advantage of the RSSI circuitry simplicity, it can be used to perform localization. Some practical experiments demonstrate location errors in the order of a few centimetres or a few meters [43][10][44].

2.2.3 Positioning strategy

This section is addressed to give an explanation of the different positioning strategies that can be used in WSN. As explained in section 2.1 there are two positioning strategies to be considered: non-cooperative and cooperative. In the non-cooperative the unknwon nodes have to communicate with at least three reference nodes to obtain the 2-D coordinates or 4 anchor nodes to obtain 3-D coordinates. It happens in GPS. With this strategy the problem of obtaining the coordinates can be solved with the known trilateration/multilateration. However this positioning procedure can be carried with the requirement that the unknown nodes must receive signals from the anchor nodes of course with a power greater than its sensibility. This requirement makes difficult the use of this non-cooperative strategy in those scenarios where only few anchor nodes can be positioned in a large area of sensor networks. In order to combat this issue a cooperative strategy can be taken into account. In cooperative scenarios every node in the network cooperates with all of its neighbours to carry on the positioning, for example measuring the pairwise distances. This strategy is preferable in most of the WSN scenarios since more unknown nodes than anchor nodes are distributed in big geographical areas like a forest and also there can be nodes without direct communication to the anchor nodes. On the other hand the cooperative strategy allows the use of two anchor nodes instead of three. By allowing the cooperation between nodes, the ambiguity that appears in non-cooperative with two anchor nodes is solved by the pairwise distances measurements. It is shown in figure 2.17.

When considering cooperation, the adopted algorithm can be classified in both centralized and distributed:

1. Centralized Algorithms: these algorithms require that all the unknown nodes send their distance measurements to a central processing unit which runs a centralized algorithm. The centralized algorithm computes the relative positions of all the unknown nodes. Finally

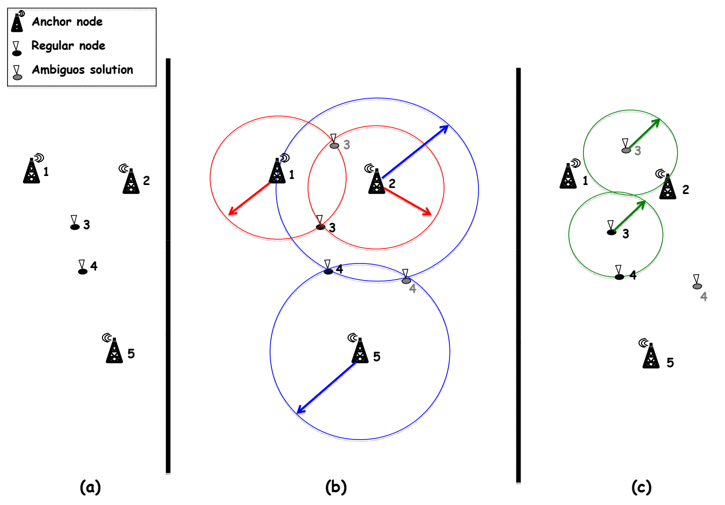


Figure 2.17: a)Network with three anchor nodes and two unknown nodes, b)Ambiguity existing in non-cooperative strategy when using only two anchor nodes, c) The ambiguity disappear when cooperation between nodes 3 and 4 is allowed. As observed only one of the green circumferences intersects with one of the blue circumferences in b).

the algorithm applies a transformation operation given the coordinates of the anchor nodes to find the absolute positions. Once the algorithm computes to absolute positions, these results are sent back to the queried nodes.

The centralized approach has the advantage that can perform complex calculations or even iterative complex calculations because the fusion center does not have energy/hardware constraints as it happens in the sensor nodes. However a disadvantage of this strategy is that all nodes must reach the central processor unit with the consequence of having an increase of the network traffic and a possible bottleneck near or at the fusion center. On the other hand, the performance of the whole network depends on the robustness of the central unit. Currently there are two suitable centralized algorithms for large-scale networks with low density deployment of reference nodes: MLE(*Maximum Likelihood Estimator*) and MDS(*Multilateration Scaling*) [45]:

- MLE assumes that the statistical of the data is known and can be described with a statistical model (e.g.,Gaussian or log-normal). MLE is theoretically optimal in the sense that it is asymptotically efficient when the SNR(*Signal To Noise Ratio*) ratio increases. When the network is large, the number of unknown parameters (the 2-D unknown coordinates) increase in a factor of $2N$ being N the number of unknown nodes. Thus it may not be computationally feasible to determine the solution of MLE by using brute-force grid-search method [45]. In these cases the MLE must be solved using iterative nonlinear algorithms such as the NLS(*Non linear Lest Squares*).

Iterative algorithms have the drawback that they depend on the initial value and in general they need a well-conditioned matrix to assure a convergence. Unless the MLE is initialized to a value closed to the optimum it is possible that the algorithm does not find the global optimum [5]. Hence convex constraints are presented to force the unknown sensor's location within a radius r and/or angle range from another sensor. Thus the MLE can be treated as a convex optimization problem. Another additional problem related to MLE is the statistical model dependency with the measurements. MDS combats these problems.

- MDS algorithms are widely used in many areas, including statistics, psychology, sociology, political science and can be used as well to formulate sensor localization from range measurements as an LS problem [45]. These algorithms consists of two stages: the first stages the MDS finds those locations that minimizes the mean square error with the given pairwise distances of all nodes in the network. The coordinates or relative positions found by MDS in the first stage are in a different coordinate system than the true given by the anchors positions. Thus a second stage is required which applies a transformation operation to the relative positions to obtain the absolute positions in the true coordinate system. Other MDS-based techniques use a weighting scheme for the measurements according their accuracy, and thus improving the localization error [5].

2. Distributed Algorithms: in contrast to the centralized algorithms these algorithms do not consider the use of a central unit to handle the calculations. Each node must compute its own position with the support of its neighbour nodes. It provides an advantage to the centralized algorithms since each node obtains faster its position as it does not need to reach and wait for a result from the fusion center that could be far away. Other advantage is that the positioning can be performed even if the size of the network increases. However these algorithms need to run several iterations to find an accurate estimate. Distributed algorithms for cooperative localization generally fall into one of following categories [5]:

   - *Network multilateration*: each sensor estimates the shortest path distance through multi-hop algorithms to at least three visible reference nodes. Once the distances are obtained multilateration technique is carried on. Examples of these algorithms are the well known range-free algorithms DV-hop discussed in section 2.2.1.
   - *Successive refinement*: these algorithms known as non-Bayesian algorithms try to find the coordinates of all the unknown nodes in the network with an iterative procedure. Initially there are two subsets of nodes: one subset A with known nodes locations and another subset B with unknown nodes locations. At each step those nodes in the subset B which can measure the RSSI from a minimum of three other nodes in the subset A compute multilateration to obtain their positions. Once they found their

positions they become new members in the subset A. This procedure which is known as IMM(*Iterative Multilateration*)[45] is repeated for all the unknown nodes until the subset B becomes empty.

The convergence of these algorithms is always a drawback and on the other hand the global accuracy depends on the local results computed by every node.

Most of the centralized and distributed algorithms must face the high relative costs of communication. Centralized algorithms in large networks require each sensor's measurements to be sent over many hops to a central processor. In contrast with the distributed algorithms the sensors send messages only one hope (to its direct neighbours). However these algorithms require a certain number of iterations to converge until all nodes obtain their accurate positions. Distributed algorithms performs better than centralized when the number of iterations is less than the average number of hops [5]. There exist hybrid algorithms that combine both centralized and distributed to reduce the energy consumption per node. These hybrid algorithms are applied when the network is fractioned to a group of clusters. Recall that a cluster in a distributed WSN is a set of nodes with one selected candidate called manager node or cluster head to be the local processor unit. As an example a Bluetooth piconet can be seen as a cluster with one master and several slaves. Each manager node carries out centralized algorithms by using the gathered data from the nodes in the cluster where it belongs and thus estimating a map of the cluster. Then a distributed algorithm is executed taking the computed results from all the manager nodes to merge and optimize the local estimates such as described in [46]. Such hybrid algorithms are a promising topic for future research. There are applications that use the concept of clusters for target tracking issues[22] [47].

2.3 Introduction to tracking techniques for mobile nodes

All the localization techniques discussed in previous sections are focused only in scenarios without mobility. For example, multilateration, MDS and MLE can not be applied to locate a set of one or more mobile unknown nodes. The term positioning usually is referred for fixed points. There exist a lot of algorithms for tracking but the most common and used in practice are two: KF(*Kalman filter*) and EKF(*Extended Kalman Filter*) [10][48][49]. The main differences between them is how they make the assumption of the signal model: linear in Kalman filter and nonlinear in Extended Kalman filter. This section gives a brief introduction to these tracking algorithms [50].

2.3.1 Kalman Filter

The system dynamic model used by Kalman filter is the *first order Gauss-Markov* model which has the form shown in (2.17):

$$x(k+1) = f(x(k)) + u(k) \quad k+1 \geq 0, \tag{2.17}$$

where $u[n]$, called the *process noise*, *excitation noise* or *model noise* is WGN(*White Gaussian Noice*) with variance $\sigma_u^2 \sim \mathcal{N}(\mu_u, \sigma_u^2)$. On the other hand $x[-1]$ corresponds to the initial state of Kalman and it is independent of $u[n]$. In the context of Kalman filter the equation in (2.17) is called as *state space model* which estimates the state at the next time sample. It is clear that with only one sample it cannot obtain an estimate of the temporal tendency of the signal. It needs all the stored samples from $n = 0$ up to $n = n - 1$. In other words, $\hat{x}[n]$ is an estimation based on the previous observations or measurements consisting of a scalar sequence $\{z[0], z[1], \ldots, z[n]\}$ with n increasing. Such operation is referred as *filtering* because the previous data is filtered to obtain the actual state. Therefore, $\hat{x}[n]$ is a recursive estimator which is based on the sequential Bayesian MMSE(*Minimum Mean Square Error*) estimator. Under the *Gaussian assumption* for the initial state and all the noises entering into the system, the Kalman filter is the **optimal MMSE state estimator**. If the initial state is *not Gaussian* then the Kalman filter algorithm is the **best LMMSE**(*Linear Minimum Mean Square Error*) [12]. Furthermore the scalar observation can be extended to vector observations $(\mathbf{z}[0], \mathbf{z}[1], \ldots, \mathbf{z}[n])$ and also the scalar state can be extended to vector state $\mathbf{x}[n] = \mathbf{F}\mathbf{x}[n-1] + \mathbf{u}[n]$. Summarizing to a more (not the most) general case when both the state and observation are vectors, which is the most common in practice:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{F}\mathbf{x}(k) + \mathbf{G}(k)\mathbf{u}(k) \\ & \qquad\qquad\qquad\qquad k+1 \geq 0 \\ \mathbf{z}(k+1) &= \mathbf{H}(k+1)\mathbf{x}(k+1) + \mathbf{w}(k+1), \end{aligned} \tag{2.18}$$

where $\mathbf{x}(k+1)$ is the $p \times 1$ signal vector, $\mathbf{z}(k+1)$ is the $M \times 1$ measurement vector and $\mathbf{F}, \mathbf{G}, \mathbf{H}$ are known matrices of dimensions $p \times p$, $p \times r$ and $M \times p$, respectively. The matrix $\mathbf{F}$ is called the transition matrix, as it does the transition operation from the previous state to the current state and the matrix $\mathbf{H}$ is known as observation matrix. It can be seen in (2.18) that the process noise vector $\mathbf{G}(k)\mathbf{u}(k) \sim \mathcal{N}(0, \mathbf{Q})$ has dimensions $p \times 1$. Moreover this process noise is independent from sample to sample because it is WGN but is dependent of the noise vector $\mathbf{v}$. On the other hand the observation noise vector $\mathbf{w}(k+1) \sim \mathcal{N}(0, \mathbf{R})$ as well is a WGN with dimensions $M \times 1$. The matrices $\mathbf{Q}$ and $\mathbf{R}$ are called the *process noise covariance matrix* of the process noise $\mathbf{G}(k)\mathbf{u}(k)$ and *measurement noise covariance matrix* of the measurement noise $\mathbf{w}$.

The MMSE estimator of $\mathbf{x}(k+1)$ based on $(\mathbf{x}(k), \mathbf{x}(k-1), \ldots, \mathbf{x}(-1))$ or also defined as:

$$\hat{\mathbf{x}}(k+1 \mid k, k-1, k-2, \ldots, -1) = E(\mathbf{x}(k+1) \mid \mathbf{x}(k), \mathbf{x}(k-1), \ldots, \mathbf{x}(-1)) \tag{2.19}$$

can be solved through the following two Kalman stages that minimize the mean square error.

1. Prediction stage: this stage predicts the current data based on the previous $\mathbf{x}[n-1]$, $n \geq 0$ gathered data. This prediction is applied to the state vector of dimensions $p \times 1$ and also to the MMSE matrix of dimensions $p \times p$. The set of involved equations are (2.20) and (2.21).

   <u>State Prediction</u>:

   $$\hat{\mathbf{x}}(k+1 \mid k) = \mathbf{F}\hat{\mathbf{x}}(k \mid k) + \mathbf{G}(k)\mathbf{u}(k) \tag{2.20}$$

   <u>State Prediction Covariance</u> $(p \times p)$:

   $$\mathbf{P}(k+1 \mid k) = \mathbf{F}\mathbf{P}(k \mid k)\mathbf{F}' + \mathbf{Q}(k) \tag{2.21}$$

2. Correction stage: this stage minimizes the measurement residual between the measurement $\mathbf{z}(k+1)$ and the measurement prediction $\hat{\mathbf{z}}(k+1 \mid k)$ with the Kalman gains. The set of equations used in this stage are (2.23),(2.24) and (2.25).

   <u>Innovation Covariance</u> $(p \times p)$:

   $$\mathbf{S}(k+1) = \mathbf{R}(k+1) + \mathbf{H}(k+1)\mathbf{P}(k+1 \mid k)\mathbf{H}(k+1)' \tag{2.22}$$

   <u>Filter Gain</u> $(p \times M)$:

   $$\mathbf{W}(k+1) = \mathbf{P}(k+1 \mid k)\mathbf{H}'(k+1)\mathbf{S}(k+1)^{-1} \tag{2.23}$$

   <u>Updated state estimate</u>:

   $$\hat{\mathbf{x}}(k+1 \mid k+1) = \hat{\mathbf{x}}(k+1 \mid k) + \mathbf{W}(k+1)(\mathbf{z}(k+1) - \mathbf{H}(k+1)\hat{\mathbf{x}}(k+1 \mid k)) \tag{2.24}$$

   <u>Updated state covariance</u> $(p \times p)$:

   $$\mathbf{P}(k+1 \mid k+1) = \mathbf{P}(k+1 \mid k) - \mathbf{W}(k+1)\mathbf{S}(k+1)\mathbf{W}(k+1)' \tag{2.25}$$

$\mathbf{P}(k+1 \mid k)$ can be seen as a *priori* estimate state covariance matrix whereas $\mathbf{P}(k+1 \mid k+1)$ is a *posteriori* estimate state covariance [13].The Kalman gain matrix $\mathbf{W}$ is recursively found such that the *posteriori* error covariance matrix is minimized. Also $\mathbf{W}$ can be seen as a weighting factor that reflects the relative accuracy of the predicted state versus the new measurement [12]:

- If the new measurement is deemed to be "more accurate" (low variance) than the predicted state(large variance), then the filter gain will be relatively high.
- If the predicted state is deemed to be "more accurate" (low variance) than the new measurement, then the gain will be relatively low.

Figure 2.18 offers a complete block diagram of the operation of the filter with the set of involved a priori and a posteriori estimates.

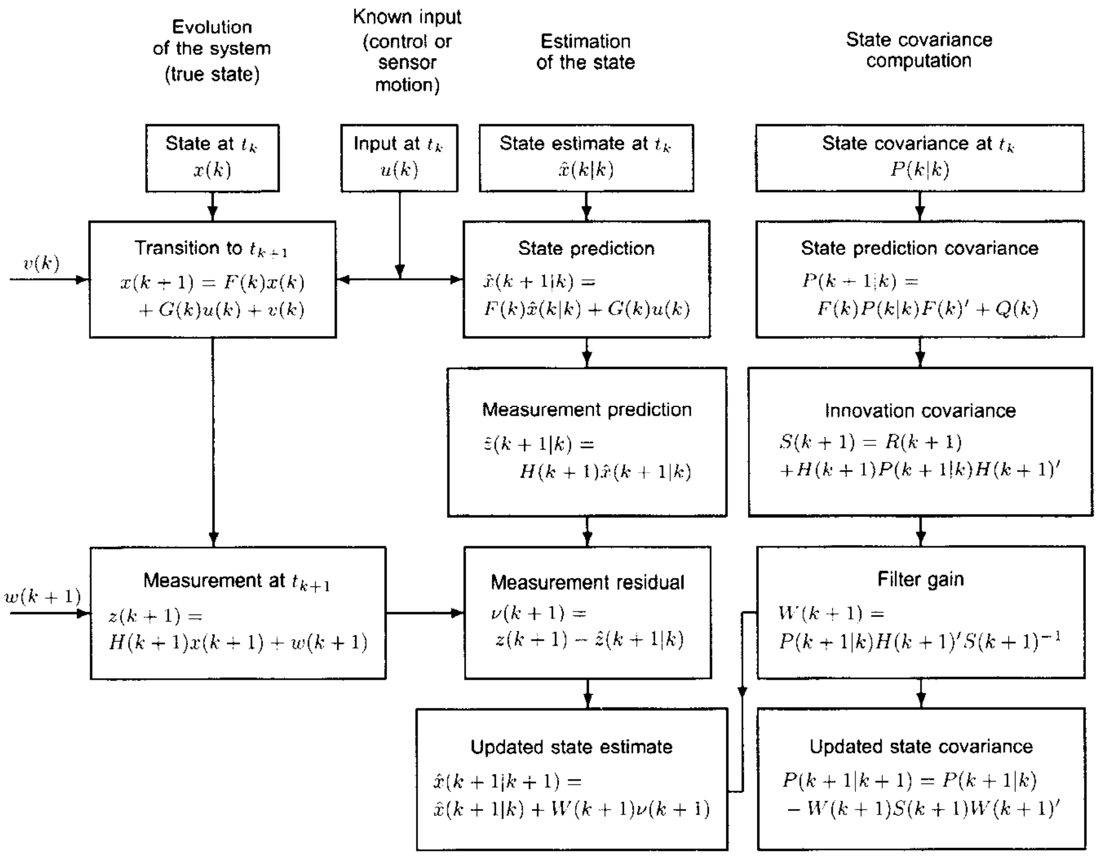

Evolution of the system (true state)
Known input (control or sensor motion)
Estimation of the state
State covariance computation
State at t_k $x(k)$
Input at t_k $u(k)$
State estimate at t_k $\hat{x}(k|k)$
State covariance at t_k $P(k|k)$
$v(k)$
Transition to t_{k+1} $x(k+1) = F(k)x(k) + G(k)u(k) + v(k)$
State prediction $\hat{x}(k+1|k) = F(k)\hat{x}(k|k) + G(k)u(k)$
State prediction covariance $P(k+1|k) = F(k)P(k|k)F(k)' + Q(k)$
Measurement prediction $\hat{z}(k+1|k) = H(k+1)\hat{x}(k+1|k)$
Innovation covariance $S(k+1) = R(k+1) + H(k+1)P(k+1|k)H(k+1)'$
$w(k+1)$
Measurement at t_{k+1} $z(k+1) = H(k+1)x(k+1) + w(k+1)$
Measurement residual $\nu(k+1) = z(k+1) - \hat{z}(k+1|k)$
Filter gain $W(k+1) = P(k+1|k)H(k+1)'S(k+1)^{-1}$
Updated state estimate $\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + W(k+1)\nu(k+1)$
Updated state covariance $P(k+1|k+1) = P(k+1|k) - W(k+1)S(k+1)W(k+1)'$

Figure 2.18: Kalman filter operation [12]

2.3.2 Extended Kalman Filter (EKF

In practice the state space and/or the observation equation can become nonlinear. In these cases the Kalman filter can not be applied since its closed expressions were obtained assuming that state space model and the observation equations are linear. An example where extended Kalman filter can be applied is the following: consider a tracking problem when the estimated measurements are both the range $\hat{R}[n] = R[n] + w_R[n]$ and the angle $\hat{\beta} = \beta + w_\beta$ formed between the 2-D coordinates $(r_x[n], r_y[n])$ that must be found. The relationship between the noisy measurements and the unknown parameters is shown in (2.30).

$$\begin{aligned} \hat{R}(k+1) &= \sqrt{r_x^2(k+1) + r_y^2(k+1)} + w_R(k+1) \\ \hat{\beta}(k+1) &= \arctan \tfrac{r_y(k+1)}{r_x(k+1)} + w_\beta(k+1), \end{aligned} \tag{2.26}$$

where $r_x(k+1)$ and $r_y(k+1)$ are directly related to the known velocity $(v_x(k+1), v_y(k+1))$, the time interval transcribed between samples Δ and the initial position $(r_x[0], r_y[0])$ of the mobile target in the form of:

$$\begin{aligned} r_x(k+1) &= v_x(k+1)\Delta + r_x[0] \\ r_y(k+1) &= v_y(k+1)\Delta + r_x[0] \end{aligned} \tag{2.27}$$

The equation (2.27) can be related to the first Gauss Markov process as:

$$\begin{aligned} r_x(k+1) &= r_x(k+1 \mid k) + r_x[0] \\ r_y(k+1) &= r_y(k+1 \mid k) + r_x[0] \end{aligned} \tag{2.28}$$

Clearly it can be seen that the expression in (2.26) is nonlinear in range and bearing. Therefore it cannot be expressed in the form of the following linear model as in the case of the Kalman filter:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{F}(k)\mathbf{x}(k) + \mathbf{G}(k)\mathbf{u}(k) \\ \mathbf{z}(k+1) &= \mathbf{H}(k+1)\mathbf{x}(k+1) + \mathbf{w}(k+1) \end{aligned} \tag{2.29}$$

Therefore the extended Kalman filter applies when the process/state model and/or observation/measurement model are nonlinear in the form of (2.30):

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{f}(\mathbf{x}(k)) + \mathbf{G}(k)\mathbf{u}(k) \\ \mathbf{z}(k+1) &= \mathbf{h}(\mathbf{x}(k+1)) + \mathbf{w}(k+1), \end{aligned} \tag{2.30}$$

being $\mathbf{f}$ a p-dimensional function and $\mathbf{h}$ an M-dimensional function. The dimensions of the remaining matrices and vectors are the same as before. Again $\mathbf{f}(\mathbf{x}[n-1])$ represents the theoretical model and $\mathbf{h}(\mathbf{x}[\mathrm{n}])$ is the transformation operation from the predicted state to the predicted measurement. Clearly MMSE cannot be applied directly because it applies to linear problems. The used approach is to linearize both equations with the first-order Taylor expansion so that the linearised state space model and the observation equation become linear:

$$\begin{gathered}\mathbf{x}(k+1) = \mathbf{F}(k)\mathbf{x}(k) + \mathbf{G}(k)\mathbf{u}(k) + (\mathbf{f}(\hat{\mathbf{x}}(k \mid k)) - \mathbf{F}(k)\hat{\mathbf{x}}(k \mid k)) \\ \mathbf{z}(k+1) = \mathbf{H}(k+1)\mathbf{x}(k+1) + \mathbf{w}(k+1) + (\mathbf{H}(\hat{\mathbf{x}}(k+1 \mid k)) - \mathbf{H}(k+1)\mathbf{x}(k+1 \mid k)).\end{gathered} \tag{2.31}$$

The new linearised equations in (2.31) differ from the original ones in (2.29) in that $\mathbf{F}$ is now time varying and both equations have known terms added to them. Therefore the set prediction and correction equations for the extended Kalman filter are described following [12]:

1. Prediction stage.

   State Prediction:
   $$\hat{\mathbf{x}}(k+1 \mid k) = \mathbf{f}[k, \hat{\mathbf{x}}(k \mid k), u(k)] \tag{2.32}$$

   State prediction covariance $(p \times p)$:
   $$\mathbf{P}(k+1 \mid k) = \mathbf{F}(k)\mathbf{P}[k \mid k)\mathbf{F}'(k) + \mathbf{Q}(k) \tag{2.33}$$

2. Correction stage:

   Residual covariance $(p \times M)$:
   $$\mathbf{S}(k+1) = \mathbf{R}(k+1) + \mathbf{H}(k+1)\mathbf{P}(k+1 \mid k)\mathbf{H}(k+1)' \tag{2.34}$$

   Filter gain $(p \times M)$:
   $$\mathbf{W}(k+1) = \mathbf{P}(k+1 \mid k)\mathbf{H}(k+1)'\mathbf{S}(k+1)^{-1} \tag{2.35}$$

   Updated state estimate:
   $$\hat{\mathbf{x}}(k+1 \mid k+1) = \hat{\mathbf{x}}(k+1 \mid k) + \mathbf{W}(k+1)(\mathbf{z}(k+1) - \mathbf{h}[k+1, \hat{\mathbf{x}}(k+1 \mid k)]) \tag{2.36}$$

   Updated state covariance $(p \times p)$:
   $$\mathbf{P}(k+1 \mid k+1) = \mathbf{P}(k+1 \mid k) - \mathbf{W}(k+1)\mathbf{S}(k+1)\mathbf{W}(k+1)' \tag{2.37}$$

being

$$\begin{aligned}\mathbf{F}(k) &= \tfrac{\partial \mathbf{f}}{\partial \mathbf{x}}|_{\mathbf{x}=\hat{\mathbf{x}}(k|k)} \\ \mathbf{H}(k+1) &= \tfrac{\partial \mathbf{h}(k+1)}{\partial \mathbf{x}}|_{\mathbf{x}=\hat{\mathbf{x}}(k+1|k)}\end{aligned} \tag{2.38}$$

Notice that in comparison with the Kalman filter the gain and covariance matrices must be computed on-line as they depend upon the state estimates through $\mathbf{F}(k)$ and $\mathbf{H}[(k+1)$. Moreover the term MSE matrix is not the real one since an approximation has been realized. To finish this section figure 2.19 offers a summary of the seen equations (2.32)-(2.37), the same as the Kalman filter shown in figure 2.18.

An example of EKF applied to vehicle tracking can be found in section 13.8 of [13]. Applications of both KF and EKF for target tracking are shown in chapter 3 and some simulations showing their behaviour for target tracking in a practical scenario can be shown in chapter 4.

2.4 Literature Review

This section is a collection of some of the works done by other entities which are related to both cooperative positioning and tracking in wireless sensor networks. Then this section is divided in three other sections: cooperative positioning, tracking and applications in WSN. The applications section is a general overview of where WSN can be used.

2.4.1 Cooperative Positioning in Wireless Sensor Networks

The average location error is an important parameter for comparison between different localization techniques or algorithms in the WSN localization field. The people in [51] present a home application based on centralized WSN and they show measured location errors. In this paper they discuss the design, implementation and evaluation of a RSS-based location determination system. The implementation includes three components: the location determination system which adapts and extends Motetrack [52] for in-home use, a location storage system (centralized architecture) that receive messages from the unknown nodes and a user interface to allow home applications to get both current and historical. The anchor nodes are placed in known positions and they send beacon messages periodically containing the power level in addition to other parameters. The unknown node is moving randomly as it is carried by home residents. The position computation is carried with two phases:

1. Initial data collection phase: at the deployment setup the mobile unknown sensor sends

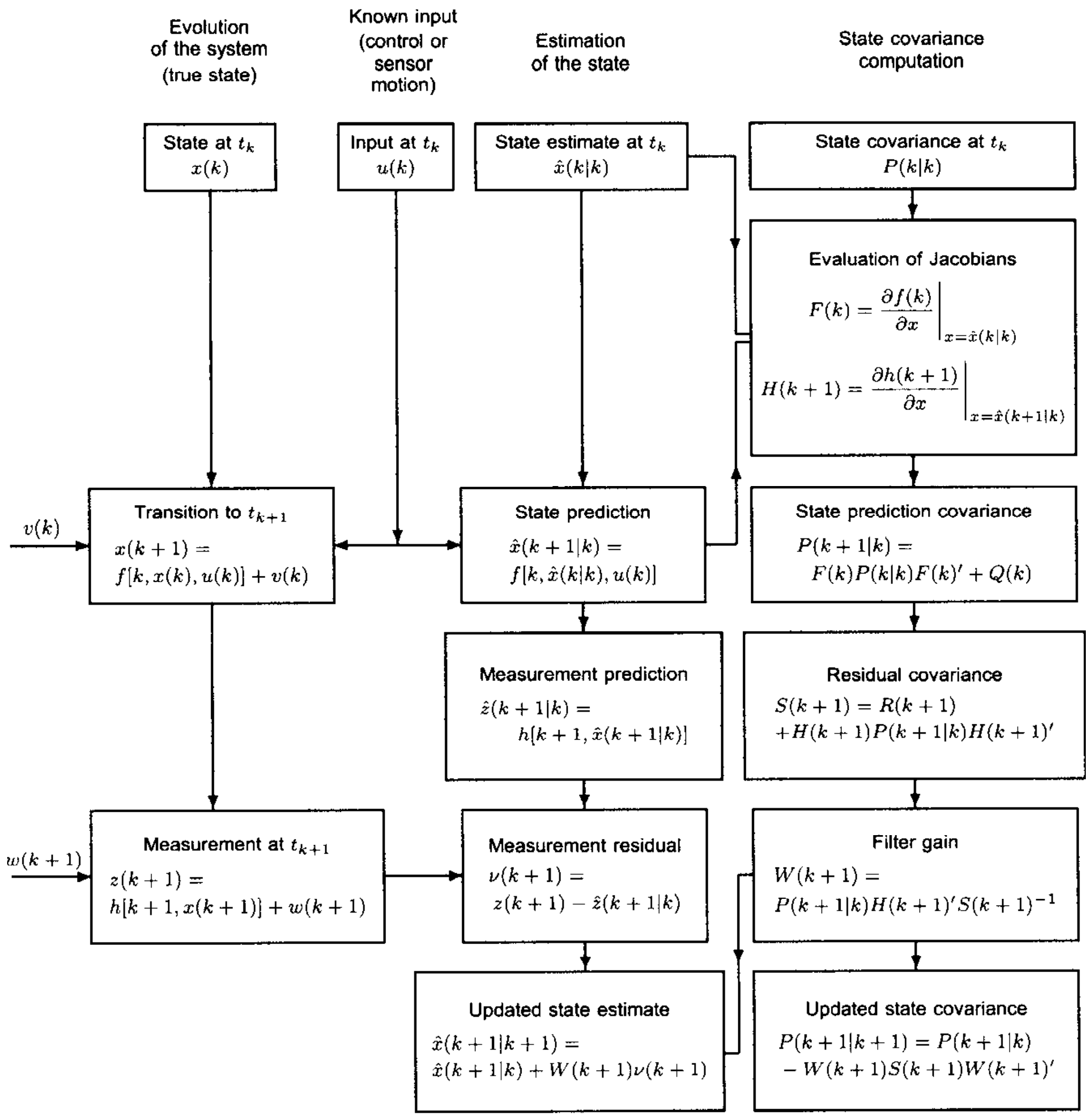

Evolution of the system (true state)
Known input (control or sensor motion)
Estimation of the state
State covariance computation
State at t_k $x(k)$
Input at t_k $u(k)$
State estimate at t_k $\hat{x}(k|k)$
State covariance at t_k $P(k|k)$
Evaluation of Jacobians
$F(k) = \left.\frac{\partial f(k)}{\partial x}\right|_{x=\hat{x}(k|k)}$
$H(k+1) = \left.\frac{\partial h(k+1)}{\partial x}\right|_{x=\hat{x}(k+1|k)}$
$v(k)$
Transition to t_{k+1} $x(k+1) = f[k, x(k), u(k)] + v(k)$
State prediction $\hat{x}(k+1|k) = f[k, \hat{x}(k|k), u(k)]$
State prediction covariance $P(k+1|k) = F(k)P(k|k)F(k)' + Q(k)$
Measurement prediction $\hat{z}(k+1|k) = h[k+1, \hat{x}(k+1|k)]$
Residual covariance $S(k+1) = R(k+1) + H(k+1)P(k+1|k)H(k+1)'$
$w(k+1)$
Measurement at t_{k+1} $z(k+1) = h[k+1, x(k+1)] + w(k+1)$
Measurement residual $\nu(k+1) = z(k+1) - \hat{z}(k+1|k)$
Filter gain $W(k+1) = P(k+1|k)H(k+1)'S(k+1)^{-1}$
Updated state estimate $\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + W(k+1)\nu(k+1)$
Updated state covariance $P(k+1|k+1) = P(k+1|k) - W(k+1)S(k+1)W(k+1)'$

Figure 2.19: Extended Kalman Filter operation [13]

to the location storage system all the received beacon messages including the computed RSSI values. All the beacon messages are recorded in a reference database which is used in the the normal operation.

2. Normal operation: during this phase the anchor nodes send periodically beacon messages in the same way as during the data collection phase. When mobile unknown sensors receive beacon messages they find a match of the received beacons against entries in the reference database. Then the node's current location is calculated as the center of the of the matching reference-points using the centroid approach explained in section 2.2.1.1.

The system uses Crossbow Mica2 and Mica2Dot sensors to provide 28th, 50th, 85th and 97th percentile location errors of under 1, 1.5, 2 and 3 m, good for room level accuracy.

The work by [51] estimates the position of mobile unknown nodes with fixed anchors. There are works based on range-free schemes that computes the location in a different way by means of geometrical approaches. For example a proposed technique based on several mobile anchor nodes is proved in [24]. Each anchor node moves inside the sensing field and broadcasts its current position periodically. The location estimation algorithm is based on the geometry conjecture (perpendicular bisector of a chord) [24]. What does this means? Consider that the transmission range of an unknown node is a circle and the centre of the circle are the coordinates of the unknown sensor node. Consider also that a mobile anchor node broadcasts its position in its beacon messages every sampling time while it is moving randomly through the sensor field and at time t_k the mobile anchor enters to the coverage circle of the unknown node. The unknown sensor stores in a database both the RSSI and position of the mobile anchor for each received beacon. Then the unknown sensor uses the the coordinates A and B associated to the first and last values of database in the circle in order to form a chord. Another point is needed in the circle form another chord. The conjecture states that a perpendicular bisector of a chord passes through the centre of the circle. If any two chords are obtained, the location of the sensor node can be easily computed as follows. First the set of linear equations corresponding to the perpendicular bisectors of each chord is formulated. The number of unknown is the same than the number of equations. Then the Cramer's rule is used to find the unknowns (x, y). Coming back to the paper [24] they demonstrates that this range-free technique is able to achieve fine-grained accuracy with an average location error of 0.74 m in a 100×100 m^2 sensor field. Their technique is robust against obstacles getting errors of 3.82 m in the average. Furthermore, this technique is distributed because the computation is performed locally and also it is scalable.

Despite of the good accuracy provided by [24] their algorithm is not energy efficient as being evaluated by [53]. It is said not energy efficient because the unknown sensor nodes are continuously listening for beacon messages containing the locations of the mobile anchors when only two beacons are needed. An energy efficient localization algorithm is more important than an algorithm that requires more power consumption although the provided accuracy is greater. In [53] is presented a range-based localization algorithm with the TDoA as a ranging metric and an ultrasound signal to obtain the signal propagation time travelling at the light speed and so the distance can be easily computed. Although the obtained accuracy is 0.82 m in a sensor field with dimensions 500×500 m^2 which is greater than the obtained by [24] their localization algorithm based on the Newton iteration method is more energy efficient because only three three anchor location points with a SNR higher than a specified threshold are recorded. By simulation the energy consumption due to packet reception is always below 0.5 megajoules with their proposed method whereas the energy consumption using the method in [24] is upper 1.75 megajoules, for any packet transmission interval.

The ranging metric used by [53] is more expensive than other cheaper ranging metrics such as RSSI contributing to the deployment cost of the network. In other words each node would need a TDoA external circuitry to compute the signal propagation time as well as an ultrasound transmitter/receiver. Localization with RSSI is a very cheap solution because nodes are able to measure the received signal strength which is related to the geometric distance from the anchor nodes. Although RSSI values are quite random due to shadowing and multipath effects it is possible to obtain better accuracy in good scenarios as demonstrated by [36]. They discuss and analyse some RSSI-based techniques studying different factors (antenna orientation, transmit power and frequency variation) that affect the measured RSSI values or the estimated distances. For the experimental tests they develop a prototype based on BTnode sensor nodes that includes the following components: a Bluetooth interface with a 433-915 MHz low power radio chip (5 dBm of transmission power) and the mobile robot system Robertino where one sensor node is located at the top acting as an anchor node. They focus to examine the short range RSSI deviations under standard conditions for indoor localization. The used ranging metric is the RSSI and the localization procedure is based on multilateration. They obtain location errors of 0.80 m in the worst cases and 0.54 m in the best cases both using linear regression on the RSSI data measured in an area of about 3.5×5.0 m^2.

2.4.2 Tracking in Wireless Sensor Networks

The presented literature right now is a general perspective on the different positioning techniques existent in WSN. A general case of positioning is tracking or navigation where the localization computation of a mobile unknown node position is carried on in each sampling time. In this case the unknown sensor node can move with a uniform motion, an acceleration motion or both together. There are several practical or theoretical papers focused on tracking in WSN and they are presented following.

At first a simulations based comprehensive studies comparing the performance of KF versus

EKF for target tracking in WSN are carried on by [54]. The authors of this paper compare the effectiveness, limitations and other related implementation issues in applying KF and EKF for target tracking in WSN assuming that the system dynamic model and/or the observation equation might be linear or nonlinear depending on the specific scenario. The work shown in the paper puts in practice KF and EKF in different scenarios: 1) the state space model is nonlinear and 2) nonlinear in the measurement equation taking into account that the observations can be of one-dimensional (range or bearing are measured) or two-dimensional (range and bearing). Their results show that the performance of EKF is poor in comparison with KF when the problem is one-dimensional. Notice that for the comparison only distances from sensors-to-anchors are used and multilateration is not performed.

There are other simpler techniques in addition to KF or EKF for localization and tracking. For example, [55] develops an MSE algorithm as an ML(*Maximum Likelihood*) estimator for the localization problem using the RSSI values (instead of the distances) measured by an unknown node in a WSN. Their results show that the proposed MSE method outperforms the traditional trilateration technique with a greatly improve on the accuracy of location estimation. Furthermore the paper shows that the accuracy for the outdoor environment is higher compared to that in indoor because there is less variability in the RSSI. Their results show that a 52% of the position estimates have an accuracy less than 1 m in a 40×40 m^2 sensor field whereas for the rest percentage the accuracy in the position estimates is up to 2 m. Finally they show that their simple method based on the RSSI can be used effectively to track the movement of an unknown node in a WSN by repeatedly running the MSE algorithm over a predefined period of time. This scheme is fully distributed as each sensor node is able to track its location using the received RSSI values from the other nodes.

The method proposed by [55] does not allow to obtain other target moving characteristics such as velocity ,acceleration, turn rates, etc. Moreover every anchor node must send every sampling time a signal to the target meaning that they must be active and consuming some energy for transmission. The people in [14] develop another tracking method based on IMM(*Interacting Multiple Model*) with the combination of the particle filtering algorithm for collaborative target tracking. When the object enters in a monitored region only the nodes sensing the target become active, form a cluster and choose that sensor to act as cluster head which RSSI from the target is the largest. The tracking algorithm is performed at the cluster head. When the target goes on moving, a new cluster and cluster head is formed dynamically with self-adaptation. They compare three hypotheses by means of the RMSE: the uniform motion, the uniform acceleration motion and the uniform motion together with the uniform acceleration motion. The best obtained accuracy in terms of RMSE is 0.58 m in the uniform motion case when the nodes are deployed uniformly in the sensor field whereas the worst accuracy is 1.35 m in the acceleration uniform case and a random deployment. The measures are carried on in a 100×100 m^2 sensor field.

The conclusion obtained by [54] that EKF provides poor accuracy with one-dimensional scenarios, when only distance can be measured is not true at all. In the other hand the complexity of IMM is higher than the EKF due to IMM is a set of kalman filters. The accuracy obtained by [14] have been improved by [10][56] using the EKF algorithm. The application in [10][56] is the tracking of pallet jacks that are moving in a warehouse. In this paper the authors present a technique to monitor the manual transportation processes of goods in a warehouse in order to update the database automatically. Tracking of forklift trucks or pallet jacks equipped with wireless sensor nodes is performed with nanLOC sensor nodes. To obtain range measurements the used ranging metric is RToA. The initial state estimate is computed with trilateration as the first observation and tracking is achieved with an EKF. Their experimental results show an accuracy better than 0.40 m in both x and y coordinates in the 71% of the measurements in a measuring field with the dimensions 9.48×3.25 m^2. Some deviations about 1.38 m, 1.33 m are obtained due to the blockage of the LOS.

The work shown by [57] is a simulation of the IMM for tracking maneuvering vehicles, something which is quite related this master thesis. But instead of using EKF they prefer to use UKF(*Unscented Kalman Filter*) for the turns because in accordance to their opinion EKF has some drawbacks in nonlinear systems such as the approximation of a non-Gaussian density by a Gaussian density. By simulations they show that UKF performance regarding the RMSE of both the position and velocity is much less when using UKF than EKF. For the straight trajectories a simple Kalman filter is used. However the complexity grade of UKF is greater than EKF.

A design of the IMM for target tracking using both filters KF and EKF is given in chapter 3.

2.4.3 Applications using Wireless Sensor Networks

By now several applications that use WSN can be found in the literature. One interesting application is the monitoring concentration of carbon dioxide (CO2) gas in areas of interest within a VSN *Vehicular Sensor Network*) such as the proposed by [58]. In this work they employ Zigbee-based Jennic motes connected to the available vehicle GPS. The sensors send the CO2 record via short GSM messages to the FC integrated with GoogleMaps as a user interface. This scenario is centralized but not cooperative since the application does not need a cooperative strategy as all the sensor nodes have to send its sensing data to the FC. In the other hand cooperative strategy is useful in localization/detection-oriented applications.

Another centralized application that uses WSN is the implementation of an intelligent car park management system [59]. Their work consists to implement a software application to detect and monitor the occupation of the parking slots using low-cost sensor nodes. The status of the parking lot is reported periodically to a database via the WSN and its gateway. Through a management system it can guide/inform in an easy way the users to find a free parking slot in a urban street network. They have implemented a prototype using: commercial MPR2400 crossbow motes with MTS310 sensor, a data acquisition board (equipped with light, temperature, acoustic and a sounder sensor) and a MIB510 as an interface board acting as a gateway. As a database they use PosgreSQL and as a WSN monitoring application they use MOTE-VIEW. The network protocol adopted by the mote in their system is XMesh.

Chapter 3

System Design

The purpose of this chapter mainly based on [12] is to show the set of models used by the algorithms KF and EKF for a target tracking application in a wireless sensor network. All the content that appears in this chapter has been developed in Matlab code and integrated in a global GUI Matlab simulator. This chapter deals to show the used models for both KF and EKF which can be summarized below:

- A nearly constant velocity model for the **uniform motion**, implemented as a WNA (white noise acceleration) model with low-process noise variance.
- A maneuvering model implemented as a *nearly "coordinated turn model"* or known as uniform acceleration motion.

The both words *state* and *process* which appear along this chapter refer to the *statespacemodel*.

3.1 KF with the nearly constant velocity model

Consider the system with the following state vector:

$$\mathbf{x} = \begin{bmatrix} x \\ v_x \\ y \\ v_y \end{bmatrix} \tag{3.1}$$

with v_x and v_y the velocities along the x and y dimension, which evolves according to

$$\mathbf{x}(k+1) = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} T^2/2 & 0 \\ T & 0 \\ 0 & T^2/2 \\ 0 & T \end{bmatrix} \mathbf{u}(k) \quad n = 1, 2, \ldots \tag{3.2}$$

with a random initial condition $\mathbf{x}(0)$. The equation (3.2) represents a two-dimension uniform motion with the state vector (3.1) sampled at time intervals T and a constant acceleration noise $\mathbf{u} \sim \mathcal{N}(0, \sigma_u^2)$. This random noise $\mathbf{u}$ is a zero mean white sequence with variance σ_u^2 that models the small acceleration produced by external unavoidable factors such as wind change, small variations in the car accelerator pedal, etc. Values of σ_u^2 in practical scenarios can be for instance $0.1 m/s^2$ or even $2 m/s^2$ [12]. This acceleration noise $\mathbf{u}$ affects both the (x, y) positions and (v_x, v_y) velocities which is formulated with the matrix on the right side. The right term in (3.2) that is adding to the left term is called as process noise. It can be understood as follows. Notice that (3.2) has the form of a first-order Markov process:

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{G}(k)\mathbf{u}(k), \tag{3.3}$$

where $\mathbf{G}$(k)$\mathbf{u}$(k) is the process noise with an appropriate covariance matrix $\mathbf{Q}$ which depends on σ_u^2 being it a design parameter that must be known. The process noise covariance matrix $\mathbf{Q}$ has the following form:

$$\mathbf{Q} = \mathbf{G}(k)\sigma_u^2 \mathbf{I} \mathbf{G}(k)', \tag{3.4}$$

where $\sigma_u^2 \mathbf{I}$ is the covariance matrix $\mathbf{U}$ of the acceleration noise vector $\mathbf{u}$ with the form shown below:

$$\mathbf{U} = \begin{bmatrix} \sigma_u^2 & 0 \\ 0 & \sigma_u^2 \end{bmatrix} \tag{3.5}$$

Section 4 deals with the simulation of Kalman Filters for different values of σ_u^2.

Regarding the measurements they consist of the state's position components corrupted by additive noise:

$$\mathbf{z}(k+1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{x}(k+1) + \mathbf{w}(k+1) \tag{3.6}$$

where the measurement noise is assumed zero-mean with variance $\sigma_w^2 = 1$. Also note that (3.6) follows the same structure seen in section (2.3.1).

The two noises **v** and **w** are mutually independent. Furthermore the filter is initialized according to section 3.3 which explains how to initialize both the initial state vector and the initial process covariance matrix.

3.2 EKF with the Nearly coordinated turn model

The turn of a target usually follows a pattern known as **coordinated turn (CT)** - characterized by a constant turn rate and constant velocity magnitude. In practice the target velocity is not constant at all due to some small deviations. Hence an external noise as the modeling error is taken into account and thus resulting to the nearly coordinated turn model. The **CT** *model* is necessarily a nonlinear system if the turn rate is not a known constant. Thus the vector denoted in (3.1) is increased by one more component -the turn rate Ω- resulting the following augmented state vector:

$$\mathbf{x} = [\ x \quad v_x \quad y \quad v_y \quad \Omega\ ]' \tag{3.7}$$

the *nearly coordinated turn model* is then given by

$$\mathbf{x}(k+1) = \begin{bmatrix} 1 & \frac{sin\Omega(k)T}{\Omega(k)} & 0 & -\frac{1-cos\Omega(k)T}{\Omega(k)} & 0 \\ 0 & cos\Omega(k)T & 0 & -sin\Omega(k)T & 0 \\ 0 & \frac{1-cos\Omega(k)T}{\Omega(k)} & 1 & \frac{sin\Omega(k)T}{\Omega(k)} & 0 \\ 0 & sin\Omega(k)T & 0 & cos\Omega(k)T & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} \frac{1}{2}T^2 & 0 & 0 \\ T & 0 & 0 \\ 0 & \frac{1}{2}T^2 & 0 \\ 0 & T & 0 \\ 0 & 0 & T \end{bmatrix} \mathbf{v}(k) \tag{3.8}$$

Note that the process noise v in (3.2) has different dimension from the one in (3.8). Furthermore the last row of the matrix that is multiplying to $\mathbf{x}(k)$ means that the turn rate is constant during the maneuver. Regarding the measurement equation it is similar to the one in (3.6) but with an extra column for the turn rate

$$\mathbf{z}(k+1) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{x}(k+1) + \mathbf{w}(k+1) \tag{3.9}$$

where **w** is the measurement noise.

Since the model in (3.8) is nonlinear with Ω, the estimation of the state (3.7) will be done with the EKF. In this case the system dynamic model has the form shown in (3.10).

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k)) + \mathbf{G}(k)\mathbf{u}(k) \tag{3.10}$$

In this case the EKF transition function is $\mathbf{f}(\mathbf{x}(k))$. It is not the transition matrix as in the case of the KF but it is a nonlinear function evaluated over the state vector $\mathbf{x}$. This nonlinear function is expanded into a first-order Taylor series. The first derivative denoted by $\mathbf{F_{EKF}}$(Jacobian of $\mathbf{f}(\mathbf{x}(\mathrm{k}))$ evaluated at the latest estimate of the state) is used in the on-line state covariance computation.

$$\mathbf{F_{EKF}}(k) = [\nabla_x f(\mathbf{x})']' \,|_{\mathbf{x}=\hat{\mathbf{x}}(k)(k)} = \begin{bmatrix} 1 & \frac{sin\hat{\Omega}(k)T}{\hat{\Omega}(k)} & 0 & -\frac{1-cos\hat{\Omega}(k)T}{\hat{\Omega}(k)} & f_{\Omega,1}(k) \\ 0 & cos\hat{\Omega}(k)T & 0 & -sin\hat{\Omega}(k)T & f_{\Omega,2}(k) \\ 0 & \frac{1-cos\hat{\Omega}(k)T}{\hat{\Omega}(k)} & 1 & \frac{sin\hat{\Omega}(k)T}{\hat{\Omega}(k)} & f_{\Omega,3}(k) \\ 0 & sin\hat{\Omega}(k)T & 0 & cos\hat{\Omega}(k)T & f_{\Omega,4}(k) \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.11}$$

being $f_{\Omega,1}(k)$ the partial derivatives with respect to Ω which are calculated as shown in (3.12)

$$\begin{bmatrix} f_{\Omega,1}(k) \\ f_{\Omega,2}(k) \\ f_{\Omega,3}(k) \\ f_{\Omega,4}(k) \end{bmatrix} = \begin{bmatrix} \frac{(\cos\hat{\Omega}(k)T)T\hat{v}_x(k)}{\hat{\Omega}(k)} - \frac{(\sin\hat{\Omega}(k)T)\hat{v}_x(k)}{\hat{\Omega}(k)^2} - \frac{(\sin\hat{\Omega}(k)T)T\hat{v}_y(k)}{\hat{\Omega}(k)} - \frac{(-1+cos\hat{\Omega}(k)T)\hat{v}_y(k)}{\hat{\Omega}(k)^2} \\ -(\sin\hat{\Omega}(k)T)T\hat{v}_x(k) - (\cos\hat{\Omega}(k)T)T\hat{v}_y(k) \\ \frac{(sin\hat{\Omega}(k)T)T\hat{v}_x(k)}{\hat{\Omega}(k)} - \frac{(1-\cos\hat{\Omega}(k)T)\hat{v}_y(k)}{\hat{\Omega}(k)^2} + \frac{(\cos\hat{\Omega}(k)T)T\hat{v}_y(k)}{\hat{\Omega}(k)} - \frac{(\sin\hat{\Omega}(k)T)\hat{v}_y(k)}{\hat{\Omega}(k)^2} \\ (\cos\hat{\Omega}(k)T)T\hat{v}_x(k) - (\sin\hat{\Omega}(k)T)T\hat{v}_y(k) \end{bmatrix} \tag{3.12}$$

Summarizing the state prediction and state prediction covariance in the EKF are

$$\begin{aligned} \hat{\mathbf{x}}[n \mid n-1] &= f(\hat{\mathbf{x}}[n-1 \mid n-1]) \\ \mathbf{P}[n \mid n-1] &= \mathbf{F_{EKF}}(k)\mathbf{P}[n-1 \mid n-1]\mathbf{F_{EKF}}(k)' + \Gamma_{CT}(k)\mathbf{Q}(k)\Gamma_{CT}(k)' \end{aligned} \tag{3.13}$$

Regarding the turn rate angle $\hat{\Omega}$, its initial value can be zero since a priori one does not know if the target is turning and with which angular speed. Regarding the sign, the turn is to the left if $\hat{\Omega} > 0$ and to the right if $\hat{\Omega} < 0$. The EKF filter is able to estimate $\hat{\Omega}$ from the received measurements. Note that in the case when $\hat{\Omega} = 0$ the form of $\mathbf{F_{EKF}}$ given in (3.11) must be replaced with the one shown in (3.14)

$$\mathbf{F_{EKF}}(k)\mid_{\hat{\Omega}(k)=0}= \begin{bmatrix} 1 & T & 0 & 0 & -\frac{1}{2}T^2\hat{v}_y(k) \\ 0 & 1 & 0 & 0 & -T\hat{v}_y(k) \\ 0 & 0 & 1 & T & \frac{1}{2}T^2\hat{v}_x(k) \\ 0 & 0 & 0 & 1 & T\hat{v}_x(k) \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

3.3 Initialization of State Estimators

The initialization of the discussed tracking algorithms is a key point in the estimator behaviour. The initialization of state estimators covers basically two aspects:

1. Initialization of state vector: basically it refers to the initialization of $x, v_x, y, \text{and}, v_y$. Usually it is better to initialize the estimator with the first measurements. Therefore with T being the sampling interval and taking into account 2 successive measurements then the following two dimensional positions and velocities can be initialized:

   <u>Two dimensions</u>

$$\begin{aligned} [x_0, y_0] &= [\mathbf{z}[0], \mathbf{z}[1]] \\ v_{x0} &= \tfrac{z[0]-z[-1]}{T} \\ v_{y0} &= \tfrac{z[1]-z[-2]}{T} \end{aligned} \tag{3.15}$$

   The equation (3.15) means that the initialization is based on the first two position measurements $z[0]\text{and}z[-1]$ for the x-dimension and $z[1]\text{and}z[-2]$ for the y-dimension.

2. Initialization of state covariance matrix: Assuming that the measurement noise $w \sim \mathcal{N}(0, r)$ with variance r then the covariance of the process is initialized as shown in (3.16).

   <u>Two dimensions</u>

$$\mathbf{P}[0 \mid 0] = \begin{bmatrix} r & \frac{r}{T} & 0 & 0 \\ \frac{r}{T} & \frac{2r}{T^2} & 0 & 0 \\ 0 & 0 & r & \frac{r}{T} \\ 0 & 0 & \frac{r}{T} & \frac{2r}{T^2} \end{bmatrix} \tag{3.16}$$

   On the one hand notice that the dimensions of the process covariance matrix is 4×4. This covariance must be extended to 5×5 for the EKF with zeros at the last row and column except that the element in the position $(5, 5)$ is nonzero, but it is the variance of the turn rate Ω. The initial value of the turn rate variance, σ^2_Ω, is a design parameter that depends on the scenario. Usually low values are chosen. For the simulations carried on in the next chapter σ_{Ω}^2 is initialized to 0.01.

On the other hand the initialization procedure is called *two-point differencing* and it guarantees consistency of the initialization of the filter.

3.4 The Interacting Multiple Model Estimator

The Interacting Multiple Model (IMM), brought forward by H.A.P.Blom in 1984 and widely used in target tracking was built as one of the techniques (GPB1(*Generic pseudo-Bayesian estimator of first order*) and GPB2 are two other techniques in the literature[12] that have the follow approach) that allow to combine several filters such as those discusses previously (KF and EKF) at the same time in one single algorithm. The use the IMM depends on the application scenario, for example, the tracking of a target that is moving. The trajectory of the target can be composed by straight stretchs and turns. In stretchs where the target goes straight on the best to do is to apply a linear model since the error of the model will be small. However in the turns, one approach is to consider a higher noise the model due to the real trajectory deviates from the linear model with small noise. Another approach is to use models that have been designed exclusively for the turns such as the CT(*Coordinated Turn*) model. Figure 3.1 shows the main idea of the IMM functionality.

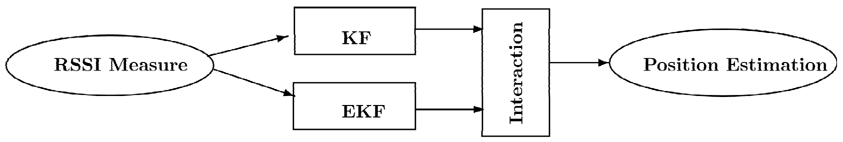


Figure 3.1: The IMM estimator as the combination with two filters [14].

In order to consider several filters at the same time, the IMM is able to quantify from the received measurements the likelihood for each of the models at each sampling time. Hence a higher weight will be given to the model with higher likelihood, i.e. the model that best approximates to the reality. The IMM takes into account the model probabilities at each time instant - i.e. each model is weighted and has a likelihood to be true. Thus the estimation given by the IMM is a weighted mixture (a combination) between the produced estimations by all the interacting models.

The structure of the IMM algorithm is

$$(N_e; N_f) = (r; r) \tag{3.17}$$

where N_e is the *number of estimates* at the start of the cycle of the algorithm, N_f is the *number* *of filters* in the algorithm and r corresponds with the number of models (or filters) taking into account. Note that having r models is equivalent to having r filters as each filter is related with one different model.

The full functionality of the IMM algorithm is shown in figure 3.2 and a description of each stage is introduced following.

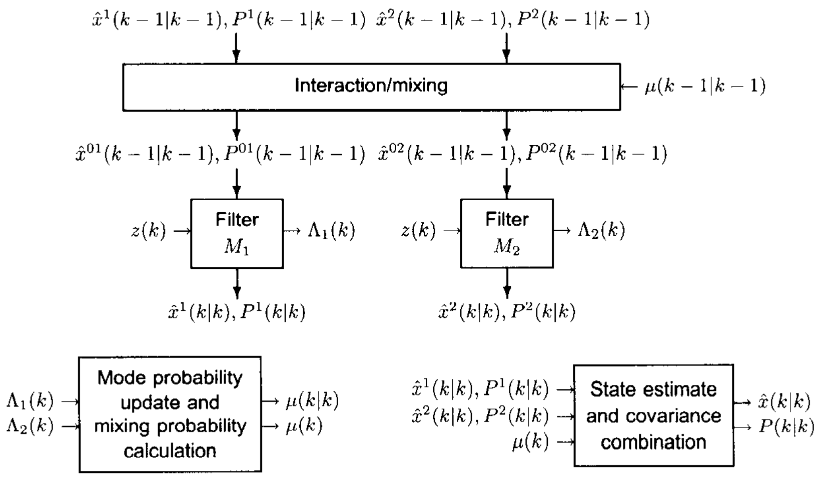

$\hat{x}^1(k-1|k-1), P^1(k-1|k-1)$
$\hat{x}^2(k-1|k-1), P^2(k-1|k-1)$
Interaction/mixing
$\mu(k-1|k-1)$
$\hat{x}^{01}(k-1|k-1), P^{01}(k-1|k-1)$
$\hat{x}^{02}(k-1|k-1), P^{02}(k-1|k-1)$
$z(k)$
Filter M_1
$\Lambda_1(k)$
$z(k)$
Filter M_2
$\Lambda_2(k)$
$\hat{x}^1(k|k), P^1(k|k)$
$\hat{x}^2(k|k), P^2(k|k)$
$\Lambda_1(k)$
$\Lambda_2(k)$
Mode probability update and mixing probability calculation
$\mu(k|k)$
$\mu(k)$
$\hat{x}^1(k|k), P^1(k|k)$
$\hat{x}^2(k|k), P^2(k|k)$
$\mu(k)$
State estimate and covariance combination
$\hat{x}(k|k)$
$P(k|k)$

Figure 3.2: Interacting Multiple Model operation [12].

First the inputs of the IMM at each time interval (k) are the state vectors and covariance matrices computed by each model at the previous time $(k-1)$. Thus $\hat{\mathbf{x}}^1$ and $\mathbf{P}^1$ is the state vector estimation and the process covariance matrix associated to the KF (the linear model). Also $\hat{\mathbf{x}}^1$ and $\mathbf{P}^1$ is the state vector estimation and the process covariance matrix associated to the EKF (the CT model). Then three stages are carried on to produce an more accuracy estimate:

1. Interaction/Mixing: this stage computes for each $j = 1, \dots, r$ and for each time sample $(k-1)$ the mixed estimates denoted by $\hat{\mathbf{x}}^{\mathbf{0j}}(\mathbf{k-1} \mid \mathbf{k-1})$ and the mixed state covariance matrix denoted by $\mathbf{P}^{\mathbf{0j}}$ from the previous filter output. Both the mixed estimates and the mixed covariance matrix are computed with the *mixing probabilities* which are denoted by $\mu(k-1 \mid k-1)$. These mixing probabilities are computed using the models transition probabilities $\mathbf{p}_{ij}$, the model probabilities $\mu(k-1)$ and the normalizing constants $\bar{c}$. On the one hand $\mathbf{p}_{ij}$ is a matrix containing the probabilities of switching from one model to another model; they are governed by a Markov chain as the one shown in figure 3.3. Figure 3.3 shows two states (or two models) and the transition probabilities between states (models) p_{12} and p_{21} as well as the probabilities to remain at the same state, p_{11} and p_{22}.

   On the other hand the vector $\mu_i(k-1)$ is computed *on-line* using the likelihoods of each model. In other words, the values obtained evaluating the received measurements vector to

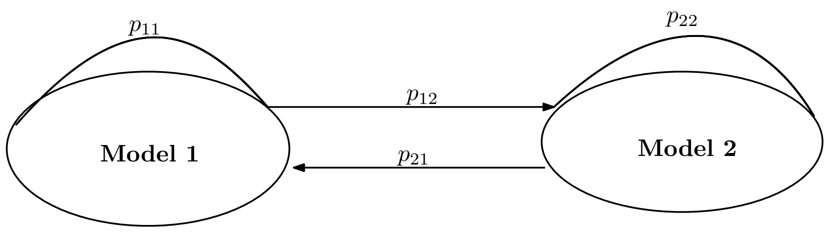


Figure 3.3: The IMM estimator with two filters [15].

the probability density function of each model are used to compute the mode probabilities.

2. Filtering: this stage deals with the execution in parallel of several filters that follows either the same model with different process noise variance or different models. Every mixed estimate $\hat{\mathbf{x}}^{\mathbf{0j}}$ and mixed state covariance $\mathbf{P}^{\mathbf{0j}}$ computed in the previous stage is an input parameter to one of the filters. The output of each filter are the state vector $\hat{\mathbf{x}}^{j}(k \mid k)$, the state covariance $\mathbf{P}^{j}(k \mid k)$ and the model probability $\Lambda_j(k)$(the corresponding value of its likelihood function for the received measurements vector $\mathbf{z}(k)$).

3. Mode Probability update and mixing probability calculation: The mode probabilities as well as the mixing probabilities are updated on-line. The mode probabilities $\mu(k-1)$ are calculated *on-line* evaluating the received measurements vector to the multi-variate probability density functions of each model. The mode probabilities tell us the probability that a model is more true than another. Therefore they can be seen as weighting factors.

4. State estimate and covariance combination: the output of all filters are combined using the updated mode probabilities. In other words, the output of the IMM is a weighted combination between the outputs of all the models. This combination is only for output purposes, it is not part of the algorithm recursions.

The set of equations that involve each of the IMM stages are shown in the following section 3.4.1.

3.4.1 The algorithm

One recursion of the algorithm consists of the following:

1. **Calculation of the mixing probabilities:** $(i, j = 1, ..., r)$. The probability that a certain mode was used at time instant $k-1$ given that current model M_j at time instant k.

$$\boxed{\mu_{i|j}(k-1 \mid k-1) = \frac{1}{\bar{c}_j} p_{ij}\mu_i(k-1) \quad i, j = 1, \ldots, r} \tag{3.18}$$

where the normalizing constants are

$$\bar{c}_j = \sum_{i=1}^{r} p_{ij}\mu_i(k-1) \quad j = 1, \ldots, r \tag{3.19}$$

2. **Mixing** ($j = 1, \ldots, r$). Starting with $\hat{\mathbf{x}}^i(k-1 \mid k-1)$, one computes the mixed initial state vector and state covariance for the filter matched to the current model $M_j(k)$ as

$$\boxed{\hat{\mathbf{x}}^{0j}(k-1 \mid k-1) = \sum_{i=1}^{r} \hat{\mathbf{x}}^i(k-1 \mid k-1)\mu_{i|j}(k-1 \mid k-1) \quad j = 1, \ldots, r} \tag{3.20}$$

The mixed initial state covariance is computed as follows

$$\boxed{\begin{array}{rcl} \mathbf{P}^{0j} & = & \sum_{i=1}^{r} \mu_{i|j}(k-1 \mid k-1) \left\{ \mathbf{P}^i(k-1 \mid k-1) \right. \\ & & +[\hat{\mathbf{x}}^i(k-1 \mid k-1) - \hat{\mathbf{x}}^{0j}(k-1 \mid k-1)] \\ & & \left. \cdot[\hat{\mathbf{x}}^i(k-1 \mid k-1) - \hat{\mathbf{x}}^{0j}(k-1 \mid k-1)]' \right\} \\ & & j = 1, \ldots, r \end{array}} \tag{3.21}$$

3. **Mode-matched filtering** ($j = 1, \ldots, r$). The estimates in (3.20) and (3.21) are used as an input to the filter j. The likelihood functions corresponding to the r filters are computed using both the mixed initial state vector and state covariance as

$$\boxed{\Lambda_j(k) = \frac{1}{(2\pi)^{1/2}|\mathbf{S}^j|^{1/2}} exp\left(-\frac{1}{2}(\mathbf{z} - \mathbf{z}^j)'(\mathbf{S}^j)^{-1}(\mathbf{z} - \mathbf{z}^j\right),} \tag{3.22}$$

where $\mathbf{z}$ are the position measurements, $\mathbf{z}^j$ are the predicted measurements by each filter and $\mathbf{S}^j$ are the innovance or residual covariances of each filter. The equation (3.22) says that each filter has a pdf(*probability density function*) centred at its prediction $\hat{\mathbf{z}}^j$ with a residual covariance $\mathbf{S}^j$. The measurement $\mathbf{z}$ is evaluated for each pdf resulting a probability of occurrence $\Lambda_j(k)$ at the time sample k. As an example, if $\Lambda_1(k) > \Lambda_2(k)$ means that the predicted measurement $\hat{\mathbf{z}}$ by the model 1 (in our case would be the linear model) is closer to the received measurement $\mathbf{z}$. As a result a higher weight is given to the output estimates from model 1.

4. **Mode probability update** ($j = 1, \ldots, r$). This is calculated as follows:

$$\boxed{\mu_j(k) = \frac{1}{c}\Lambda_j(k)\bar{\mathbf{c}}_j \quad j = 1, \ldots, r} \tag{3.23}$$

where $\bar{c}_j$ is the expression from (3.19) and

$$\boxed{c = \sum_{j=1}^{r} \Lambda_j(k)\bar{\mathbf{c}}_j} \tag{3.24}$$

is the normalization constant for (3.23).

5. **Estimate and covariance combination** The outputs of each filter are combined (mixed) in accordance to the mixture equations (3.20) and (3.21) with the mode probabilities factors.

$$\boxed{\hat{\mathbf{x}}(k \mid k) = \sum_{j=1}^{r} \hat{\mathbf{x}}^j(k \mid k)\mu_j(k)} \tag{3.25}$$

$$\boxed{\mathbf{P}(k \mid k) = \sum_{j=1}^{r} \mu_j(k) \left\{\mathbf{P}^j(k \mid j) + [\hat{\mathbf{x}}^j(k \mid k) - \hat{\mathbf{x}}(k \mid k)][\hat{\mathbf{x}}^j(k \mid k) - \hat{\mathbf{x}}(k \mid k)]'\right\}} \tag{3.26}$$

3.4.2 Examples with the IMM Estimator

This section shows two configurations of the IMM estimator with a simple target route that is composed by straight lines, left and right turns.

Configuration 1: A scenario of dimensions 160 $m\times$ 90 m and a constant velocity of $v = 30$ Km/h. The Kalman initial state vector is shown in (3.27) whereas for the Extended Kalman, it is the same but with $\Omega = 0$ at the end. The sampling time is $T = 60$ ms and the position measurements have a standard deviation of $\sigma_z = 1.5$ m. The true track of the target is from the left to the right. The turn left radius is 10 m (the target turns trough the perimeter of a circle of radius 10 m) and the turn right radius is 2 m.

$$\mathbf{x}_0 = \begin{bmatrix} 20 & 0 & 30 & 0 \end{bmatrix} \tag{3.27}$$

In this configuration the following filters are used by the IMM-CT estimator (*IMM-Cordinated Turn*)

1. One Kalman Filter with a constant velocity model with $v = 30$ Km/h and a low level noise with standard deviation of $\sigma_u^2 = 0.1$ m/s^2 that models the the uniform motion model.

2. An Extended Kalman Filter with a nearly coordinated turn model with the following process acceleration noise covariance matrix:

$$\mathbf{V} = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.2 \end{bmatrix} \tag{3.28}$$

where the elements in the diagonal are the process noise standard deviations for the linear ($0.5\ m/s^2$) and turn ($0.2 rad/s^2$) portions of the state, respectively.

Furthermore the initial mode transition probability matrix $\pi_{\mathbf{CT}}$ is

$$\pi_{\mathbf{CT}} = \begin{bmatrix} 0.95 & 0.06 \\ 0.10 & 0.90 \end{bmatrix} \tag{3.29}$$

The initial estimates are based on section 3.3; that is, the initial position estimate is the first measured position and the initial velocity/turn rate is zero. The initial model probabilities are set to

$$\mu_{\mathbf{0}} = [0.5\,0.5]' \tag{3.30}$$

The results of the tracking as well as the true track can be seen in figure 3.4 shown below.

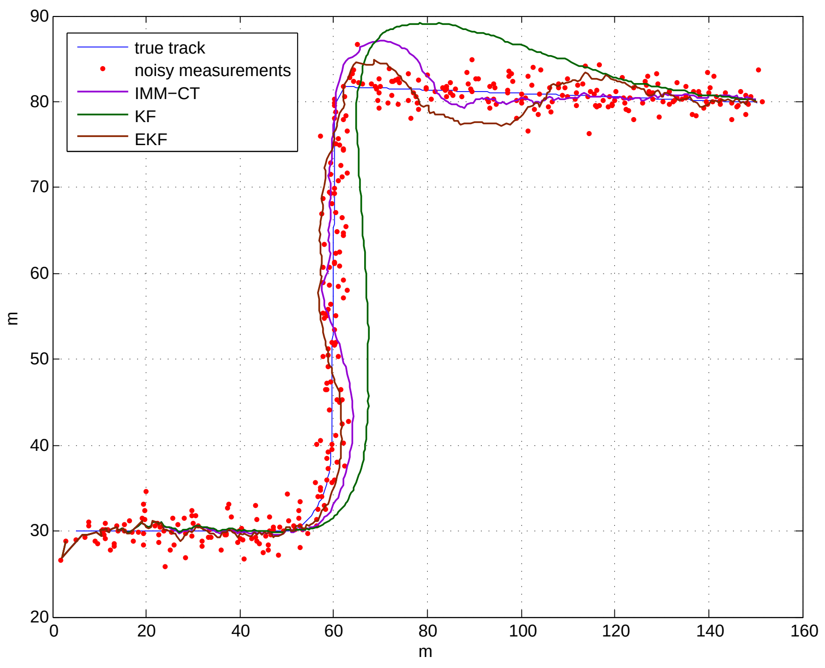


Figure 3.4: Comparison of KF, EKF and IMM-CT with $\sigma_z = 1.5\,m$. Configuration 1

It can be seen that the estimations given by the IMM are between the KF and the EKF estimations due to the IMM is a weighted combination between the KF and the EKF (the sum of the weights or mode probabilities are 1). After the second turn, the IMM tends to the true track much faster than the other two estimators. Table 3.1 provides a comparison between the three estimators in terms of RMSE position error given the (RMSE_Z) position error, obtained by multilateration.

RMSE_z	1.8977 m
RMSE_KF	4.6848 m
RMSE_EKF	1.7979 m
RMSE_IMM	1.91 m

Table 3.1: RMSE errors of the different tracking estimators

One can see that the performance of the IMM estimator is much better than the KF and worst than EKF and multilateration. It is due to that the simulations are performed in quasi ideal conditions, i.e. small measurements variance is considered. Next the performance will be evaluated when noise higher, $\sigma_z >> 1.5\,m$. It will be seen that for higher noise variance then EKF begins to fail but in contrast the IMM tries to follow the true trajectory of the target.

The turns are generated using the CT model. The number of samples in the turns depends on the sampling time, the radius of the right bend and the velocity. As the turn to the right has smaller radius than the turn to the left then only two samples are obtained with the chosen parameters. As a result with only two samples in the right turn the performance of all estimators are poor around this region due to the turn is quite sudden.

Figure 3.5 represents the EKF mode probability which is proportional to the EKF likelihood at each sample. One can see the following:

- initially the EKF mode probability is 0.5 and it starts to decrease upto sample $k-1 = 100$.
- The only regions with a quasi-constant EKF mode probability between $0.3 - 0.4$ are the first and the last. In those regions the KF in the linear model has greater likelihood. It is because the linear model fits better to the true trajectory which is straight (linear). Remind that the sum of EKF and KF mode probabilities are 1.
- The higher EKF mode probabilities is due to that the EKF in the CT model is more accurate than the KF in the linear model, thus providing higher likelihoods. The cases where the EKF gives position estimates with more accuracy than the KF are in the turns. Thus the two peaks corresponds to the left and right turns.

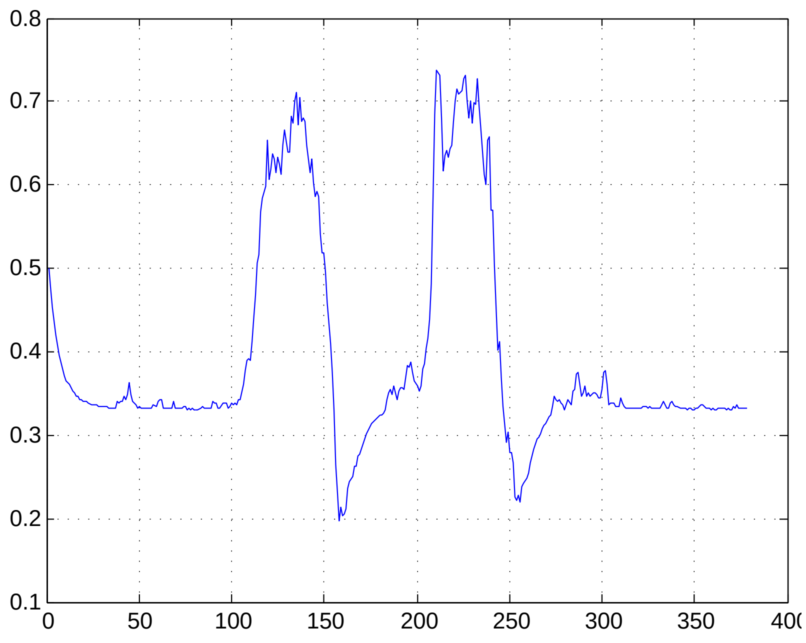
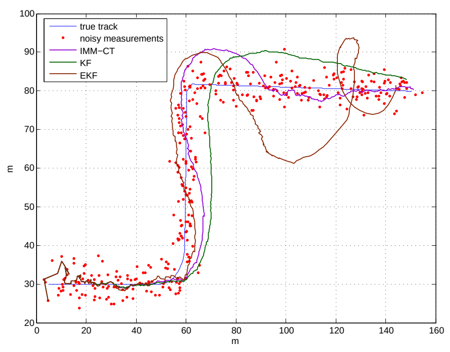
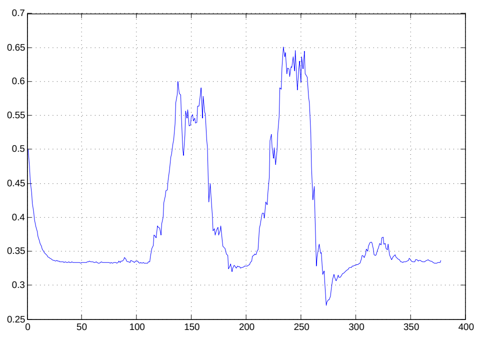


Figure 3.5: Maneuvering mode probability for the EKF. Configuration 1

Next the IMM is evaluated with a measurement noise higher, σ_z from $1.5\,m$ to $3\,m$. The results in 3.6(a) show that the EKF performance is bad after the second turn. In contrast the IMM tracks more accurately the target. For instance, after the position $(120\,m, 80\,m)$ the IMM is tracking the trajectory within a position error less than $4\,m$ approximately whereas the KF seems to track with a similar position error than the IMM after the position $(140\,m, 80\,m)$. Table 3.2 shows the RMSE of all tracking algorithms. Notice that the error of IMM is the smallest one.



(a) Tracking with EKF, KF and IMM:$\sigma_z = 3\,m$



(b) Maneuvering mode probability for EKF

Figure 3.6: Configuration 1. Comparison of KF,EKF and IMM with higher measurements standard deviation in 3.6(a)(b) Maneuvering mode probability for EKF in 3.6(b)

RMSE$_z$	3.7986 m
RMSE$_{\text{KF}}$	6.9962 m
RMSE$_{\text{EKF}}$	7.1958 m
RMSE$_{\text{IMM}}$	3.6734 m

Table 3.2: RMSE errors of the different tracking estimators

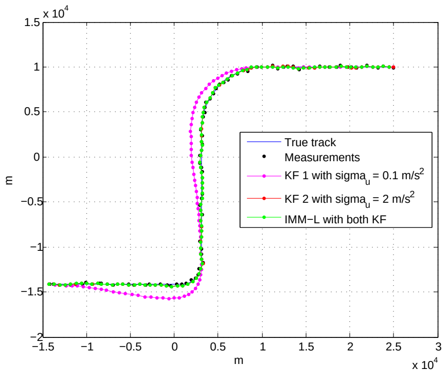
Following, another configuration taken from [12] is used to evaluate the tracking algorithms.

Configuration 2: The scenario area is of dimensions 30000 $m\times$ 15000 m and a constant velocity $v = 120\,m/s$ has been considered (note that this is for a typical ATC tracking application). The interval between samples is $T = 5\,s$. In the scenario under consideration, the initial point at time $t = 0\,s$ where the state vector is initialized is at the position $[25000\,m, 10000\,m]$. The flight goes westward during the first 125 s. Then it performs a $1\,°/s$ coordinated turn to the left during 90 s. After the left turn the flight goes southward during 125 s. Then another $3\,°/s$ coordinated turn to the right is executed during 30 s and finally the flights flies westward again with constant velocity. The position measurements have a standard deviation of $\sigma_z = 100\,m$. The first turn corresponds to a radius of 6875.5 m whereas the second turn corresponds to a radius of 3600. This IMM configuration is referred to us as IMM-L since 2 linear state estimators have been used (2 Kalman filters):

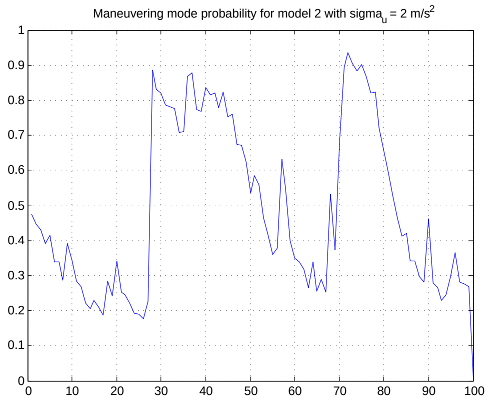
1. One Kalman Filter called KF-1 with a low level noise with a standard deviation of $\sigma_u^2 = 0.1\,m/s^2$ m modelling the uniform motion model.
2. Another Kalman Filter called KF-2 with a higher level process noise (higher acceleration) with a standard deviation of $\sigma_u^2 = 2\,m/s^2$ modelling the turns. Taking into account an certain error to the model will be useful to follow the turns since the motion model is rectilinear.

The tracking of the flight trajectory with one IMM-L and 2 independent KF is shown in figure 3.7(a). Again the maneuvering mode probability for model 2 associated to KF-2 can be seen in 3.7(b).

A comparison of the performance between all tracking algorithms in terms of average position error is given in table 3.3. One can see that the performance of IMM is the best in comparison to the two independent Kalman filters. However multilateration performs better than IMM. Furthermore KF-2 provides more accurate estimations than KF-1 since a higher process noise is given to the model 2 to track the turns.



(a) Tracking with EKF, KF and IMM:$\sigma_z = 100\,m$



(b) Maneuvering mode probability for EKF

Figure 3.7: Configuration 2. (3.7(a)): Comparison of IMM-L with two single KF's: one $\sigma_u^2 = 0.1\,m/s^2$ and another with $\sigma_u^2 = 2\,m/s^2$, (3.7(b)): Maneuvering mode probability for the model 2

RMSE_z	10.1864 m
RMSE_{KF1}	667.6266 m
RMSE_{KF2}	93.0959 m
RMSE_{IMM}	84.3506 m

Table 3.3: RMSE errors of the different tracking estimators used in the Configuration 2

The next chapter deals with the theoretical validation of the set of algorithms that are explained in this chapter. For that a a simulator called "Car Positioning Simulator" is developed in Matlab.

Chapter 4

Matlab Tracking Simulator

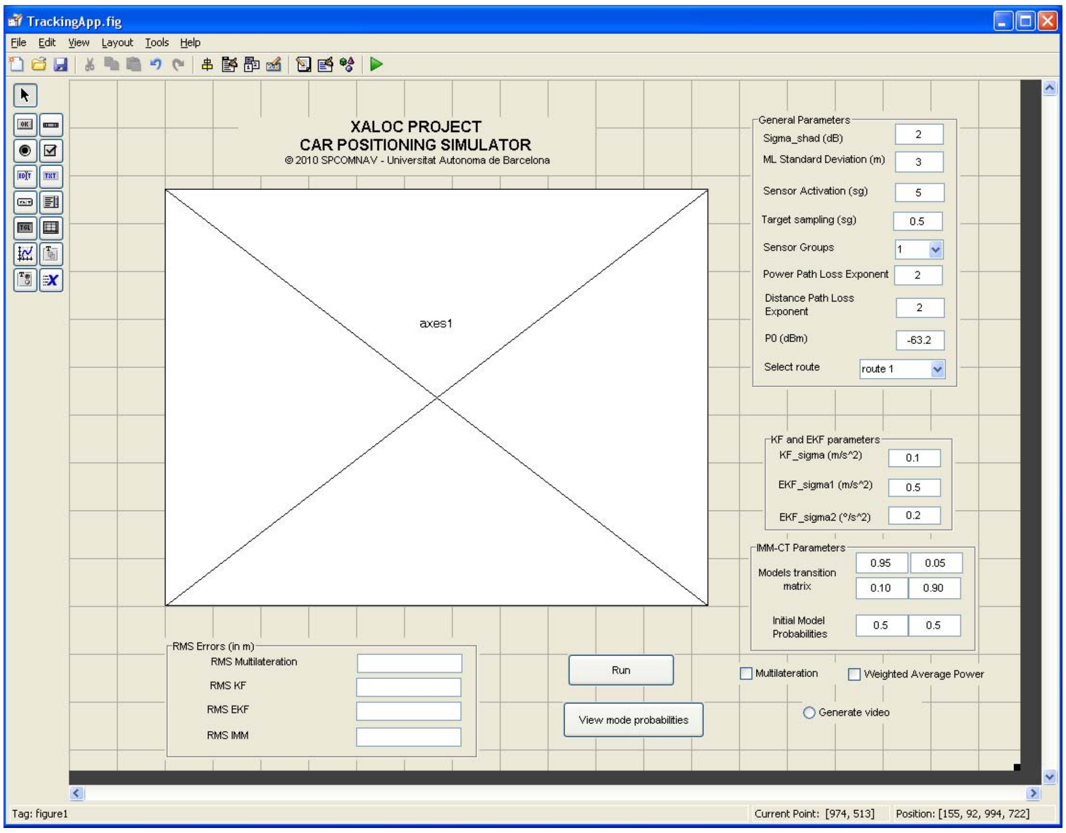
In this chapter we present the simulator we have developed to validate the tracking algorithms. First it is shown how multilateration is able to estimate the position of the target with a certain error around 2 or 3 m. Once the positioning technique is implemented, several tracking algorithms are also developed in the simulator. This simulator, called **"Car Positioning Simulator"** allows to provide a theoretical validation of the tracking techniques in a real scenario: 9 building blocks (3 vertical streets and 2 horizontal streets), several random target routes previously created and the application of KF, EKF and IMM over each of the routes. Furthermore the simulator is designed with a GUI(*Graphical User Interface*) as a centralized application that allows to perform several simulations for different input control parameters. It will be seen later that these control parameters are: the s.t.d(standard deviation) of the measurements, the s.t.d of the turn rate, the transition probability matrix, the initial model probabilities, the path loss exponent, the target sampling and the variance of the shadowing.

This chapter deals with the following sections:

- Introducing the created Matlab GUI-based street simulator as centralized application and give an explanation of this simulator.
- Showing several results for different routes and scenarios of the set of tracking estimators analysed in the previous chapter. For example the performance of all the tracking algorithms is analysed for different scenarios such as low-high shadowing noise, low-high measurement errors The performance of the tracking estimators depends on their parameters $(\mathbf{R}, \mathbf{P}, \mu, \sigma_a)$ which can be adjusted in the GUI application for each scenario.

4.1 Introduction to the Matlab GUI Simulator

Matlab© provides a useful utility to create graphical user interfaces. Graphical interfaces provides facility and comfort to interact with the global application allowing the possibility to carry on several simulations for different values of the input control parameters. The Matlab GUI design application used to built the simulator is shown in Figure 4.1.


TrackingApp.fig
File Edit View Layout Tools Help
XALOC PROJECT
CAR POSITIONING SIMULATOR
© 2010 SPCOMNAV - Universitat Autonoma de Barcelona
axes1
Run
View mode probabilities
Tag: figure1

General Parameters

Sigma_shad (dB)	2
ML Standard Deviation (m)	3
Sensor Activation (sg)	5
Target sampling (sg)	0.5
Sensor Groups	1
Power Path Loss Exponent	2
Distance Path Loss Exponent	2
P0 (dBm)	-63.2
Select route	route 1

KF and EKF parameters

KF_sigma (m/s^2)	0.1
EKF_sigma1 (m/s^2)	0.5
EKF_sigma2 (°/s^2)	0.2

IMM-CT Parameters

Models transition matrix	0.95	0.05
	0.10	0.90
Initial Model Probabilities	0.5	0.5

RMS Errors (in m)

RMS Multilateration	
RMS KF	
RMS EKF	
RMS IMM	

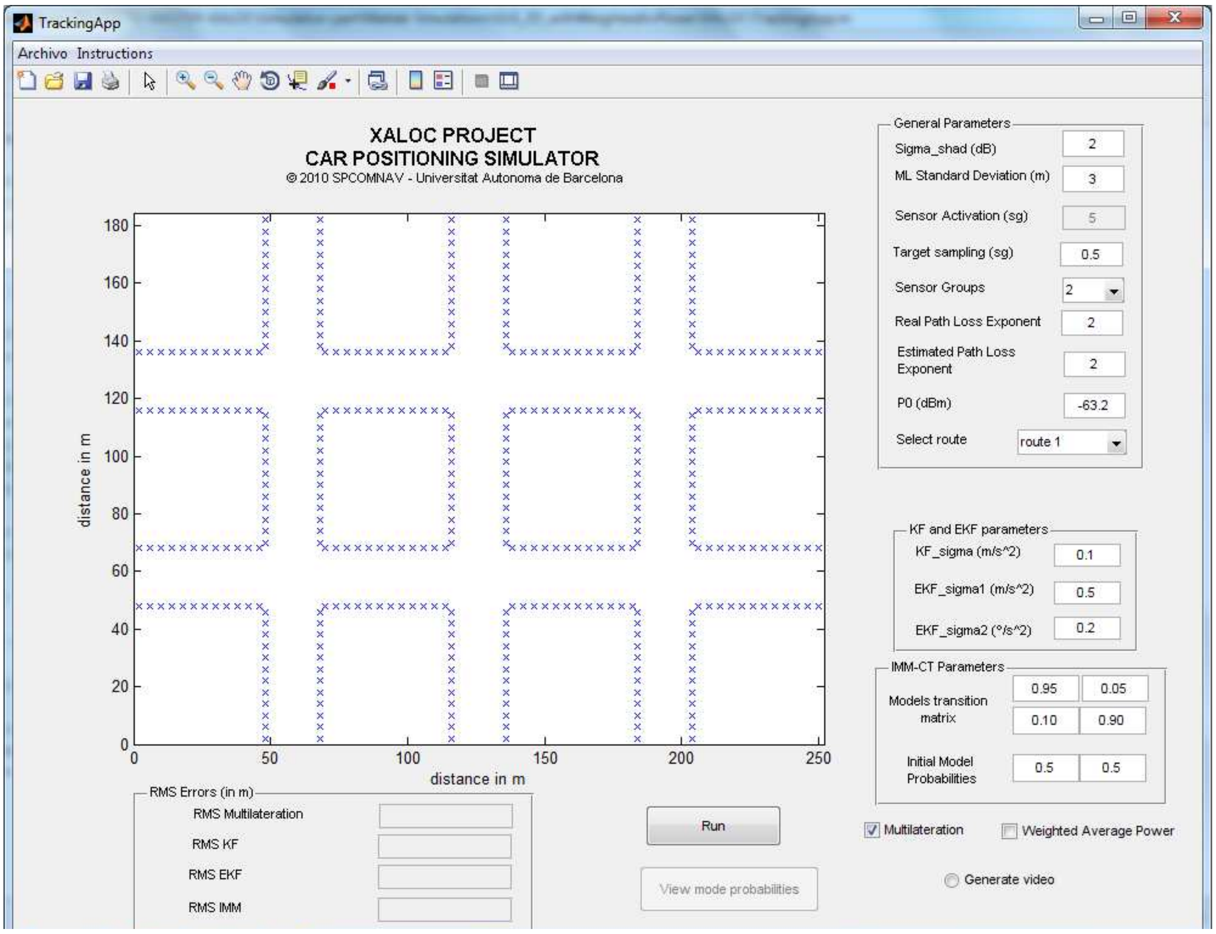
☐ Multilateration ☐ Weighted Average Power
○ Generate video

Current Point: [974, 513] Position: [155, 92, 994, 722]

Figure 4.1: "Matlab GUI Design Application" based on Matlab©

The Matlab GUI utility is easy to use; it provides several graphic objects such as Textfields, Buttons, radiobuttons and checkboxes. The final design of the application should be practical and understood by the user. On the right side of the figure, one can see the input control parameters of the application, whereas the output will be represented in the graphical axes and the RMSE errors will be shown in the bottom of the figure.

The scenario is a set of streets with dimensions $20 \times 48\, m^2$. The distance between the anchor nodes or the sensor nodes located in each of the parkings slots is set to $4\, m$. The anchor nodes are represented by the symbol $\times$ in the simulations. The number of anchor nodes are 12 per street side. The built application of the simulator is shown in figure 4.2.



General Parameters

Sigma_shad (dB)	2
ML Standard Deviation (m)	3
Sensor Activation (sg)	5
Target sampling (sg)	0.5
Sensor Groups	2
Real Path Loss Exponent	2
Estimated Path Loss Exponent	2
P0 (dBm)	-63.2
Select route	route 1

KF and EKF parameters

KF_sigma (m/s^2)	0.1
EKF_sigma1 (m/s^2)	0.5
EKF_sigma2 (°/s^2)	0.2

IMM-CT Parameters

Models transition matrix	0.95	0.05
	0.10	0.90
Initial Model Probabilities	0.5	0.5

RMS Errors (in m)

RMS Multilateration	
RMS KF	
RMS EKF	
RMS IMM	

[✓] Multilateration
[ ] Weighted Average Power
○ Generate video

Figure 4.2: "Car Positioning Simulator" based on Matlab$^{©}$

A description of the input parameters of the simulator is the following:

<u>**General Parameters**</u>:

- **Sigma_shad**(dB): this parameter corresponds to the variance of the shadow fading, σ^2_{shad}, affecting the RSSI.
- **ML standard deviation(*m*)**: this parameter corresponds to σ_z.
- **Sensor Activation**(sg): it corresponds to the sensor activation time T_a in seconds. By default it is set to $5\,s$ since initially it was a requirement.
- **Target sampling**(sg): it is the time T used to represent the vehicle positions.
- **Sensor Groups**: at the beginning of the project it was supposed that the activation time of all parking sensors was $5\,s$. Due to the activation time was considered too high to obtain enough position measurements for the tracking a new strategy was found. This strategy consisted to have two groups of active sensors with a shift in the activation times. Thus

more position measurements could be obtained for the same activation time. For example an activation time of $5\,s$ with one group of active sensors corresponds to one measurement per street at the travel speed (all sensors active at the same time) whereas with two groups of active sensors it doubles the number of measures (two position measurements per street). This is shown in the simulations section.

- **Real Path Loss Exponent**: this parameter is the γ_{RSSI} taken into account for the pathloss model to calculate the received power at a distance d_i from the transmitter, i.e. $RSSI = P_t - 10\gamma_{rssi} log_{10} d_i$ being P_t the transmitted power in dBm.
- **Estimated Path Loss Exponent**: in practice γ_{rssi} is not known but an estimate $\hat{\gamma}_{rssi}$ is found. This parameter is taken into account because in practice the path loss exponent estimated by the user, $\hat{\gamma}_{rssi}$ from the RSSI can be different from the real one, γ_{rssi}.
- **P0(dBm)**: it is the received power measure at $1\,m$ from the transmitter. By default a measurement of the RSSI at a distance of $1\,m$ from the transmitter at the frequency $2.4\,GHz$ is $-63.2\,dBm$. Although this value seems to be very small at this distance it has sense because the sensor motes are not powerful and cannot transmit high power levels. It is worth recalling that the sensor motes are energy constrained.
- **Select route**: routes are previously computed allowing then to be loaded on the simulator.

<u>**KF and EKF Parameters**</u>:

- $\mathbf{KF_{sigma}}$ (m/s^2): this parameter corresponds to σ_u^2 of the KF.
- $\mathbf{EKF_{sigma1}}$ (m/s^2): this parameter corresponds to σ_u^2 of the EKF in the both dimensions x and y corresponding to the elements $(1,1)$ and $(2,2)$ of the acceleration noise covariance matrix $\mathbf{U}$.
- $\mathbf{EKF_{sigma2}}$$(^\circ/s^2)$: it is the variance of the turn rate Ω, that is, the element $(3,3)$ of $\mathbf{U}$.

<u>**IMM-CT Parameters**</u>:

- **Models transition matrix**: this matrix corresponds to π_{CT}.
- **Initial Modal Probabilities**: initial $\mu_{\mathbf{0}}$.

- <u>**Multilateration**</u>: if this option is set then multilateration is used to estimate the target position.
- <u>**Weighted Average Power**</u>: this technique is used in the practical scenario shown in chapter 5. It is another strategy to estimate the position of the target from the known

positions of the anchors and a set of weighting coefficients based on the received power from the selected anchor nodes. The mathematical model of this technique is shown in section 5.2. As shown in the following chapter this strategy offers better results in a real scenario where the estimate positions can be corrupted by high shadowing noise. In particular the position is obtained as follows:

$$\begin{gathered}(\hat{x}, \hat{y}) = \mathbf{A} \cdot \alpha \\ \alpha = \frac{\mathrm{RSSI}_n}{\sum_{i=1}^{N} \mathrm{RSSI}_i} \quad n = 1, \cdots N\end{gathered} \tag{4.1}$$

being $\mathbf{A}_{2\times N}$ a matrix containing the (x, y) coordinates of N selected anchors with highest RSSI. The α is a vector of dimensions $N \times 1$ containing the weighting coefficients; every anchor node has associated a certain weight which is dependent on the ratio between the RSSI from a specific anchor and the sum of all the RSSIs from all the anchors (it is a normalization of the received power).

- **Generate video**: this option gives to the user the possibility to generate and store locally a Quicktime movie of the performed simulation.
- **Run**: this button starts the simulator
- **View mode probabilities**: this button allows the user to see the mode probabilities computed by the IMM estimator.

A useful output parameter is the RMS between the actual target position and the estimated one. It allows to compare numerically the performance of all the estimators in terms of average position error.

Next we show a block diagram of the simulator.

4.1.1 Simulator block diagram

The block diagram of the simulator is shown in figure 4.3. It can be seen the set of matlab scripts and the wirings between them.

The set of scripts are described as follows:

- TrackingApp.m: This is the Matlab GUI tracking application.
- TrackingIMMKFandEKF.m: This code implements the simulator, that is, when user click over the button run that appears in the application. This function calls all the tracking estimators. The inputs of this script are all the input parameters of the application.

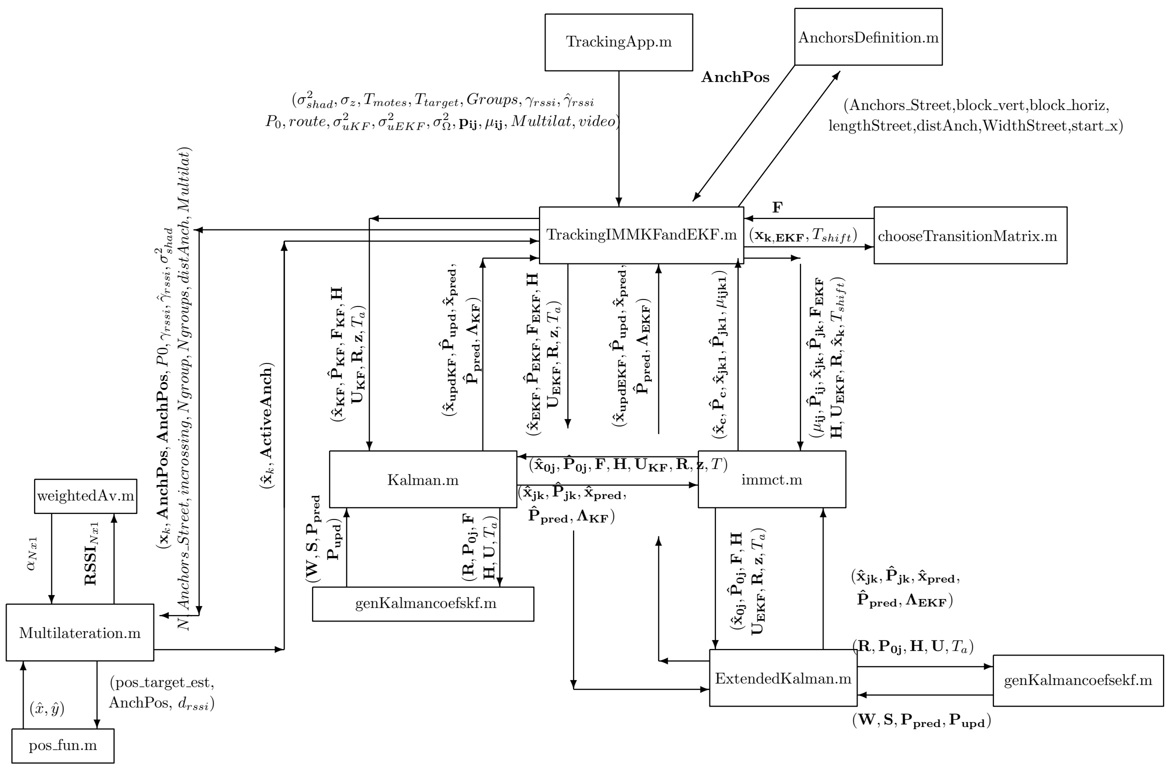

TrackingApp.m
AnchorsDefinition.m
AnchPos
$(\sigma^2_{shad}, \sigma_z, T_{motes}, T_{target}, Groups, \gamma_{rssi}, \hat{\gamma}_{rssi}$
$P_0, route, \sigma^2_{uKF}, \sigma^2_{uEKF}, \sigma^2_{\Omega}, \mathbf{p_{ij}}, \mu_{ij}, Multilat, video)$
(Anchors_Street,block_vert,block_horiz,
lengthStreet,distAnch,WidthStreet,start_x)
TrackingIMMKFandEKF.m
F
$(\mathbf{x_{k,EKF}}, T_{shift})$
chooseTransitionMatrix.m
$(\mathbf{x}_k, \mathbf{AnchPos}, \mathbf{AnchPos}, P0, \gamma_{rssi}, \hat{\gamma}_{rssi}, \sigma^2_{shad}$
$N, Anchors\_Street, incrossing, Ngroup, Ngroups, distAnch, Multilat)$
$(\hat{\mathbf{x}}_k, \mathbf{ActiveAnch})$
$(\hat{\mathbf{x}}_{\mathbf{KF}}, \hat{\mathbf{P}}_{\mathbf{KF}}, \mathbf{F}_{\mathbf{KF}}, \mathbf{H}$
$\mathbf{U_{KF}}, \mathbf{R}, \mathbf{z}, T_a)$
$(\hat{\mathbf{x}}_{\mathbf{updKF}}, \hat{\mathbf{P}}_{\mathbf{upd}}, \hat{\mathbf{x}}_{\mathbf{pred}},$
$\hat{\mathbf{P}}_{\mathbf{pred}}, \mathbf{\Lambda_{KF}})$
$(\hat{\mathbf{x}}_{\mathbf{EKF}}, \hat{\mathbf{P}}_{\mathbf{EKF}}, \mathbf{F}_{\mathbf{EKF}}, \mathbf{H}$
$\mathbf{U_{EKF}}, \mathbf{R}, \mathbf{z}, T_a)$
$(\hat{\mathbf{x}}_{\mathbf{updEKF}}, \hat{\mathbf{P}}_{\mathbf{upd}}, \hat{\mathbf{x}}_{\mathbf{pred}},$
$\hat{\mathbf{P}}_{\mathbf{pred}}, \mathbf{\Lambda_{EKF}})$
$(\hat{\mathbf{x}}_{\mathbf{c}}, \hat{\mathbf{P}}_{\mathbf{c}}, \hat{\mathbf{x}}_{\mathbf{jk1}}, \hat{\mathbf{P}}_{\mathbf{jk1}}, \mu_{\mathbf{ijk1}})$
$(\mu_{\mathbf{ij}}, \hat{\mathbf{P}}_{\mathbf{ij}}, \hat{\mathbf{x}}_{\mathbf{jk}}, \hat{\mathbf{P}}_{\mathbf{jk}}, \mathbf{F}_{\mathbf{EKF}}$
$\mathbf{H}, \mathbf{U_{EKF}}, \mathbf{R}, \hat{\mathbf{x}}_{\mathbf{k}}, T_{shift})$
weightedAv.m
Kalman.m
$(\hat{\mathbf{x}}_{\mathbf{0j}}, \hat{\mathbf{P}}_{\mathbf{0j}}, \mathbf{F}, \mathbf{H}, \mathbf{U_{KF}}, \mathbf{R}, \mathbf{z}, T)$
$(\hat{\mathbf{x}}_{\mathbf{jk}}, \hat{\mathbf{P}}_{\mathbf{jk}}, \hat{\mathbf{x}}_{\mathbf{pred}},$
$\hat{\mathbf{P}}_{\mathbf{pred}}, \mathbf{\Lambda_{KF}})$
immct.m
α_{Nx1}
$\mathbf{RSSI}_{Nx1}$
$(\mathbf{W}, \mathbf{S}, \mathbf{P_{pred}}$
$\mathbf{P_{upd}})$
$(\mathbf{R}, \mathbf{P_{0j}}, \mathbf{F}$
$\mathbf{H}, \mathbf{U}, T_a)$
genKalmancoefskf.m
$(\hat{\mathbf{x}}_{\mathbf{0j}}, \hat{\mathbf{P}}_{\mathbf{0j}}, \mathbf{F}, \mathbf{H}$
$\mathbf{U_{EKF}}, \mathbf{R}, \mathbf{z}, T_a)$
$(\hat{\mathbf{x}}_{\mathbf{jk}}, \hat{\mathbf{P}}_{\mathbf{jk}}, \hat{\mathbf{x}}_{\mathbf{pred}},$
$\hat{\mathbf{P}}_{\mathbf{pred}}, \mathbf{\Lambda_{EKF}})$
Multilateration.m
$(\mathbf{R}, \mathbf{P_{0j}}, \mathbf{H}, \mathbf{U}, T_a)$
ExtendedKalman.m
genKalmancoefsekf.m
$(\mathbf{W}, \mathbf{S}, \mathbf{P_{pred}}, \mathbf{P_{upd}})$
$(\hat{x}, \hat{y})$
(pos_target_est,
AnchPos, d_{rssi})
pos_fun.m

Figure 4.3: Block diagram of the "Car Positioning Simulator".

- AnchorsDefinition.m: this script builds creates a matrix containing the 2-D coordinates of all the anchors of the scenario taking into account the following input parameters:
  - Anchors_Street: The total number of Anchors per street. The number of anchors per street is set to 24, corresponding to 24 parking slots (12 per street side).
  - block_vert, block_horiz: the number of building blocks in the vertical and horizontal domain. In the scenario block_vert=3, and block_horiz=4.
  - lengthStreet: the length of the street is set to 48 m.
  - distAnch: the distance between anchors is 4 m corresponding to one parking slot.
  - WidthStreet: the width of the streets is set to 20 m.
  - start_x: this is the distance from either $y = 0$ or $x = 0$ where the parking slots starts.
- Multilateration.m: This code performs multilateration with N=4 anchor nodes with maximum RSSI. If the option weighted average is selected in the application, instead of applying multilateration the weighted average method is used. The inputs/outputs of this script are the following:
  - $\mathbf{x}_k$: it is a vector containing the true positions of the target. This vector is used to

compute the real distances to the specified anchors in order to obtain the RSSIs using the path loss model.

  - **AnchPos**: it is a matrix containing the 2-D coordinates of those anchors that belong to the street where the target is located. The second input **AnchPos** are all the anchors and it is used to set which of them are active or not depending on N_{groups}.
  - P0: the RSSI at $1\,m$.
  - γ_{RSSI}: the real path loss exponent.
  - $\hat{\gamma}_{RSSI}$: the estimated path loss exponent.
  - σ^2_{shad}: the variance of the shadowing.
  - N: the number of used anchors to carry on multilateration.
  - Anchors_Street: it is described above.
  - incrossing: it controls if the target is inside or outside the crossings. If the target is inside , multilateration is performed with the active anchors located at the border of the crossing.
  - Ngroup: it controls which group of anchors is active at each time instant. This variable will be always 1 if there is only one group of anchors.
  - Ngroups: the number of existing groups of anchors. It is selected in the application with the parameter **Sensor Groups**.
  - distAnch: the distance between Anchors.
  - Multilat: if the option multilateration is selected in the application then this variable is set to 1.
  - $\hat{\mathbf{x_k}}$: it is an output vector that gives the position estimates of the target.
  - **ActiveAnch**: it is an output matrix that contains the coordinates of the active anchors. It is used to show the user in the axes of the application which anchors are active.

- weightedAverage.m: This code generates the coefficients used in the weighted average method to calculate the target position.

- posfun.m: It corresponds to the cost function $C(x, y)$ appeared in 2.9. This cost function is the euclidean distance from the target to each of the anchors. The function fminunc from matlab optimization toolbox finds the values (x, y) that minimize $C(x, y)$.

- Kalman.m: It implements the Kalman Filter. On the one hand, the inputs from TrackingIMMKFandEKF.m are: the state vector $\hat{\mathbf{x}}_{\mathbf{KF}}$ and the state covariance matrix $\hat{\mathbf{P}}_{\mathbf{KF}}$ at the previous time instant, the transition matrix $\mathbf{F}$, the observation matrix $\mathbf{H}$, the acceleration noise covariance matrix of the process $\mathbf{U_{KF}}$, the measurement noise covariance matrix $\mathbf{R}$, the position measurements $\mathbf{z}$, and the sensor activation time T_a. The outputs are: the updated state vector $\hat{\mathbf{x}}_{\mathbf{updKF}}$ and state covariance matrix $\hat{\mathbf{P}}_{\mathbf{upd}}$, the predicted state vector $\hat{\mathbf{x}}_{\mathbf{pred}}$ and state covariance matrix $\hat{\mathbf{P}}_{\mathbf{pred}}$, and the likelihood probability $\mathbf{\Lambda_{KF}}$. The last is used in the IMM. On the other hand, the inputs from IMM are the same than the other set of inputs except: the mixed state vector $\hat{\mathbf{x}}_{\mathbf{0j}}$, the mixed state covariance matrix $\hat{\mathbf{P}}_{\mathbf{0j}}$ and the transition matrix $\mathbf{F}$ correspond to the first elements of a 1×2 cell array. A cell array can be used to store matrices of any dimension in each slot. The outputs can be interpreted in the same way than in the other set of outputs.

- genKalmancoefskf.m: This script computes the Kalman coefficients from the state covariance matrix $\mathbf{P_{0j}}$, measurement noise covariance matrix,the transition matrix, the observation matrix, acceleration noise covariance and the sensor activation time.

- ExtendedKalman.m: It implements the Extended Kalman Filter. The set of inputs and outputs can be interpreted in the same way as in the Kalman Filter. Regarding the inputs from IMM, the mixed state state vector, the mixed covariance matrix and the transition matrix correspond to the second elements of the cell array.

- genKalmancoefsekf.m: This script computes the Extended Kalman coefficients in the same way as in Kalman Filter except that the transition matrix is not passed as an input parameter. In fact the Jacobian of the nonlinear function of the dynamic state is updated online at each time step since it depends on the actual target turn rate Ω of the state vector.

- chooseTransitionMatrix.m: This function chooses the transition matrix depending on Ω. If $\Omega < threshold$ the used transition matrix corresponds to the uniform motion model whereas if $\Omega > threshold$ the used transition matrix corresponds to the coordinated turn model. This threshold is set to 10^{-12}.

- immct.m: This code implements the IMM-CT tracking estimator. It requires the functions Kalman.m, ExtendedKalman.m and chooseTransitionMatrix.m. The inputs of this script are described next:

  – $\mu_{\mathbf{ij}}$: this vector corresponds to the prior model probabilities.
  – $\hat{\mathbf{P}}_{\mathbf{ij}}$: it corresponds to the models transition matrix $\pi_{\mathbf{CT}}$.
  – $\hat{\mathbf{x}}_{\mathbf{jk}}, \hat{\mathbf{P}}_{\mathbf{jk}}$: These parameters are a 1×2 cell array containing the outputs of both KF and EKF, $\hat{\mathbf{x}}_{\mathbf{jk}}, \hat{\mathbf{P}}_{\mathbf{jk}}$, at the previous time step. It is feedback.
  – $\mathbf{F}, \mathbf{H}, \mathbf{U}, \mathbf{R}$: these parameters are described above.
  – $\hat{\mathbf{x}}_{\mathbf{k}}$: the position estimates obtained with the script Multilateration.m.
  – T_{shift}: it is the sensor activation time.

The outputs are:

- $\hat{\mathbf{x}}_\mathbf{c}, \hat{\mathbf{P}}_\mathbf{c}$: a 1×2 cell array containing the updated combined state vector and state covariance from both filters.
- $\hat{\mathbf{x}}_\mathbf{jk1}, \hat{\mathbf{P}}_\mathbf{jk1}$: a 1×2 cell array containing the updated state and covariance computed by each of the filters.
- $\mu_\mathbf{jk1}$: the updated model probabilities at each time step.

Next we show the set of target routes used in the simulations.

4.1.2 Target Routes

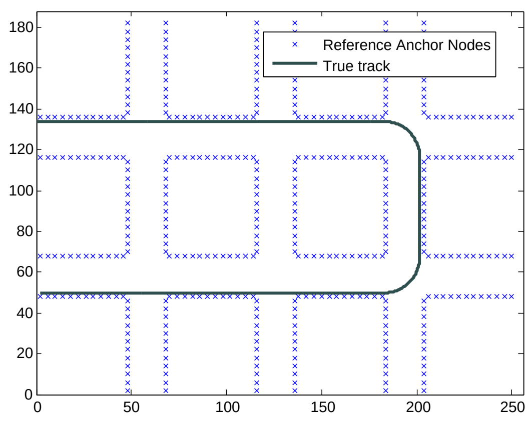
The target routes allows to load one of the seven target roads stored locally. For instance, three target routes are shown in figures 4.4(a) 4.4(b) and 4.4(c). All routes have both left and turn rights. The following section use these true target routes to apply positioning techniques and the tracking estimators.

4.2 Tracking and Positioning Simulation

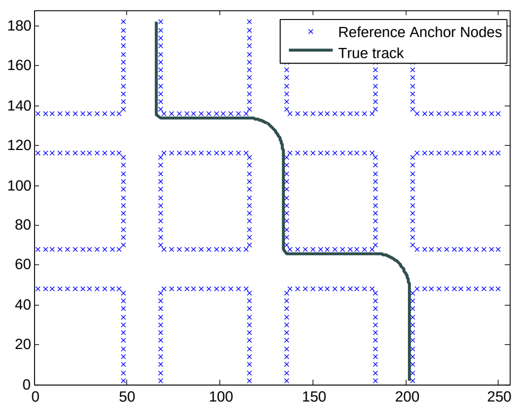
This section is devoted to test the estimators described in the previous chapter for different scenarios. The main idea is to show how the performance of the positioning and tracking techniques vary by modifying the input parameters of the simulator. First, the three target routes are tracked by using the following default values for the input parameters:

General Parameters:

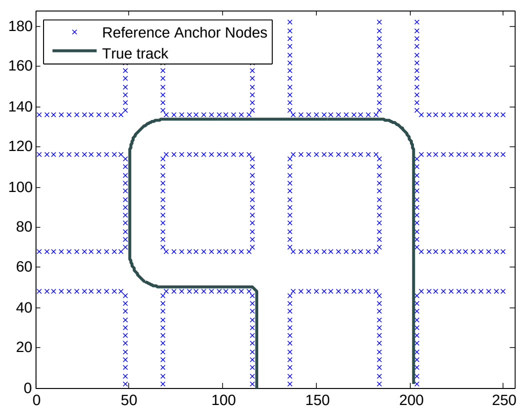
- **Sigma_shad**(dB): $2\,dB$
- **ML standard deviation(dB)**: $3\,m$
- **Sensor Activation (s)**: $5\,s$
- **Target sampling (s)**: $0.5\,s$
- **Sensor Groups**: 1
- **Power Path Loss Exponent**: 2
- **Distance Path Loss Exponent**: 2
- **P0(dBm)**: -63.2



(a) Route 1



(b) Route 2



(c) Route 5

Figure 4.4: Target routes definition.

KF, EKF and IMM Parameters:

- $\mathbf{KF_{sigma}}$ (m/s^2): 0.1
- $\mathbf{EKF_{sigma1}}$ (m/s^2): 0.5
- $\mathbf{EKF_{sigma2}}$$(^\circ/s^2)$: 0.2
- **Models transition matrix:**

$$\pi_{\mathbf{CT}} = \begin{bmatrix} 0.95 & 0.05 \\ 0.10 & 0.90 \end{bmatrix} \tag{4.2}$$

- **Initial Modal Probabilities:**

$$\mu_{\mathbf{0}} = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix} \tag{4.3}$$

Multilateration: on

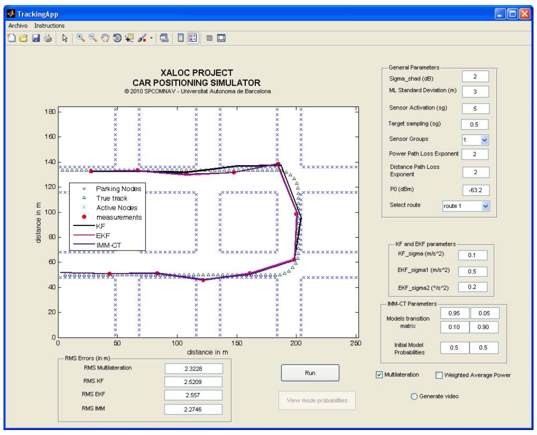
<u>**Case 1. Modifying Ngroups**</u> The results for the defined default input parameters are shown respectively in figures 4.5(a), 4.5(b) and 4.5(c) for the three target routes. The corresponding RMSE errors of all the estimators for the different routes are shown in table 4.1. On the one hand it can be seen that the number of total measurements is not enough to have efficient tracking since the estimators converge with a high number of measurements. It is due to the high sensor activation time for a target speed of $30\,Km/h = 8.33\,m/s$ meaning approximately one position measure per street every $5\,s$ (the length of one street is $48\,m$). The sensor activation time comes from a requirement of the company. In order to have more position measurements, two groups of sensors were considered with a delay T_{shift} in the activation time. For example two groups of active sensors imply that $T_{shift} = T_a/Ngroups$. For the case of $Ngroups = 2$ then $T_{shift} = 2.5\,s$ meaning that the computation of multilateration is done every $2.5\,s$. As a result the double of the position measurements with one group is obtained as shown in figures 4.6(a), 4.6(b) and 4.6(c) with the corresponding RMSE errors shown in table 4.2.

On the other hand it can be seen that the performance of IMM is the best one in comparison to KF and EKF, since it provides the smallest RMSE for any route giving an average difference less than $70\,cm$ with respect to the multilateration. Moreover the EKF performance is better than the KF performance in most of the situations when both values RMSE_{EKF} and RMSE_{KF} are compared. It is because EKF behaves as a KF when the turn rate is zero as demonstrated by (3.14).

The number of output estimates from all the tracking estimators is the same than the number of multilateration position estimates. In order to show the estimated trajectory of the target, all the points are connected with lines (it would be better to have a lot of continuous measurements but in practice it is impossible due to sensors energy constraints).

	Route 1	Route 2	Route 3
$\text{RMSE}_z(m)$	2.3228	2.0484	1.9477
$\text{RMSE}_{\text{KF}}(m)$	2.5209	3.7951	2.9892
$\text{RMSE}_{\text{EKF}}(m)$	2.557	2.8116	2.3482
$\text{RMSE}_{\text{IMM}}(m)$	2.2746	2.7016	2.2618

Table 4.1: RMSE errors associated to the results in figures 4.5(a), 4.5(b) and 4.5(c).



General Parameters

Sigma_shad (dB)	2
ML Standard Deviation (m)	3
Sensor Activation (sg)	5
Target sampling (sg)	0.5
Sensor Groups	1
Power Path Loss Exponent	2
Distance Path Loss Exponent	2
P0 (dBm)	-63.2
Select route	route 1

KF and EKF parameters

KF_sigma (m/s^2)	0.1
EKF_sigma1 (m/s^2)	0.5
EKF_sigma2 (°/s^2)	0.2

IMM-CT Parameters

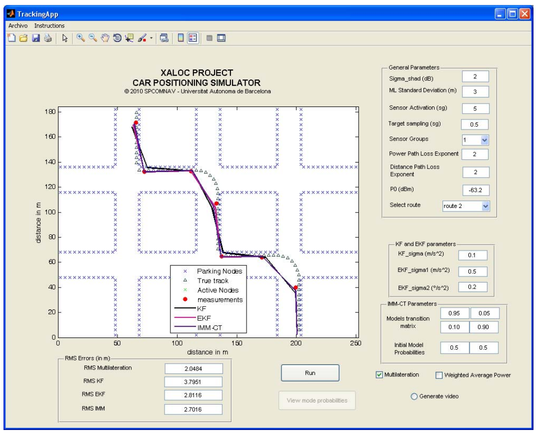
Models transition matrix	0.95	0.05
	0.10	0.90
Initial Model Probabilities	0.5	0.5

[✓] Multilateration [ ] Weighted Average Power ○ Generate video

RMS Errors (in m)

RMS Multilateration	2.3228
RMS KF	2.5209
RMS EKF	2.557
RMS IMM	2.2746

(a) Route 1



General Parameters

Sigma_shad (dB)	2
ML Standard Deviation (m)	3
Sensor Activation (sg)	5
Target sampling (sg)	0.5
Sensor Groups	1
Power Path Loss Exponent	2
Distance Path Loss Exponent	2
P0 (dBm)	-63.2
Select route	route 2

KF and EKF parameters

KF_sigma (m/s^2)	0.1
EKF_sigma1 (m/s^2)	0.5
EKF_sigma2 (°/s^2)	0.2

IMM-CT Parameters

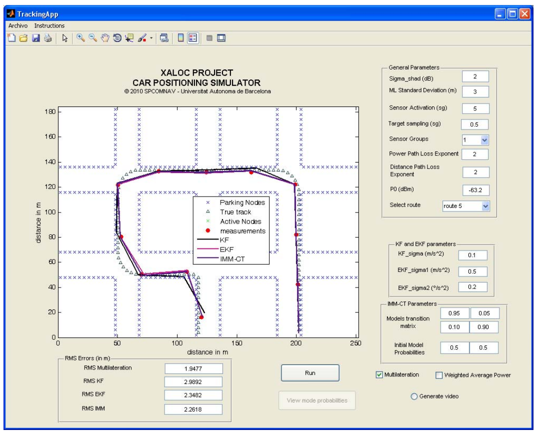
Models transition matrix	0.95	0.05
	0.10	0.90
Initial Model Probabilities	0.5	0.5

[✓] Multilateration [ ] Weighted Average Power ○ Generate video

RMS Errors (in m)

RMS Multilateration	2.0484
RMS KF	3.7951
RMS EKF	2.8116
RMS IMM	2.7016

(b) Route 2



General Parameters

Sigma_shad (dB)	2
ML Standard Deviation (m)	3
Sensor Activation (sg)	5
Target sampling (sg)	0.5
Sensor Groups	1
Power Path Loss Exponent	2
Distance Path Loss Exponent	2
P0 (dBm)	-63.2
Select route	route 5

KF and EKF parameters

KF_sigma (m/s^2)	0.1
EKF_sigma1 (m/s^2)	0.5
EKF_sigma2 (°/s^2)	0.2

IMM-CT Parameters

Models transition matrix	0.95	0.05
	0.10	0.90
Initial Model Probabilities	0.5	0.5

[✓] Multilateration [ ] Weighted Average Power ○ Generate video

RMS Errors (in m)

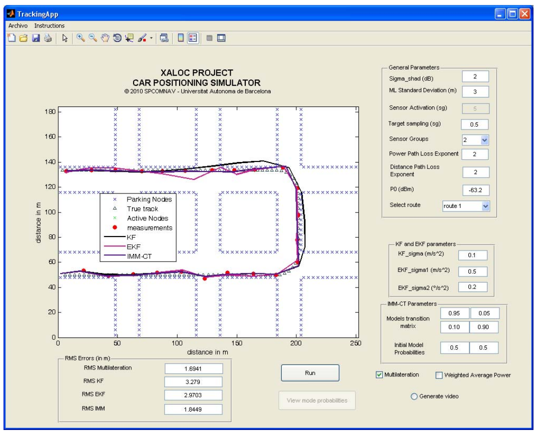
RMS Multilateration	1.9477
RMS KF	2.9892
RMS EKF	2.3482
RMS IMM	2.2618

(c) Route 5

Figure 4.5: Simulation of tracking algorithms for different target routes and the default parameters.

	Route 1	Route 2	Route 3
$\mathrm{RMSE}_z(m)$	1.6941	2.0311	1.8244
$\mathrm{RMSE}_{\mathrm{KF}}(m)$	3.279	4.5699	4.7661
$\mathrm{RMSE}_{\mathrm{EKF}}(m)$	2.9703	4.3902	2.9681
$\mathrm{RMSE}_{\mathrm{IMM}}(m)$	1.8449	2.4737	2.1897

Table 4.2: RMSE errors associated to the results in figures 4.6(a), 4.6(b) and 4.6(c).



General Parameters
Sigma_shad (dB) 2
ML Standard Deviation (m) 3
Sensor Activation (sg) 5
Target sampling (sg) 0.5
Sensor Groups 2
Power Path Loss Exponent 2
Distance Path Loss Exponent 2
P0 (dBm) -63.2
Select route: route 1

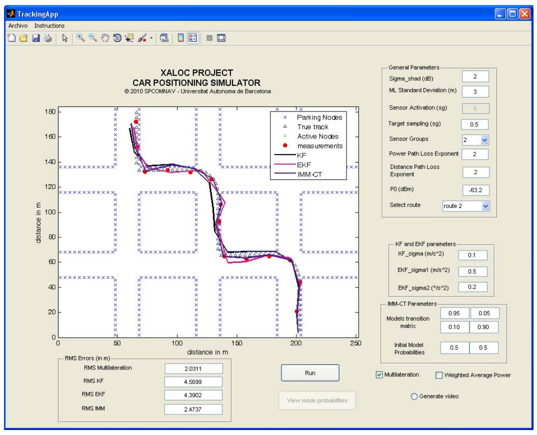
KF and EKF parameters
KF_sigma (m/s^2) 0.1
EKF_sigma1 (m/s^2) 0.5
EKF_sigma2 (°/s^2) 0.2

IMM-CT Parameters
Models transition matrix: 0.95 0.05 / 0.10 0.90
Initial Model Probabilities: 0.5 0.5

[✓] Multilateration [ ] Weighted Average Power ○ Generate video

RMS Errors (in m)
RMS Multilateration 1.6941
RMS KF 3.279
RMS EKF 2.9703
RMS IMM 1.8449

(a) Route 1



General Parameters
Sigma_shad (dB) 2
ML Standard Deviation (m) 3
Sensor Activation (sg) 5
Target sampling (sg) 0.5
Sensor Groups 2
Power Path Loss Exponent 2
Distance Path Loss Exponent 2
P0 (dBm) -63.2
Select route: route 2

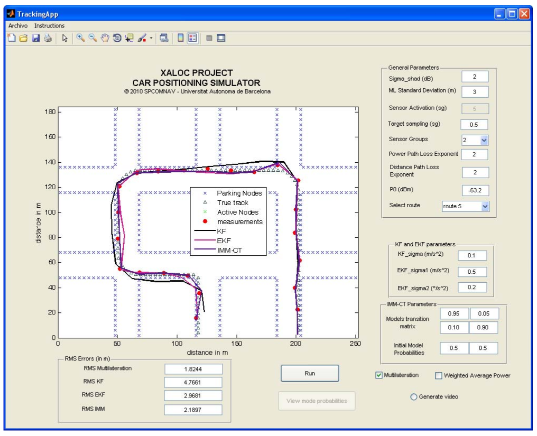
KF and EKF parameters
KF_sigma (m/s^2) 0.1
EKF_sigma1 (m/s^2) 0.5
EKF_sigma2 (°/s^2) 0.2

IMM-CT Parameters
Models transition matrix: 0.95 0.05 / 0.10 0.90
Initial Model Probabilities: 0.5 0.5

[✓] Multilateration [ ] Weighted Average Power ○ Generate video

RMS Errors (in m)
RMS Multilateration 2.0311
RMS KF 4.5699
RMS EKF 4.3902
RMS IMM 2.4737

(b) Route 2



General Parameters
Sigma_shad (dB) 2
ML Standard Deviation (m) 3
Sensor Activation (sg) 5
Target sampling (sg) 0.5
Sensor Groups 2
Power Path Loss Exponent 2
Distance Path Loss Exponent 2
P0 (dBm) -63.2
Select route: route 5

KF and EKF parameters
KF_sigma (m/s^2) 0.1
EKF_sigma1 (m/s^2) 0.5
EKF_sigma2 (°/s^2) 0.2

IMM-CT Parameters
Models transition matrix: 0.95 0.05 / 0.10 0.90
Initial Model Probabilities: 0.5 0.5

[✓] Multilateration [ ] Weighted Average Power ○ Generate video

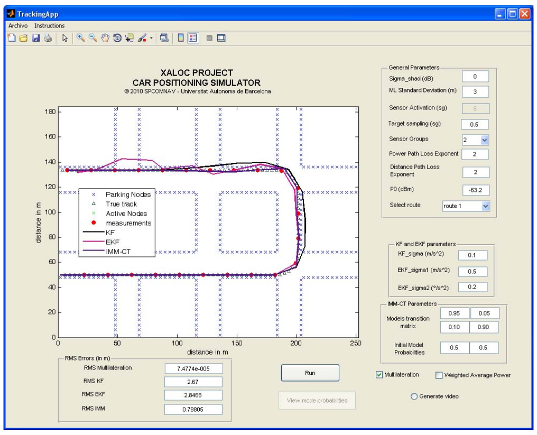
RMS Errors (in m)
RMS Multilateration 1.8244
RMS KF 4.7661
RMS EKF 2.9681
RMS IMM 2.1897

(c) Route 5

Figure 4.6: Tracking simulation with Ngroups= 2 and $T_a = 5\,s$.

Therefore from now on the simulations will be done with 2 groups of sensors, lay adopting an activation time of $T_{motes} = 5\,s$ and an activation shift time of $T_{shift} = 2.5\,s$. The target sampling will be set to $T_{target} = 0.5\,s$.

Case 2. Impact of σ^2_{shad} variation: How the parameter σ^2_{shad} have an influence on the position estimates?. This parameter is related to the received power and it accounts for the shadowing in wireless channels. The shadowing effect is modelled as a lognormal with mean 0 and variance σ^2_{shad}. For example if $\sigma^2_{shad} = 0$ means that the received power is only dependent on the distance and as a result the computed position measurements with multilateration are exactly the true target position as shown in figure 4.7. However the tracking estimators still gives bad estimations, due to the dependency on the std of the position measurements. It will be seen in figures 4.9(a), 4.9(b) and 4.9(c) that when considering $\sigma_z = 0$ then all tracking estimators estimate the same true track and therefore the RMSE errors shown in table 4.5 will be zero. It is worth recalling that the target position estimates are found with those four anchor nodes with higher received power. The results where $\sigma^2_{shad} = 0$ are shown in figure 4.7 with the corresponding RMSE errors in table 4.3. One can see in table 4.3 that RMSE$_z(m)$ is almost negligible (not zero because it comes from estimates) for all routes since the shadowing noise is set to a zero value.



General Parameters
Sigma_shad (dB): 0
ML Standard Deviation (m): 3
Sensor Activation (sg): 5
Target sampling (sg): 0.5
Sensor Groups: 2
Power Path Loss Exponent: 2
Distance Path Loss Exponent: 2
P0 (dBm): -63.2
Select route: route 1

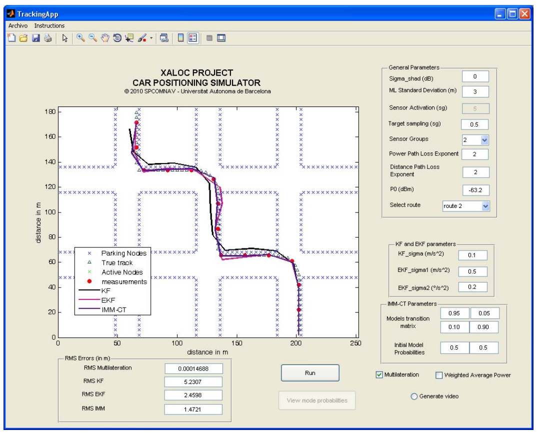
KF and EKF parameters
KF_sigma (m/s^2): 0.1
EKF_sigma1 (m/s^2): 0.5
EKF_sigma2 (°/s^2): 0.2

IMM-CT Parameters
Models transition matrix: 0.95 0.05 / 0.10 0.90
Initial Model Probabilities: 0.5 0.5

[✓] Multilateration [ ] Weighted Average Power

RMS Errors (in m)
RMS Multilateration: 7.4774e-005
RMS KF: 2.67
RMS EKF: 2.8468
RMS IMM: 0.78805

(a) Route 1



General Parameters
Sigma_shad (dB): 0
ML Standard Deviation (m): 3
Sensor Activation (sg): 5
Target sampling (sg): 0.5
Sensor Groups: 2
Power Path Loss Exponent: 2
Distance Path Loss Exponent: 2
P0 (dBm): -63.2
Select route: route 2

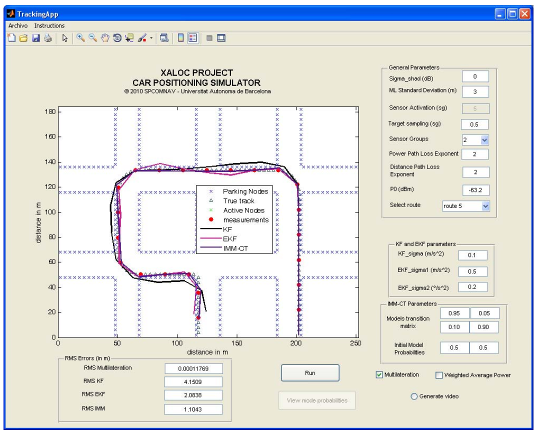
KF and EKF parameters
KF_sigma (m/s^2): 0.1
EKF_sigma1 (m/s^2): 0.5
EKF_sigma2 (°/s^2): 0.2

IMM-CT Parameters
Models transition matrix: 0.95 0.05 / 0.10 0.90
Initial Model Probabilities: 0.5 0.5

[✓] Multilateration [ ] Weighted Average Power

RMS Errors (in m)
RMS Multilateration: 0.00014688
RMS KF: 5.2307
RMS EKF: 2.4598
RMS IMM: 1.4721

(b) Route 2



General Parameters
Sigma_shad (dB): 0
ML Standard Deviation (m): 3
Sensor Activation (sg): 5
Target sampling (sg): 0.5
Sensor Groups: 2
Power Path Loss Exponent: 2
Distance Path Loss Exponent: 2
P0 (dBm): -63.2
Select route: route 5

KF and EKF parameters
KF_sigma (m/s^2): 0.1
EKF_sigma1 (m/s^2): 0.5
EKF_sigma2 (°/s^2): 0.2

IMM-CT Parameters
Models transition matrix: 0.95 0.05 / 0.10 0.90
Initial Model Probabilities: 0.5 0.5

[✓] Multilateration [ ] Weighted Average Power

RMS Errors (in m)
RMS Multilateration: 0.00011769
RMS KF: 4.1509
RMS EKF: 2.0838
RMS IMM: 1.1043

(c) Route 5

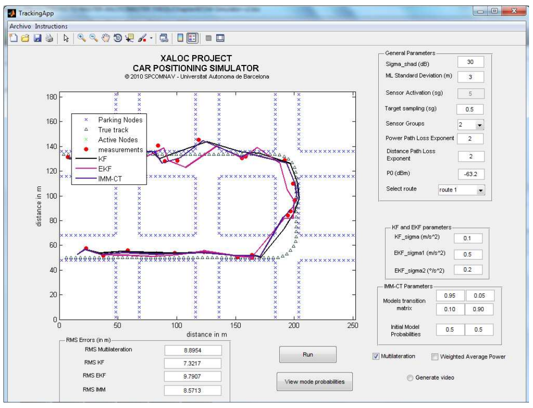
Figure 4.7: Tracking simulation with $\sigma^2_{shad} = 0\,dB$.

Increasing the value of $\sigma^2_{shad} = 0$ to $\sigma^2_{shad} = 30$ dB, for instance, will affect the position measurements. The results are shown in figure 4.8 with the RMSE errors shown in table 4.5. On the one hand one can see that the RMSE errors are greater than in previous simulations. On the other hand the performance of KF is better than IMM and EKF for the case of route 1 and route 3 whereas the IMM performance is the best in route 2. Depending on the scenario

	Route 1	Route 2	Route 3
$\mathrm{RMSE}_z(m)$	7.477e-05	0.00014688	0.00011769
$\mathrm{RMSE}_{\mathrm{KF}}(m)$	2.67	5.2307	4.1509
$\mathrm{RMSE}_{\mathrm{EKF}}(m)$	2.8468	2.4598	2.0838
$\mathrm{RMSE}_{\mathrm{IMM}}(m)$	0.78805	1.4721	1.1043

Table 4.3: RMSE errors associated to the results in figures 4.7(a), 4.7(b) and 4.7(c).

under consideration the KF can be more accurate than IMM as happen in route 1 and route 3. However this kind of scenarios cannot occur because if high shadowing noise is considered also the variance of the position estimates must be higher.



General Parameters

Sigma_shad (dB)	30
ML Standard Deviation (m)	3
Sensor Activation (sg)	5
Target sampling (sg)	0.5
Sensor Groups	2
Power Path Loss Exponent	2
Distance Path Loss Exponent	2
P0 (dBm)	-63.2
Select route	route 1

KF and EKF parameters

KF_sigma (m/s^2)	0.1
EKF_sigma1 (m/s^2)	0.5
EKF_sigma2 (°/s^2)	0.2

IMM-CT Parameters

Models transition matrix	0.95	0.05
	0.10	0.90
Initial Model Probabilities	0.5	0.5

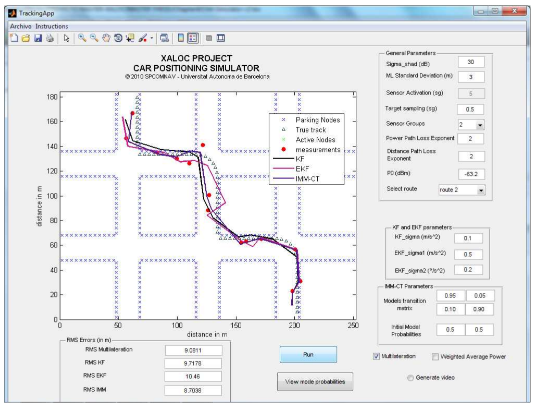
[✓ Multilateration] Weighted Average Power

Generate video

RMS Errors (in m)

RMS Multilateration	8.8954
RMS KF	7.3217
RMS EKF	9.7907
RMS IMM	8.5713

(a) Route 1



General Parameters

Sigma_shad (dB)	30
ML Standard Deviation (m)	3
Sensor Activation (sg)	5
Target sampling (sg)	0.5
Sensor Groups	2
Power Path Loss Exponent	2
Distance Path Loss Exponent	2
P0 (dBm)	-63.2
Select route	route 2

KF and EKF parameters

KF_sigma (m/s^2)	0.1
EKF_sigma1 (m/s^2)	0.5
EKF_sigma2 (°/s^2)	0.2

IMM-CT Parameters

Models transition matrix	0.95	0.05
	0.10	0.90
Initial Model Probabilities	0.5	0.5

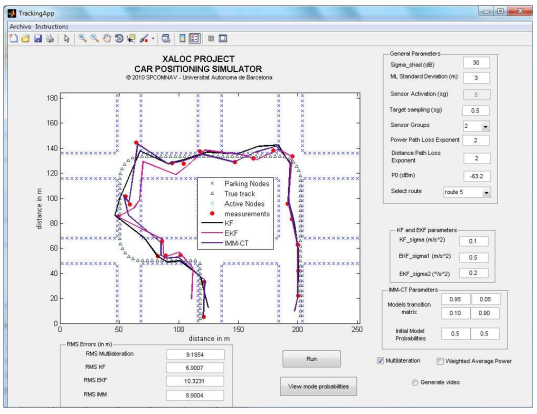
[✓ Multilateration] Weighted Average Power

Generate video

RMS Errors (in m)

RMS Multilateration	9.0811
RMS KF	9.7178
RMS EKF	10.46
RMS IMM	8.7038

(b) Route 2



General Parameters

Sigma_shad (dB)	30
ML Standard Deviation (m)	3
Sensor Activation (sg)	5
Target sampling (sg)	0.5
Sensor Groups	2
Power Path Loss Exponent	2
Distance Path Loss Exponent	2
P0 (dBm)	-63.2
Select route	route 5

KF and EKF parameters

KF_sigma (m/s^2)	0.1
EKF_sigma1 (m/s^2)	0.5
EKF_sigma2 (°/s^2)	0.2

IMM-CT Parameters

Models transition matrix	0.95	0.05
	0.10	0.90
Initial Model Probabilities	0.5	0.5

[✓ Multilateration] Weighted Average Power

Generate video

RMS Errors (in m)

RMS Multilateration	9.1854
RMS KF	6.9007
RMS EKF	10.3231
RMS IMM	8.9004

(c) Route 5

Figure 4.8: Tracking simulation with $\sigma_{shad} = 30\,dB$.

	Route 1	Route 2	Route 3
RMSE$_z(m)$	8.8954	9.0811	9.1854
RMSE$_{\text{KF}}(m)$	7.3217	9.7178	6.9007
RMSE$_{\text{EKF}}(m)$	9.7907	10.46	10.3231
RMSE$_{\text{IMM}}(m)$	8.5713	8.7038	8.9004

Table 4.4: RMSE errors associated to the results in figures 4.8(a), 4.8(b) and 4.8(c).

The values σ^2_{shad} and σ_z are somehow related. In other words σ_z must be fitted to σ^2_{shad} as σ_z is a design parameter. For instance, if $\sigma^2_z = 0$ it is clear that σ_{shad} must be zero (because the position estimates do not have any variance) and therefore the position measurements obtained by multilateration and the position estimations obtained by the tracking methods are the same as shown in 4.9 with the RMSE errors in table 4.5.

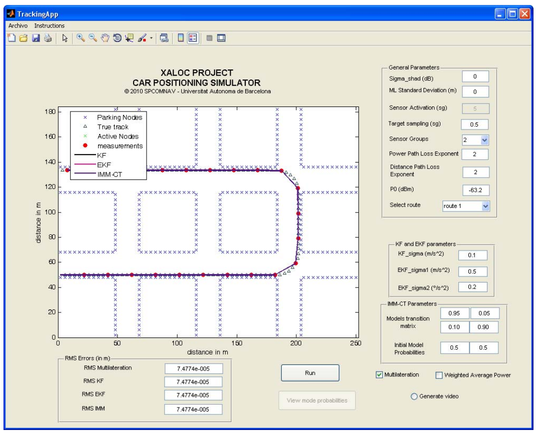
	Route 1	Route 2	Route 3
RMSE$_z(m)$	0	0	0
RMSE$_{\text{KF}}(m)$	0	0	0
RMSE$_{\text{EKF}}(m)$	0	0	0
RMSE$_{\text{IMM}}(m)$	0	0	0

Table 4.5: RMSE errors associated to the results in figures 4.9(a), 4.9(b) and 4.9(c).

Increasing σ^2_z deals with an increase of σ^2_{shad}. The value of σ_z to be considered must reflect to the reality with σ^2_{shad}. In other words, in practice we have that the RSSI is dependent on σ^2_{shad}. But if we do not know σ^2_{shad} because perhaps we do not have enough measurements to obtain the wireless channel statistics, then we should adjust the value of σ_z to fit the estimation positions with the true positions. It is worth recalling that multilateration is dependent on $|sigma_{shad}$ and the tracking is dependent on both z and σ^2_z in the simulations.

Case 3. σ_z dependency: Here we show how the values of σ_z have an influence on the position estimations obtained by KF, EKF and IMM. Figure 4.10 shows the tracking for $\sigma_z = 30\,m$ and $\sigma^2_{shad} = 2\,dB$ with the associated RMSE values in table 4.6 and figure 4.11 are the results for the tracking with $\sigma_z = 30\,m$ and $\sigma^2_{shad} = 30\,dB$ with the corresponding RMSE values shown in table 4.7.

Considering for instance the values $\sigma_z = 30\,m$ and $\sigma^2_{shad} = 2\,dB$ has no sense. It is because a small deviation on the RSSI due to the shadowing effect (the RSSI variation range is $RSSI \pm$



General Parameters
Sigma_shad (dB): 0
ML Standard Deviation (m): 0
Sensor Activation (sg): 5
Target sampling (sg): 0.5
Sensor Groups: 2
Power Path Loss Exponent: 2
Distance Path Loss Exponent: 2
P0 (dBm): -63.2
Select route: route 1

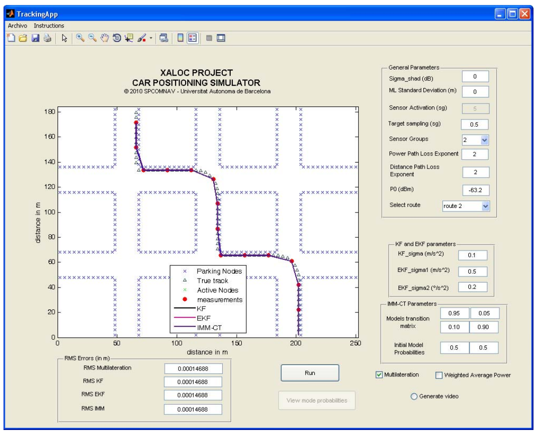
KF and EKF parameters
KF_sigma (m/s^2): 0.1
EKF_sigma1 (m/s^2): 0.5
EKF_sigma2 (°/s^2): 0.2

IMM-CT Parameters
Models transition matrix: 0.95 0.05 / 0.10 0.90
Initial Model Probabilities: 0.5 0.5

[✓] Multilateration
[ ] Weighted Average Power
( ) Generate video

RMS Errors (in m)
RMS Multilateration: 7.4774e-005
RMS KF: 7.4774e-005
RMS EKF: 7.4774e-005
RMS IMM: 7.4774e-005

(a) Route 1



General Parameters
Sigma_shad (dB): 0
ML Standard Deviation (m): 0
Sensor Activation (sg): 5
Target sampling (sg): 0.5
Sensor Groups: 2
Power Path Loss Exponent: 2
Distance Path Loss Exponent: 2
P0 (dBm): -63.2
Select route: route 2

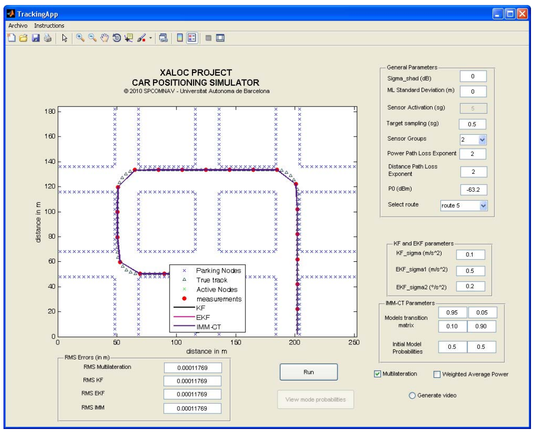
KF and EKF parameters
KF_sigma (m/s^2): 0.1
EKF_sigma1 (m/s^2): 0.5
EKF_sigma2 (°/s^2): 0.2

IMM-CT Parameters
Models transition matrix: 0.95 0.05 / 0.10 0.90
Initial Model Probabilities: 0.5 0.5

[✓] Multilateration
[ ] Weighted Average Power
( ) Generate video

RMS Errors (in m)
RMS Multilateration: 0.00014688
RMS KF: 0.00014688
RMS EKF: 0.00014688
RMS IMM: 0.00014688

(b) Route 2



General Parameters
Sigma_shad (dB): 0
ML Standard Deviation (m): 0
Sensor Activation (sg): 5
Target sampling (sg): 0.5
Sensor Groups: 2
Power Path Loss Exponent: 2
Distance Path Loss Exponent: 2
P0 (dBm): -63.2
Select route: route 5

KF and EKF parameters
KF_sigma (m/s^2): 0.1
EKF_sigma1 (m/s^2): 0.5
EKF_sigma2 (°/s^2): 0.2

IMM-CT Parameters
Models transition matrix: 0.95 0.05 / 0.10 0.90
Initial Model Probabilities: 0.5 0.5

[✓] Multilateration
[ ] Weighted Average Power
( ) Generate video

RMS Errors (in m)
RMS Multilateration: 0.00011769
RMS KF: 0.00011769
RMS EKF: 0.00011769
RMS IMM: 0.00011769

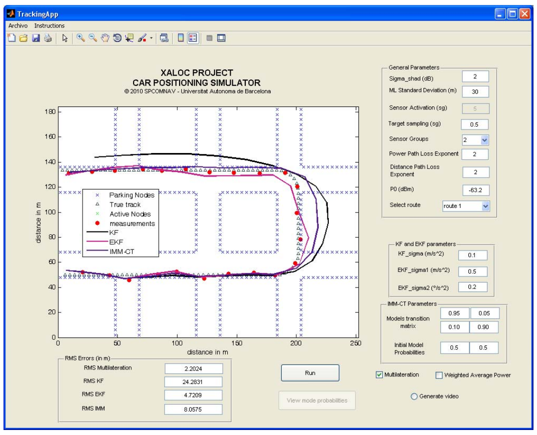
(c) Route 5

Figure 4.9: Tracking simulation with $\sigma^2_{shad} = 0\,dB$ and $\sigma_z = 0\,m$.

	Route 1	Route 2	Route 3
$\text{RMSE}_z(m)$	2.2024	2.1854	1.8432
$\text{RMSE}_{\text{KF}}(m)$	24.2831	8.1546	25.4734
$\text{RMSE}_{\text{EKF}}(m)$	4.7209	10.5379	11.4415
$\text{RMSE}_{\text{IMM}}(m)$	8.0575	6.4664	9.3138

Table 4.6: RMSE errors associated to the results in figures 4.10(a), 4.10(b) and 4.10(c).

$3\sigma_{shad}$) corresponds to a small variation on the distance between the receiver and the transmitter. For instance, consider the following RSSI mean value $P_{r1} = -72.7424\,dBm$ from the path loss



General Parameters

Sigma_shad (dB)	2
ML Standard Deviation (m)	30
Sensor Activation (sg)	5
Target sampling (sg)	0.5
Sensor Groups	2
Power Path Loss Exponent	2
Distance Path Loss Exponent	2
P0 (dBm)	-63.2
Select route	route 1

KF and EKF parameters

KF_sigma (m/s^2)	0.1
EKF_sigma1 (m/s^2)	0.5
EKF_sigma2 (°/s^2)	0.2

IMM-CT Parameters

Models transition matrix	0.95	0.05
	0.10	0.90
Initial Model Probabilities	0.5	0.5

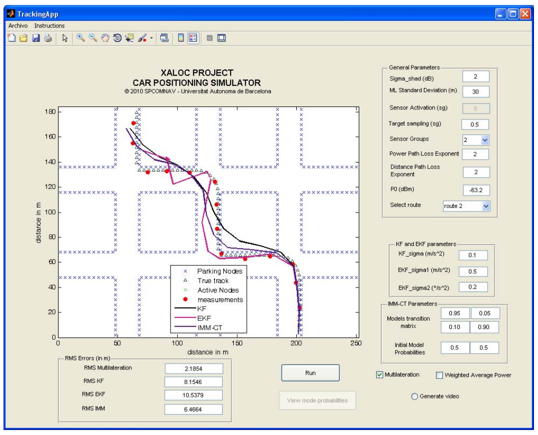
[✓ Multilateration] Weighted Average Power

Generate video

RMS Errors (in m)

RMS Multilateration	2.2024
RMS KF	24.2831
RMS EKF	4.7209
RMS IMM	8.0575

(a) Route 1



General Parameters

Sigma_shad (dB)	2
ML Standard Deviation (m)	30
Sensor Activation (sg)	5
Target sampling (sg)	0.5
Sensor Groups	2
Power Path Loss Exponent	2
Distance Path Loss Exponent	2
P0 (dBm)	-63.2
Select route	route 2

KF and EKF parameters

KF_sigma (m/s^2)	0.1
EKF_sigma1 (m/s^2)	0.5
EKF_sigma2 (°/s^2)	0.2

IMM-CT Parameters

Models transition matrix	0.95	0.05
	0.10	0.90
Initial Model Probabilities	0.5	0.5

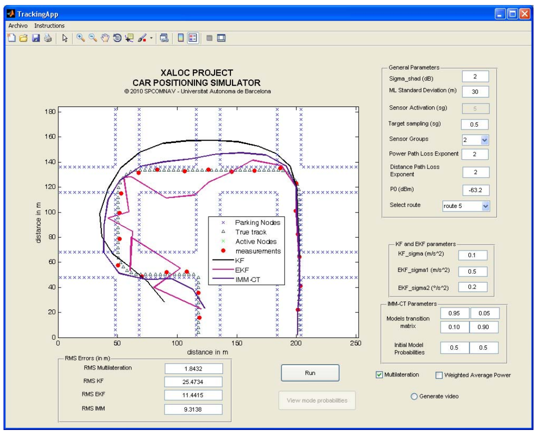
[✓ Multilateration] Weighted Average Power

Generate video

RMS Errors (in m)

RMS Multilateration	2.1854
RMS KF	8.1546
RMS EKF	10.5379
RMS IMM	6.4664

(b) Route 2



General Parameters

Sigma_shad (dB)	2
ML Standard Deviation (m)	30
Sensor Activation (sg)	5
Target sampling (sg)	0.5
Sensor Groups	2
Power Path Loss Exponent	2
Distance Path Loss Exponent	2
P0 (dBm)	-63.2
Select route	route 5

KF and EKF parameters

KF_sigma (m/s^2)	0.1
EKF_sigma1 (m/s^2)	0.5
EKF_sigma2 (°/s^2)	0.2

IMM-CT Parameters

Models transition matrix	0.95	0.05
	0.10	0.90
Initial Model Probabilities	0.5	0.5

[✓ Multilateration] Weighted Average Power

Generate video

RMS Errors (in m)

RMS Multilateration	1.8432
RMS KF	25.4734
RMS EKF	11.4415
RMS IMM	9.3138

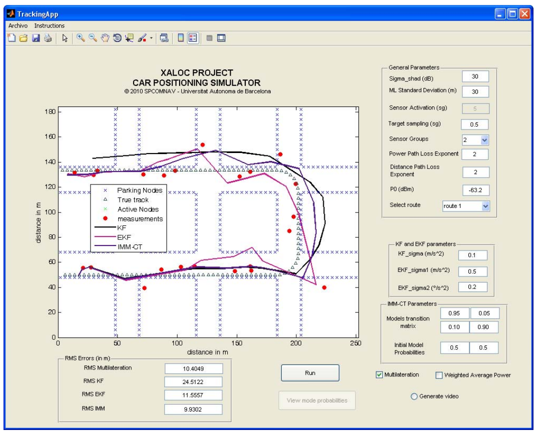
(c) Route 5

Figure 4.10: Tracking simulation with $\sigma^2_{shad} = 2\,dB$ and $\sigma_z = 30\,m$.

model with $P0 = -63.2\,dBm$, $\gamma_{rssi} = 2$ and $d = 3\,m$. The maximum RSSI value assuming $\sigma^2_{shad} = 2\,dB$ is $P_{r2} = P_{r1} - 3\sigma_{shad} = -76.9851\,dBm$ which corresponds to a maximum distance deviation of $d_2 = 4.8894\,m$. Hence a value of $\sigma_z = d_2 - d \approx 2\,m$ could be more appropriated instead of $\sigma_z = 30\,m$. Although it has no sense to choose the values $\sigma_z = 30\,m$ and $\sigma^2_{shad} = 2\,dB$ the set of simulations in figure 4.10 are shown in order to demonstrate that the behaviour of the algorithms depend on:

1. Position measurements $\mathbf{z}$ obtained by a position technique like Multilateration. If the position measurements have large errors it also leads large errors in the outputs of the tracking techniques.

2. Measurement noise covariance $\mathbf{R}$ related to σ_z.



General Parameters
Sigma_shad (dB) 30
ML Standard Deviation (m) 30
Sensor Activation (sg) 5
Target sampling (sg) 0.5
Sensor Groups 2
Power Path Loss Exponent 2
Distance Path Loss Exponent 2
P0 (dBm) -63.2
Select route route 1

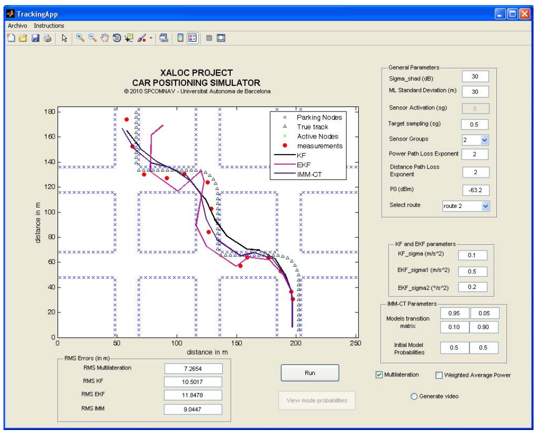
KF and EKF parameters
KF_sigma (m/s^2) 0.1
EKF_sigma1 (m/s^2) 0.5
EKF_sigma2 (°/s^2) 0.2

IMM-CT Parameters
Models transition matrix 0.95 0.05 / 0.10 0.90
Initial Model Probabilities 0.5 0.5

RMS Errors (in m)
RMS Multilateration 10.4049
RMS KF 24.5122
RMS EKF 11.5557
RMS IMM 9.9302

[✓] Multilateration [ ] Weighted Average Power
( ) Generate video

(a) Route 1



General Parameters
Sigma_shad (dB) 30
ML Standard Deviation (m) 30
Sensor Activation (sg) 5
Target sampling (sg) 0.5
Sensor Groups 2
Power Path Loss Exponent 2
Distance Path Loss Exponent 2
P0 (dBm) -63.2
Select route route 2

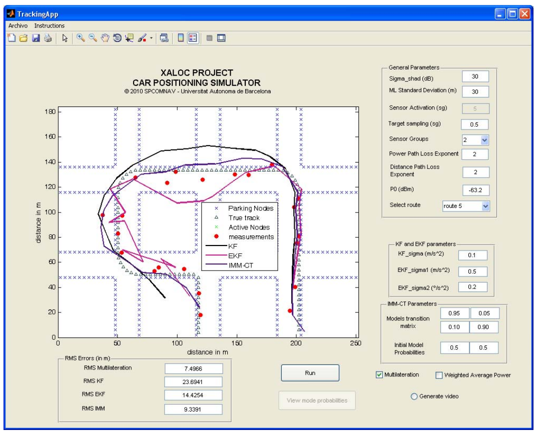
KF and EKF parameters
KF_sigma (m/s^2) 0.1
EKF_sigma1 (m/s^2) 0.5
EKF_sigma2 (°/s^2) 0.2

IMM-CT Parameters
Models transition matrix 0.95 0.05 / 0.10 0.90
Initial Model Probabilities 0.5 0.5

RMS Errors (in m)
RMS Multilateration 7.2654
RMS KF 10.5017
RMS EKF 11.8478
RMS IMM 9.0447

[✓] Multilateration [ ] Weighted Average Power
( ) Generate video

(b) Route 2



General Parameters
Sigma_shad (dB) 30
ML Standard Deviation (m) 30
Sensor Activation (sg) 5
Target sampling (sg) 0.5
Sensor Groups 2
Power Path Loss Exponent 2
Distance Path Loss Exponent 2
P0 (dBm) -63.2
Select route route 5

KF and EKF parameters
KF_sigma (m/s^2) 0.1
EKF_sigma1 (m/s^2) 0.5
EKF_sigma2 (°/s^2) 0.2

IMM-CT Parameters
Models transition matrix 0.95 0.05 / 0.10 0.90
Initial Model Probabilities 0.5 0.5

RMS Errors (in m)
RMS Multilateration 7.4966
RMS KF 23.6941
RMS EKF 14.4254
RMS IMM 9.3391

[✓] Multilateration [ ] Weighted Average Power
( ) Generate video

(c) Route 5

Figure 4.11: Tracking simulation with $\sigma^2_{shad} = 30\,dB$ and $\sigma_z = 30\,m$.

	Route 1	Route 2	Route 3
$\text{RMSE}_z(m)$	10.4049	7.2654	7.4966
$\text{RMSE}_{\text{KF}}(m)$	24.5122	10.5017	23.6941
$\text{RMSE}_{\text{EKF}}(m)$	11.55577	11.8478	14.4254
$\text{RMSE}_{\text{IMM}}(m)$	9.9302	9.0447	9.3391

Table 4.7: RMSE errors associated to the results in figures 4.11(a), 4.11(b) and 4.11(c).

The results shown in figure 4.11 show that for larger error in the position measurements larger errors appear in the position estimates obtained by the tracking algorithms. Re-call that

σ^2_{shad} and σ_z are related. Figure 4.11 with its corresponding RMSE errors in table 4.7 also shows that the behaviour of IMM-CT in terms of RMSE is the best in comparison with both KF and EKF. In cases when the shadowing noise can be high -i.e. the deviation of the position estimates is also high- it is necessary to adjust the process noise covariance matrix $\mathbf{Q}$ to achieve a good performance of the tracking estimators. Next we demonstrate the dependency of the tracking estimators under $\mathbf{U}$ ($\mathbf{Q}$ is directly proportional to $\mathbf{U}$).

Case 4. $\mathbf{U}_{KF}$ and $\mathbf{U}_{EKF}$ adaptation: On the one hand, when having larger position errors it is necessary to find those values for the acceleration noise covariance $\mathbf{U}$ that allow the tracking algorithms to adapt to the measurements and thus minimizing the RMSE. On the other hand $\mathbf{R}$ is not a design parameter since it is related to the measurement noise. It is worth recalling that the values of its diagonal are σ_z in m.

Therefore values to be chosen will be those that the algorithm performance in the noisy environment satisfies the designer. For example, with the following values for $\mathbf{U_{KF}}$ and $\mathbf{U_{EKF}}$ instead of the default ones:

$$\mathbf{U_{KF}} = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix} \quad \mathbf{U_{EKF}} = \begin{bmatrix} 6 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 3 \end{bmatrix} \tag{4.4}$$

the achieved RMSE errors shown in table 4.8 which corresponds to the simulations shown in figure 4.12 are much less than the ones seen in table 4.7 for the previous simulations in figure 4.11. As a result we can see that the chosen values in (4.4) are more suitable than the default ones when the noise is high. It is important to keep in mind that the performance of the tracking algorithms depends on $\mathbf{U}$ or σ_u. For example, if KF is used to follow a uniform motion model, increasing $\mathbf{U}_{KF}$ makes that the belief of the model decreases (believing more with the measurements) and viceversa decreasing $\mathbf{U}_{KF}$ means that the belief of the model increases. Figure 4.13 are the simulation results with the same values than the used in the simulations seen in figure fig:figure410 but with $\mathbf{U}_{EKF} = \mathbf{0}$ (all values in its diagonal are zero). One can see in figures 4.13(a) and 4.13(c) that the EKF tendency is to follow a nearly coordinated turn model, i.e. EKF deviates from the target trajectory. In figure 4.13(a) the KF provides better position estimations than EKF, achieving smaller RMSE errors as shown in the first columns of table 4.9. However as shown in table 4.9 corresponding to figure 4.13(c) the EKF achieves smaller RMSE errors than KF although the EKF track seems to be worst than the KF track. Hence a conclusion is that the performance of the tracking estimators is also dependent on the scenario under consideration; in this case the routes are different. For example, in figure 4.13(b) the EKF track is more ore less similar to the KF track, i.e. the EKF track does not deviates.

Concerning the IMM-CT performance it gets worst in comparison with respect to the results in the previous simulations in figure 4.11. This can be seen when comparing the achieved RMSE errors. The reason of that is because IMM-CT depends on the performance of of the combined filters. Thus if one of the filters gets worse then the IMM performance deteriorates. Increasing $\sigma_u^2 = 0.8\,m/s^2$, then the EKF performance improves implying also the IMM-CT performance as shown in figure 4.14 or numerically with the RMSE errors shown in table 4.10. One can see that the average positions errors of both KF and IMM-CT decreases significantly.

It is important to keep in mind that the way of finding suitable values of $\mathbf{U}_{KF,EKF}$ is by means of trial and error, there is no numerical way to find the optimum values of $\mathbf{U}_{KF,EKF}$ that achieve the minimum average position error. If one needs to apply the tracking algorithms in a real scenario the variance of the acceleration should be obtained by measurements or by means of trial and error through several campaign measurements. For example, a KF filter was used to track a car in a real scenario as detailed in chapter 5 and several campaign measurements were carried on to find the appropriate value of the acceleration noise variance.

	Route 1	Route 2	Route 3
RMSE$_z(m)$	9.0489	6.4985	7.578
RMSE$_{\text{KF}}(m)$	7.6325	8.69	6.3738
RMSE$_{\text{EKF}}(m)$	11.3928	9.8031	7.8065
RMSE$_{\text{IMM}}(m)$	7.2899	8.1902	6.2253

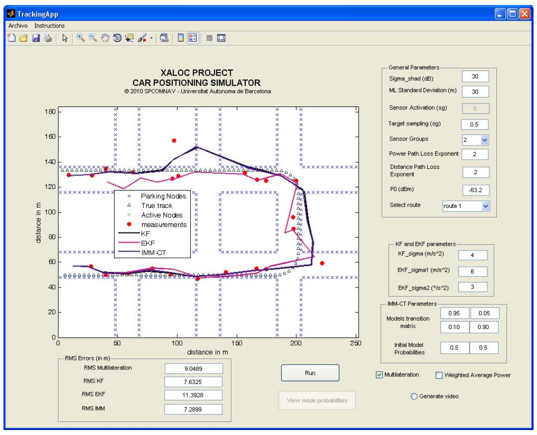
Table 4.8: RMSE errors associated to the results in figures 4.12(a), 4.12(b) and 4.12(c).

	Route 1	Route 2	Route 3
RMSE$_z(m)$	7.8532	7.1251	7.2047
RMSE$_{\text{KF}}(m)$	24.1103	10.2743	24.3379
RMSE$_{\text{EKF}}(m)$	31.2504	10.2306	17.501
RMSE$_{\text{IMM}}(m)$	24.959	10.0513	20.0396

Table 4.9: RMSE errors associated to the results in figures 4.13(a), 4.13(b) and 4.13(c).

Case 5. Adaptation of IMM-CT Parameters: The purpose of this simulation case is to deal with the IMM-CT performance, taking into account the following parameters:

1. the transition probability matrix $\pi_{\mathbf{CT}}$: the Markov chain (or state machine) transition probabilities between different states (models) and the probabilities to remain at the same



General Parameters
- Sigma_shad (dB): 30
- ML Standard Deviation (m): 30
- Sensor Activation (sg): 5
- Target sampling (sg): 0.5
- Sensor Groups: 2
- Power Path Loss Exponent: 2
- Distance Path Loss Exponent: 2
- P0 (dBm): -63.2
- Select route: route 1

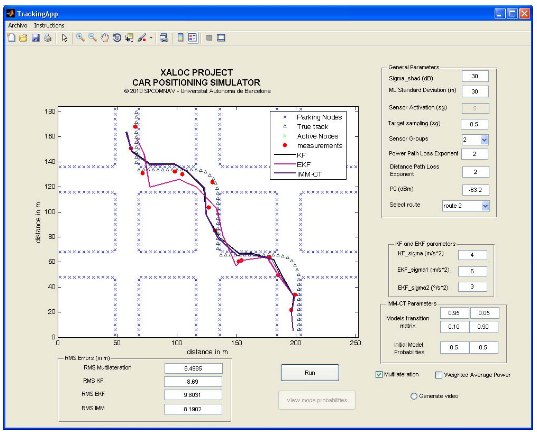
KF and EKF parameters
- KF_sigma (m/s^2): 4
- EKF_sigma1 (m/s^2): 6
- EKF_sigma2 (°/s^2): 3

IMM-CT Parameters
- Models transition matrix: 0.95 0.05 / 0.10 0.90
- Initial Model Probabilities: 0.5 0.5

[✓] Multilateration [ ] Weighted Average Power
( ) Generate video

RMS Errors (in m)
- RMS Multilateration: 9.0489
- RMS KF: 7.6325
- RMS EKF: 11.3928
- RMS IMM: 7.2899

(a) Route 1



General Parameters
- Sigma_shad (dB): 30
- ML Standard Deviation (m): 30
- Sensor Activation (sg): 5
- Target sampling (sg): 0.5
- Sensor Groups: 2
- Power Path Loss Exponent: 2
- Distance Path Loss Exponent: 2
- P0 (dBm): -63.2
- Select route: route 2

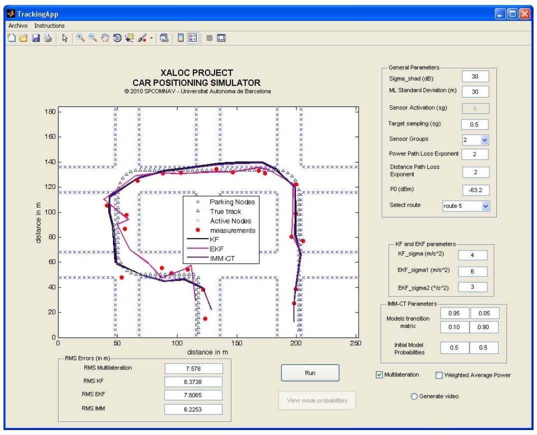
KF and EKF parameters
- KF_sigma (m/s^2): 4
- EKF_sigma1 (m/s^2): 6
- EKF_sigma2 (°/s^2): 3

IMM-CT Parameters
- Models transition matrix: 0.95 0.05 / 0.10 0.90
- Initial Model Probabilities: 0.5 0.5

[✓] Multilateration [ ] Weighted Average Power
( ) Generate video

RMS Errors (in m)
- RMS Multilateration: 6.4985
- RMS KF: 8.69
- RMS EKF: 9.8031
- RMS IMM: 8.1902

(b) Route 2



General Parameters
- Sigma_shad (dB): 30
- ML Standard Deviation (m): 30
- Sensor Activation (sg): 5
- Target sampling (sg): 0.5
- Sensor Groups: 2
- Power Path Loss Exponent: 2
- Distance Path Loss Exponent: 2
- P0 (dBm): -63.2
- Select route: route 5

KF and EKF parameters
- KF_sigma (m/s^2): 4
- EKF_sigma1 (m/s^2): 6
- EKF_sigma2 (°/s^2): 3

IMM-CT Parameters
- Models transition matrix: 0.95 0.05 / 0.10 0.90
- Initial Model Probabilities: 0.5 0.5

[✓] Multilateration [ ] Weighted Average Power
( ) Generate video

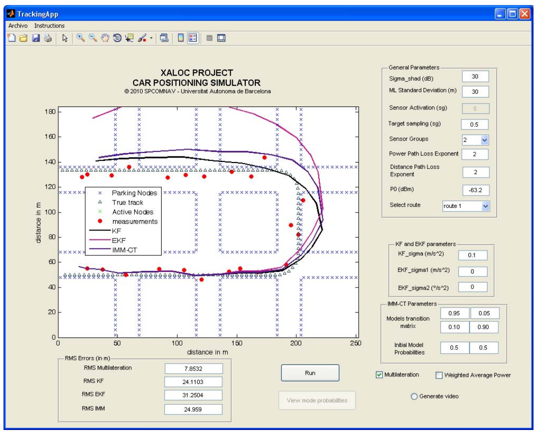
RMS Errors (in m)
- RMS Multilateration: 7.578
- RMS KF: 6.3738
- RMS EKF: 7.8065
- RMS IMM: 6.2253

(c) Route 5

Figure 4.12: Tracking simulation with $\sigma^2_{shad} = 30\,dB$, $\sigma_z = 30\,m$ and the process covariance matrices (4.4)

	Route 1	Route 2	Route 3
$\text{RMSE}_z(m)$	8.0485	9.6884	7.4144
$\text{RMSE}_{\text{KF}}(m)$	25.2994	11.6391	26.1548
$\text{RMSE}_{\text{EKF}}(m)$	11.4893	9.8404	9.5091
$\text{RMSE}_{\text{IMM}}(m)$	13.6132	10.6801	13.4004

Table 4.10: RMSE errors associated to the results in figures 4.14(a), 4.14(b) and 4.14(c).



XALOC PROJECT
CAR POSITIONING SIMULATOR
© 2010 SPCOMNAV - Universitat Autonoma de Barcelona

General Parameters
Sigma_shad (dB) 30
ML Standard Deviation (m) 30
Sensor Activation (sg) 5
Target sampling (sg) 0.5
Sensor Groups 2
Power Path Loss Exponent 2
Distance Path Loss Exponent 2
P0 (dBm) -63.2
Select route route 1

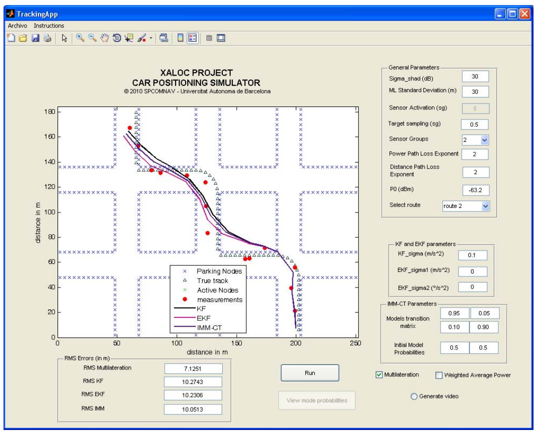
KF and EKF parameters
KF_sigma (m/s^2) 0.1
EKF_sigma1 (m/s^2) 0
EKF_sigma2 (°/s^2) 0

IMM-CT Parameters
Models transition matrix 0.95 0.05 / 0.10 0.90
Initial Model Probabilities 0.5 0.5

[✓] Multilateration
[ ] Weighted Average Power
○ Generate video

RMS Errors (in m)
RMS Multilateration 7.8532
RMS KF 24.1103
RMS EKF 31.2504
RMS IMM 24.959

(a) Route 1



XALOC PROJECT
CAR POSITIONING SIMULATOR
© 2010 SPCOMNAV - Universitat Autonoma de Barcelona

General Parameters
Sigma_shad (dB) 30
ML Standard Deviation (m) 30
Sensor Activation (sg) 5
Target sampling (sg) 0.5
Sensor Groups 2
Power Path Loss Exponent 2
Distance Path Loss Exponent 2
P0 (dBm) -63.2
Select route route 2

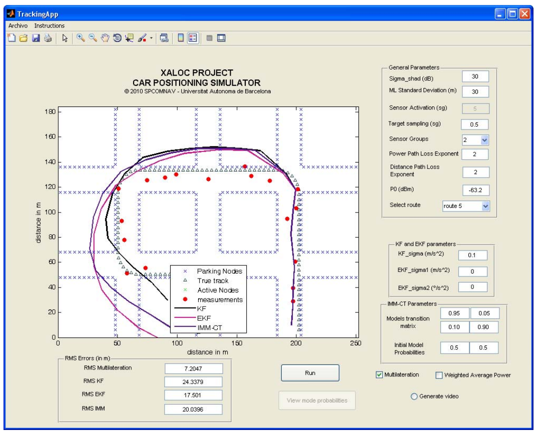
KF and EKF parameters
KF_sigma (m/s^2) 0.1
EKF_sigma1 (m/s^2) 0
EKF_sigma2 (°/s^2) 0

IMM-CT Parameters
Models transition matrix 0.95 0.05 / 0.10 0.90
Initial Model Probabilities 0.5 0.5

[✓] Multilateration
[ ] Weighted Average Power
○ Generate video

RMS Errors (in m)
RMS Multilateration 7.1251
RMS KF 10.2743
RMS EKF 10.2306
RMS IMM 10.0513

(b) Route 2



XALOC PROJECT
CAR POSITIONING SIMULATOR
© 2010 SPCOMNAV - Universitat Autonoma de Barcelona

General Parameters
Sigma_shad (dB) 30
ML Standard Deviation (m) 30
Sensor Activation (sg) 5
Target sampling (sg) 0.5
Sensor Groups 2
Power Path Loss Exponent 2
Distance Path Loss Exponent 2
P0 (dBm) -63.2
Select route route 5

KF and EKF parameters
KF_sigma (m/s^2) 0.1
EKF_sigma1 (m/s^2) 0
EKF_sigma2 (°/s^2) 0

IMM-CT Parameters
Models transition matrix 0.95 0.05 / 0.10 0.90
Initial Model Probabilities 0.5 0.5

[✓] Multilateration
[ ] Weighted Average Power
○ Generate video

RMS Errors (in m)
RMS Multilateration 7.2047
RMS KF 24.3379
RMS EKF 17.501
RMS IMM 20.0396

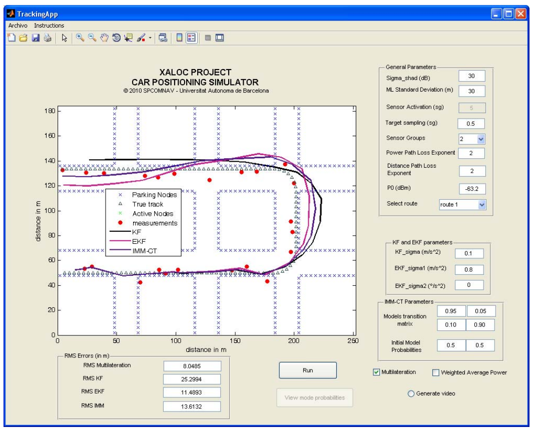
(c) Route 5

Figure 4.13: Tracking simulation with $\sigma^2_{shad} = 30\,dB$, $\sigma_z = 30\,m$, $\mathbf{Q_{KF}}$ =[0.1 0;0 0.1] and $\mathbf{Q_{EKF}} = \mathbf{0}$

state.

2. the model probabilities μ: the certainty probability of a model.

With respect to 1 we want to demonstrate that the performance of IMM is not dependent on $\pi_{\mathbf{CT}}$ as commented by [12]. For that suppose we choose the following transition matrix as the transpose of the used right now. We choose these values to represent the opposite case of the previous cases, i.e. the probability to go to model 2 from model 1 was 0.05 and now will be 0.10 whereas the probability to go to model 1 from model 2 was 0.10 and now will be 0.05. It does not matter what other values we choose since we will demonstrate that IMM efficiency is independent with the Markov chain probability matrix.



General Parameters
Sigma_shad (dB): 30
ML Standard Deviation (m): 30
Sensor Activation (sg): 5
Target sampling (sg): 0.5
Sensor Groups: 2
Power Path Loss Exponent: 2
Distance Path Loss Exponent: 2
P0 (dBm): -63.2
Select route: route 1

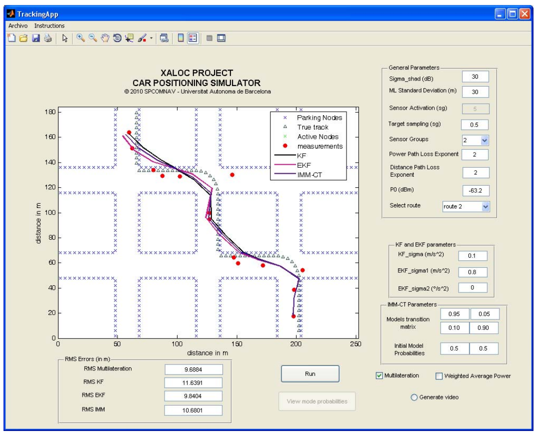
KF and EKF parameters
KF_sigma (m/s^2): 0.1
EKF_sigma1 (m/s^2): 0.8
EKF_sigma2 (°/s^2): 0

IMM-CT Parameters
Models transition matrix: 0.95 0.05 / 0.10 0.90
Initial Model Probabilities: 0.5 0.5

[✓] Multilateration [ ] Weighted Average Power

RMS Errors (in m)
RMS Multilateration: 8.0485
RMS KF: 25.2994
RMS EKF: 11.4893
RMS IMM: 13.6132

(a) Route 1



General Parameters
Sigma_shad (dB): 30
ML Standard Deviation (m): 30
Sensor Activation (sg): 5
Target sampling (sg): 0.5
Sensor Groups: 2
Power Path Loss Exponent: 2
Distance Path Loss Exponent: 2
P0 (dBm): -63.2
Select route: route 2

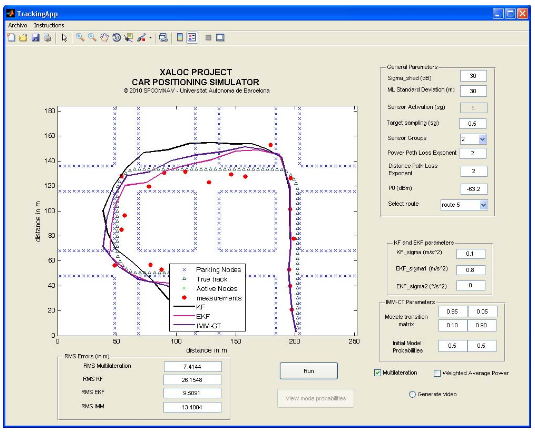
KF and EKF parameters
KF_sigma (m/s^2): 0.1
EKF_sigma1 (m/s^2): 0.8
EKF_sigma2 (°/s^2): 0

IMM-CT Parameters
Models transition matrix: 0.95 0.05 / 0.10 0.90
Initial Model Probabilities: 0.5 0.5

[✓] Multilateration [ ] Weighted Average Power

RMS Errors (in m)
RMS Multilateration: 9.6884
RMS KF: 11.6391
RMS EKF: 9.8404
RMS IMM: 10.6801

(b) Route 2



General Parameters
Sigma_shad (dB): 30
ML Standard Deviation (m): 30
Sensor Activation (sg): 5
Target sampling (sg): 0.5
Sensor Groups: 2
Power Path Loss Exponent: 2
Distance Path Loss Exponent: 2
P0 (dBm): -63.2
Select route: route 5

KF and EKF parameters
KF_sigma (m/s^2): 0.1
EKF_sigma1 (m/s^2): 0.8
EKF_sigma2 (°/s^2): 0

IMM-CT Parameters
Models transition matrix: 0.95 0.05 / 0.10 0.90
Initial Model Probabilities: 0.5 0.5

[✓] Multilateration [ ] Weighted Average Power

RMS Errors (in m)
RMS Multilateration: 7.4144
RMS KF: 26.1548
RMS EKF: 9.5091
RMS IMM: 13.4004

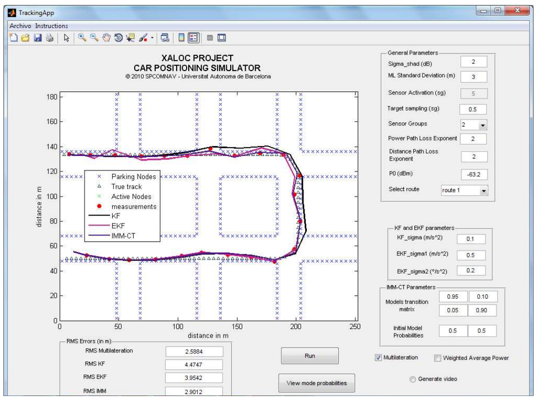
(c) Route 5

Figure 4.14: Tracking simulation with $\sigma^2_{shad} = 30\,dB$, $\sigma_z = 30\,m$, $\mathbf{Q_{KF}}$ =[0.1 0;0 0.1] and $\mathbf{Q_{EKF}}$ =[0.8 0 0;0 0.8 0;0 0 0]

$$\pi_{\mathbf{CT}} = \pi'_{\mathbf{CT}} = \begin{bmatrix} 0.95 & 0.10 \\ 0.05 & 0.90 \end{bmatrix} \tag{4.5}$$

Introducing the values of (4.5) into the simulator and keeping the other input parameters to their default values gives the results shown in figure 4.15 with the corresponding RMSE errors in table 4.11. One can see that the results are similar as in figure 4.6 in the order of few *cm* (not equal because the output of the filters are estimates). One can see that the RMSE errors of the IMM are also similar comparing table 4.11 and table 4.2.

The idea to keep the rest of the input parameters to the default values is to avoid the errors induced by high noise in the positions measurements that could make difficult the analysis of IMM estimator when varying its parameters.



General Parameters

Sigma_shad (dB)	2
ML Standard Deviation (m)	3
Sensor Activation (sg)	5
Target sampling (sg)	0.5
Sensor Groups	2
Power Path Loss Exponent	2
Distance Path Loss Exponent	2
P0 (dBm)	-63.2
Select route	route 1

KF and EKF parameters

KF_sigma (m/s^2)	0.1
EKF_sigma1 (m/s^2)	0.5
EKF_sigma2 (°/s^2)	0.2

IMM-CT Parameters

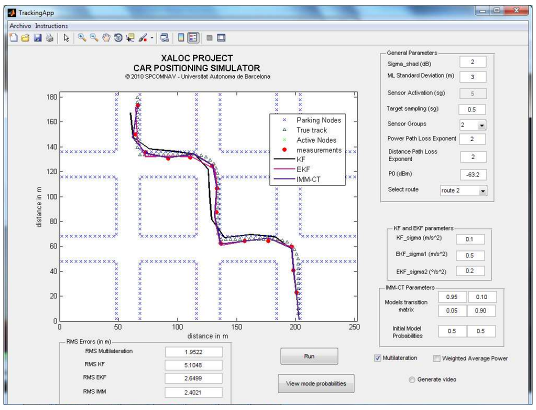
Models transition matrix	0.95	0.10
	0.05	0.90
Initial Model Probabilities	0.5	0.5

[✓] Multilateration [ ] Weighted Average Power

RMS Errors (in m)

RMS Multilateration	2.5884
RMS KF	4.4747
RMS EKF	3.9542
RMS IMM	2.9012

(a) Route 1



General Parameters

Sigma_shad (dB)	2
ML Standard Deviation (m)	3
Sensor Activation (sg)	5
Target sampling (sg)	0.5
Sensor Groups	2
Power Path Loss Exponent	2
Distance Path Loss Exponent	2
P0 (dBm)	-63.2
Select route	route 2

KF and EKF parameters

KF_sigma (m/s^2)	0.1
EKF_sigma1 (m/s^2)	0.5
EKF_sigma2 (°/s^2)	0.2

IMM-CT Parameters

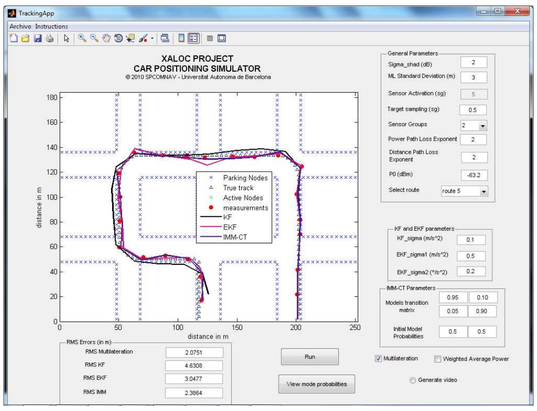
Models transition matrix	0.95	0.10
	0.05	0.90
Initial Model Probabilities	0.5	0.5

[✓] Multilateration [ ] Weighted Average Power

RMS Errors (in m)

RMS Multilateration	1.9522
RMS KF	5.1048
RMS EKF	2.6499
RMS IMM	2.4021

(b) Route 2



General Parameters

Sigma_shad (dB)	2
ML Standard Deviation (m)	3
Sensor Activation (sg)	5
Target sampling (sg)	0.5
Sensor Groups	2
Power Path Loss Exponent	2
Distance Path Loss Exponent	2
P0 (dBm)	-63.2
Select route	route 5

KF and EKF parameters

KF_sigma (m/s^2)	0.1
EKF_sigma1 (m/s^2)	0.5
EKF_sigma2 (°/s^2)	0.2

IMM-CT Parameters

Models transition matrix	0.95	0.10
	0.05	0.90
Initial Model Probabilities	0.5	0.5

[✓] Multilateration [ ] Weighted Average Power

RMS Errors (in m)

RMS Multilateration	2.0751
RMS KF	4.6308
RMS EKF	3.0477
RMS IMM	2.3864

(c) Route 5

Figure 4.15: Tracking simulation with the default parameters, Ngroups= 2 and the chosen mode transition matrix in (4.5)

.

Regarding the model probabilities they represent the initial certainty probabilities p_1 and p_2 of the model 1 and the model 2 respectively (note that $p_1 + p_2 = 1$). Next we vary the model probabilities assuming initially that the target moves according to a uniform motion model with a probability of 70% whereas in the rest 30% the target will perform some turns, i.e. the target follows a nearly coordinated turn model. The IMM needs to have an initialization of the model probabilities and in practice the route of a target is unknown. We will show that the performance of IMM with $\mu_{\mathbf{CT}}$ does not vary significantly with the model probabilities since IMM adapts the model probabilities at each time instant k with the received measurements (with the likelihoods)

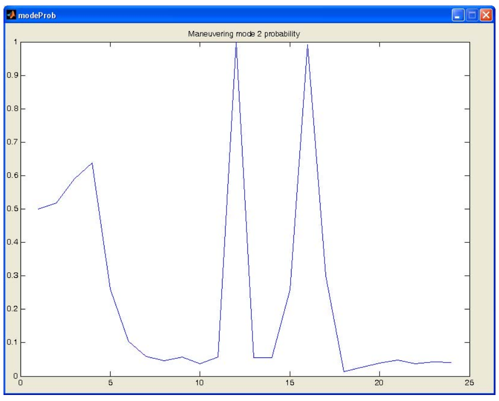
	Route 1	Route 2	Route 3
$\text{RMSE}_z(m)$	2.5884	1.9522	2.0751
$\text{RMSE}_{\text{KF}}(m)$	4.4747	5.1048	4.6308
$\text{RMSE}_{\text{EKF}}(m)$	3.9542	2.6499	3.0477
$\text{RMSE}_{\text{IMM}}(m)$	2.9012	2.4021	2.3864

Table 4.11: RMSE errors associated to the results in figures 4.15(a), 4.15(b) and 4.15(c).

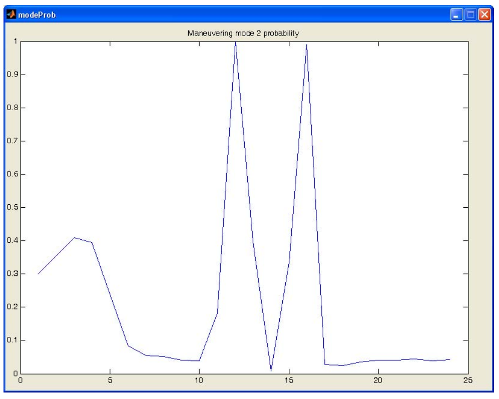
as demonstrated by (3.23).

$$\mu_{\mathbf{CT}} = \begin{bmatrix} 0.7 & 0.3 \end{bmatrix} \tag{4.6}$$

The results can be seen at the bottom of the figure 4.16. It compares the EKF model probabilities estimated by IMM between the default model probability and the newest one. These model probabilities corresponds to the route 1 with the same parameters as in the simulation shown in figure 4.6.



(a) EKF mode probabilities with the default mode probabilities: 0.5



(b) EKF mode probabilities with the new mode probability: 0.3

Figure 4.16: EKF Mode probabilities comparison between the original and the one shown in (4.6)

.

One can see that there the difference is almost small. Furthermore these results are similar to the ones shown in figure 3.7. It can be seen that the initial EKF mode probability of figure 4.16(b) is the one shown in the second column of (4.6). Then the model probabilities decreases upto the first peak. It is because the model probability of the KF increses, .i.e. the the uniform motion model followed by the KF is more certain than the nearly coordinated turn model followed by the EKF. It has sense since the target is moving eastwards upto the third crossing. Then the target performs a coordinated turn to the left and the EKF mode probability increases since the CT model is more certain than the uniform motion model. Then the target moves northwards upto the next crossing which performs again another coordinated turn to the left. It corresponds to the second peak. After the second turn the target moves westwards following a uniform motion resulting in a decrease of the EKF mode probability (or the EKF likelihood) and an increase of the KF mode probability (or the KF likelihood). The likelihood gives the probability that the position estimated by each tracking filter be certain. Thus if a filter has a likelihood greater it means that its estimates are close to the true positions.

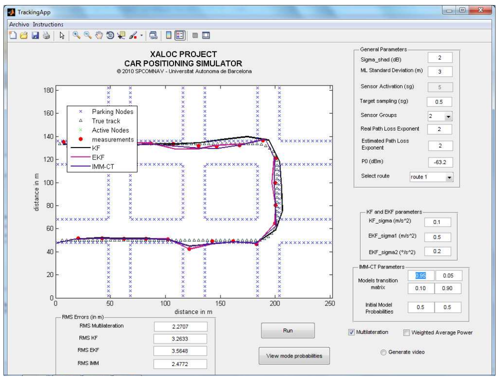
Case 6. γ_{RSSI} and $\hat{\gamma}_{RSSI}$ discrepancy: At this point we know that the estimation of the target position is found by means of a position technique that takes into account the distances to several anchor nodes. These distances are found using using both the RSSI which is related to the path loss model as shown in (4.7):

$$RSSI = P_0 - 10\gamma_{RSSI} log_{10} d_{t-r} - v, \tag{4.7}$$

where d_{t-r} is the distance between the transmitter and the receiver and v is a lognormal variable modelling the shadowing noise. On the one hand, the problem of estimating the distances is that the user needs to know the corresponding value of $\hat{\gamma}_{RSSI}$ associated to the each RSSI. In other words an estimate $\hat{\gamma}_{RSSI}$ is needed since the real one is unknown. On the other hand it is quite difficult to obtain this value since the scenario is randomly changing every time. Also it might exist different $\gamma_{RSSI,N}$ for each of the wireless links between every anchor and the receiver.

The simulator allows the possibility to realize simulations modifying both $\hat{\gamma}_{RSSI}$ and γ_{RSSI} with the parameters "Real Path Loss Exponent" and "Estimated Path Loss Exponent". We will show an example when the path loss exponent associated to the RSSI and the path loss exponent used to estimate the distances are different, i.e. $\gamma_{RSSI} \neq \hat{\gamma}_{RSSI}$. For that we consider only two routes, the same input control parameters than in figure 4.6 except the **Estimated Path Loss Exponent** that will be set for instance to 3. The results are shown respectively in figures 4.17 and 4.18 with their corresponding RMSE errors shown in table 4.12.

It can be seen that a discrepancy between the real path loss exponent and the estimated path loss exponent implies an increase of the average position errors as seen in table 4.12. Moreover notice that the noise introduced to the system is very small compared in a real scenario. It means that in practice large position errors can be due to this discrepancy in addition to other factors such as the shadow fading, multipath and interference. Furthermore larger errors would be obtained if for instance different real path loss are different for each wireless link. Thus the user should have a vector $\hat{\gamma}_{\mathbf{RSSI}}$ complicating then the computation of the estimated position. Although the simulator does not take into account this effect the idea was to demonstrate what happens when both $\hat{\gamma}_{RSSI}$ and γ_{RSSI} differs.



General Parameters
Sigma_shad (dB): 2
ML Standard Deviation (m): 3
Sensor Activation (sg): 5
Target sampling (sg): 0.5
Sensor Groups: 2
Real Path Loss Exponent: 2
Estimated Path Loss Exponent: 2
P0 (dBm): -63.2
Select route: route 1

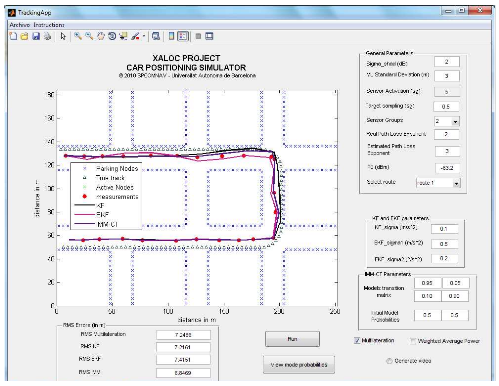
KF and EKF parameters
KF_sigma (m/s^2): 0.1
EKF_sigma1 (m/s^2): 0.5
EKF_sigma2 (°/s^2): 0.2

IMM-CT Parameters
Models transition matrix: 0.95 0.05 / 0.10 0.90
Initial Model Probabilities: 0.5 0.5

[✓] Multilateration [ ] Weighted Average Power
( ) Generate video

RMS Errors (in m)
RMS Multilateration: 2.2707
RMS KF: 3.2633
RMS EKF: 3.5648
RMS IMM: 2.4772

(a) Route 1: Positioning with $\hat{\gamma}_{RSSI} = \gamma_{RSSI} = 2$



General Parameters
Sigma_shad (dB): 2
ML Standard Deviation (m): 3
Sensor Activation (sg): 5
Target sampling (sg): 0.5
Sensor Groups: 2
Real Path Loss Exponent: 2
Estimated Path Loss Exponent: 3
P0 (dBm): -63.2
Select route: route 1

KF and EKF parameters
KF_sigma (m/s^2): 0.1
EKF_sigma1 (m/s^2): 0.5
EKF_sigma2 (°/s^2): 0.2

IMM-CT Parameters
Models transition matrix: 0.95 0.05 / 0.10 0.90
Initial Model Probabilities: 0.5 0.5

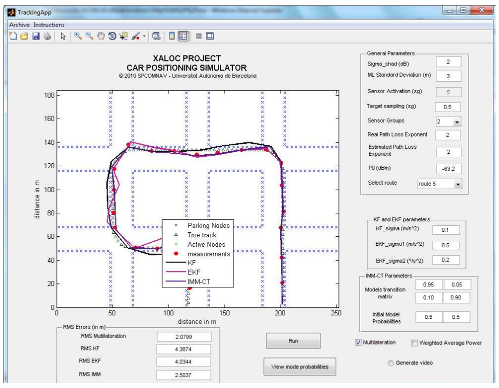
[✓] Multilateration [ ] Weighted Average Power
( ) Generate video

RMS Errors (in m)
RMS Multilateration: 7.2486
RMS KF: 7.2161
RMS EKF: 7.4151
RMS IMM: 6.8469

(b) Route 1: Positioning with $\hat{\gamma}_{RSSI} = 3, \gamma_{RSSI} = 2$

Figure 4.17: Route 1. Comparison between equal and different values of $\hat{\gamma}_{RSSI}$ and γ_{RSSI}

.



General Parameters
Sigma_shad (dB): 2
ML Standard Deviation (m): 3
Sensor Activation (sg): 5
Target sampling (sg): 0.5
Sensor Groups: 2
Real Path Loss Exponent: 2
Estimated Path Loss Exponent: 2
P0 (dBm): -63.2
Select route: route 5

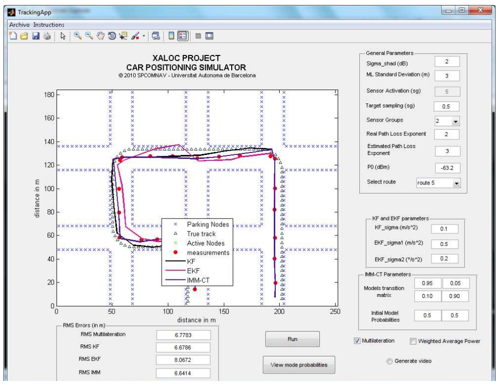
KF and EKF parameters
KF_sigma (m/s^2): 0.1
EKF_sigma1 (m/s^2): 0.5
EKF_sigma2 (°/s^2): 0.2

IMM-CT Parameters
Models transition matrix: 0.95 0.05 / 0.10 0.90
Initial Model Probabilities: 0.5 0.5

[✓] Multilateration [ ] Weighted Average Power
( ) Generate video

RMS Errors (in m)
RMS Multilateration: 2.0799
RMS KF: 4.3874
RMS EKF: 4.0344
RMS IMM: 2.5037

(a) Route 3: Positioning with $\hat{\gamma}_{RSSI} = \gamma_{RSSI} = 2$



General Parameters
Sigma_shad (dB): 2
ML Standard Deviation (m): 3
Sensor Activation (sg): 5
Target sampling (sg): 0.5
Sensor Groups: 2
Real Path Loss Exponent: 2
Estimated Path Loss Exponent: 3
P0 (dBm): -63.2
Select route: route 5

KF and EKF parameters
KF_sigma (m/s^2): 0.1
EKF_sigma1 (m/s^2): 0.5
EKF_sigma2 (°/s^2): 0.2

IMM-CT Parameters
Models transition matrix: 0.95 0.05 / 0.10 0.90
Initial Model Probabilities: 0.5 0.5

[✓] Multilateration [ ] Weighted Average Power
( ) Generate video

RMS Errors (in m)
RMS Multilateration: 6.7783
RMS KF: 6.6786
RMS EKF: 8.0672
RMS IMM: 6.6414

(b) Route 3: Positioning with $\hat{\gamma}_{RSSI} = 3, \gamma_{RSSI} = 2$

Figure 4.18: Route 3. Comparison between equal and different values of $\hat{\gamma}_{RSSI}$ and γ_{RSSI}

.

4.3 Summary

This chapter can be summarized as follows:

- A matlab simulator has been developed to test and validate both the positioning techniques and the set of tracking filters described in chapter 3.
- Several cases or scenarios have been proved in order to test the performance of the set of algorithms, specially the IMM-CT algoritm. The following main cases have been consid-

	$\gamma_{RSSI} = 2, \hat{\gamma}_{RSSI} = 2$		$\gamma_{RSSI} = 2, \hat{\gamma}_{RSSI} = 3$	
	Route 1	Route 3	Route 1	Route 3
$\text{RMSE}_z(m)$	2.2707	2.0799	7.2486	6.7783
$\text{RMSE}_{\text{KF}}(m)$	3.2633	4.3874	7.2161	6.6786
$\text{RMSE}_{\text{EKF}}(m)$	3.5648	4.0344	7.4151	8.0672
$\text{RMSE}_{\text{IMM}}(m)$	2.4772	2.5037	6.8469	6.6414

Table 4.12: Comparison of the positioning between equal and different values of γ_{RSSI} and $\hat{\gamma}_{RSSI}$ for both Route 1 and Route 3.

ered:

- Case2: Impact of σ_{shad} variation: study the performance of the system when the noise of the received power varies.
- Case3: σ_z dependency: study the performance of the system for different standard deviation of the position measurements.
- Case4: Adaptation of $\mathbf{U}_{KF}$ and $\mathbf{U}_{EKF}$: analysing the efficiency of the set of tracking filters taking into account the dependency on the acceleration noise covariance (the variance of the process acceleration noise).
- Case5: Adaptation of IMM-CT parameters: Validation of the performance of the IMM algorithm depending on the IMM-CT parameters: $\pi_{\mathbf{CT}}$ and $\mu_{\mathbf{CT}}$.
- Case6: γ_{RSSI} and $\hat{\gamma}_{RSSI}$ discrepancy: Study the performance of Multilateration when the real and the estimated path loss exponents differs.

- It is demonstrated that IMM is the best tracking solution because it provides the smallest RMSE in most of the cases. It does not mean that IMM is always the best. Its performance is dependent on both the scenario and the performance of each filter. For example, IMM performance is dependent on the both KF and EKF performances. In conclusion IMM is a novel solution for tracking targets because it evaluates the measurement over different filters and gives a high likelihood to the one that most matches to the true positions.

- The case6 is considered because that effect is quite common in practice. It demonstrates that the average position error increases significantly assuming low shadowing noise. Hence it demonstrates that multilateration performance is very bad in practice.

Chapter 5

Experimental development

This chapter deals with the implementation of the localization system carried on during a live demonstration on July 7^{th} at the UAB campus. The idea of the localization system is to use some of the techniques discussed in Chapter 3 in a real scenario. Since the live demonstration is realized along a straight road without curves, only one single Kalman filter with a linear model is used. The sections included in this chapter are the following:

1. Scenario Description: this section shows the map of the zone where the measurement campaign is carried on as well as an explanation of the scenario under consideration.
2. Measurement Campaign: this section gives an explanation of the measurement campaign.
3. Implementation of a Java-based navigator: this section explains how the full navigator is built.
4. Experimental Validation: this section is the demonstration of the developed localization system.
5. Photos of the demonstration day.

5.1 Scenario Description

Figure 5.1 shows the main idea of this scenario which is composed basically of the following elements:

- Reference nodes (occupation sensors in the figure or parking sensors) with known coordinates located at each parking slot. These nodes have a magnetic sensor that detect if there is a car or not. These nodes transfer to the supernode the state of the parking (busy/not busy).

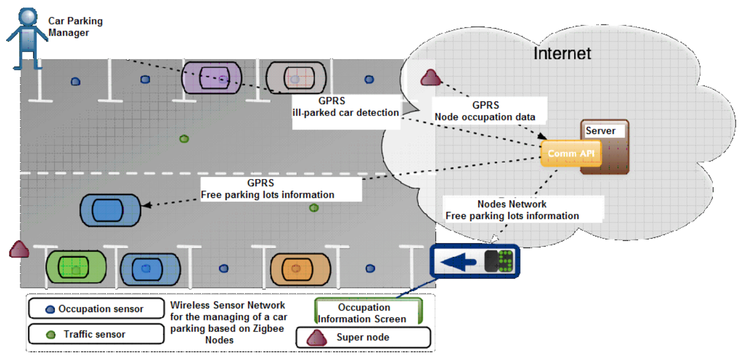

Car Parking Manager
Internet
GPRS
ill-parked car detection
GPRS
Node occupation data
Server
Comm API
GPRS
Free parking lots information
Nodes Network
Free parking lots information
Occupation sensor
Traffic sensor
Wireless Sensor Network for the managing of a car parking based on Zigbee Nodes
Occupation Information Screen
Super node

Figure 5.1: Scenario Description.

- The unknown node brought by the user or integrated in the car. This node must be real-time tracked to know from the central site through which route it moves. In this application the user only can drive in one direction.
- The super node whose actions are mainly to relay the car detection information to the central server.
- The central server which have a database to store for each parking sensor the following information: the ID(*Identificator*), the state of the parking, the coordinates, and the MAC address. The communication between either the super nodes and the unknown node is carried on via 3G(*Third Generation) or GPRS(General Packet Radio Service*).

The communication between the parking sensors and the superNode is carried on by World Sensing is realized through multihop. The detection of the parking is done by World Sensing as well.

The WSN used in the XALOC live demonstration is composed by a total of 18 sensors manufactured by the company WorldSensing; each sensor located at each parking slot. Regarding the localization strategy, each parking sensor sends periodically broadcast messages containing its node ID(*Identification*) in addition to other parameters. Then the user terminal at the vehicle is equipped with a 3G USB(*Universal Serial Bus*) dongle and a sensor node that performs the following:

- Measuring the RSSI of the received broadcast messages.
- Estimating the real-time vehicle position with the different RSSI's and parking sensor's IDs.
- Position information is used to extract parking information from XALOC database server.
- Real-time mapping of the vehicle position on the map previously downloaded from GoogleMaps server.

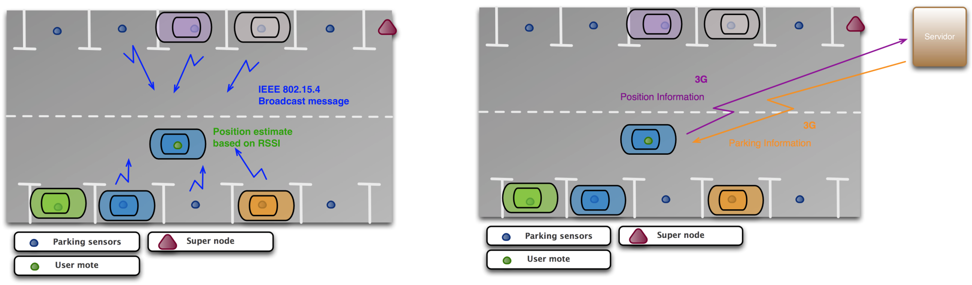


Figure 5.2: Localization Strategy

- Real-time mapping of the parking state information downloaded from the server via 3G on a the map.

The user terminal performs all the previous operations by means of a developed software that is discussed in section 5.3.

Figure 5.2 gives the idea of the localization strategy using a WSN.

The measurements zone is the fire department parking shown in the map of the figure 5.3. The parking areas in this area are in battery instead of on-line car. Therefore 18 online cars are marked as shown in the figures 5.4(a), 5.4(b) and 5.4(c). Figure 5.5(a) shows one of the sensors located in one of the parkings. This sensors are placed inside a robust box like the shown in figure 5.5(b) which are the ones seen in each of the parking slots in set of figures of 5.4. All sensor nodes are based on the ZigBit$^{\text{TM}}$600/800/900 MHz Wireless Module ATZB-900-B0 from Atmel industry.

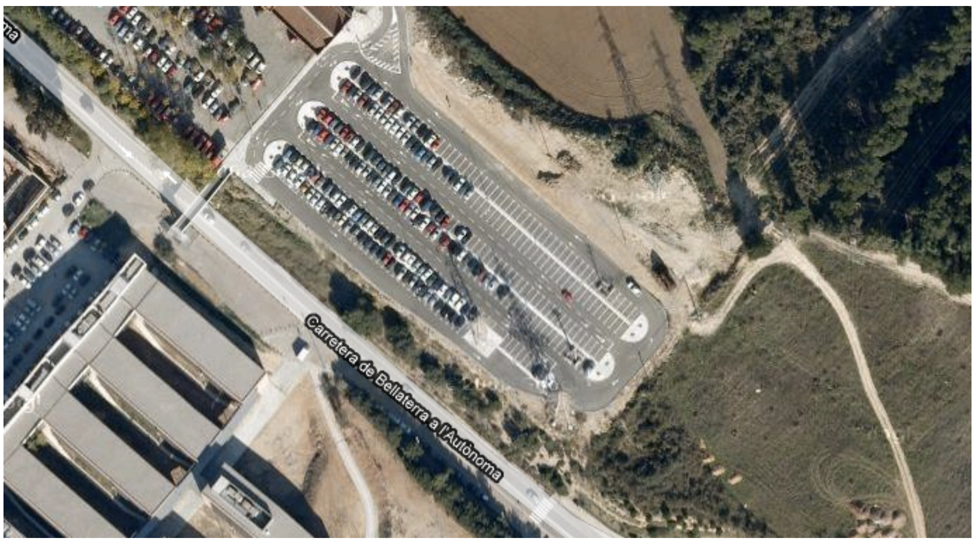
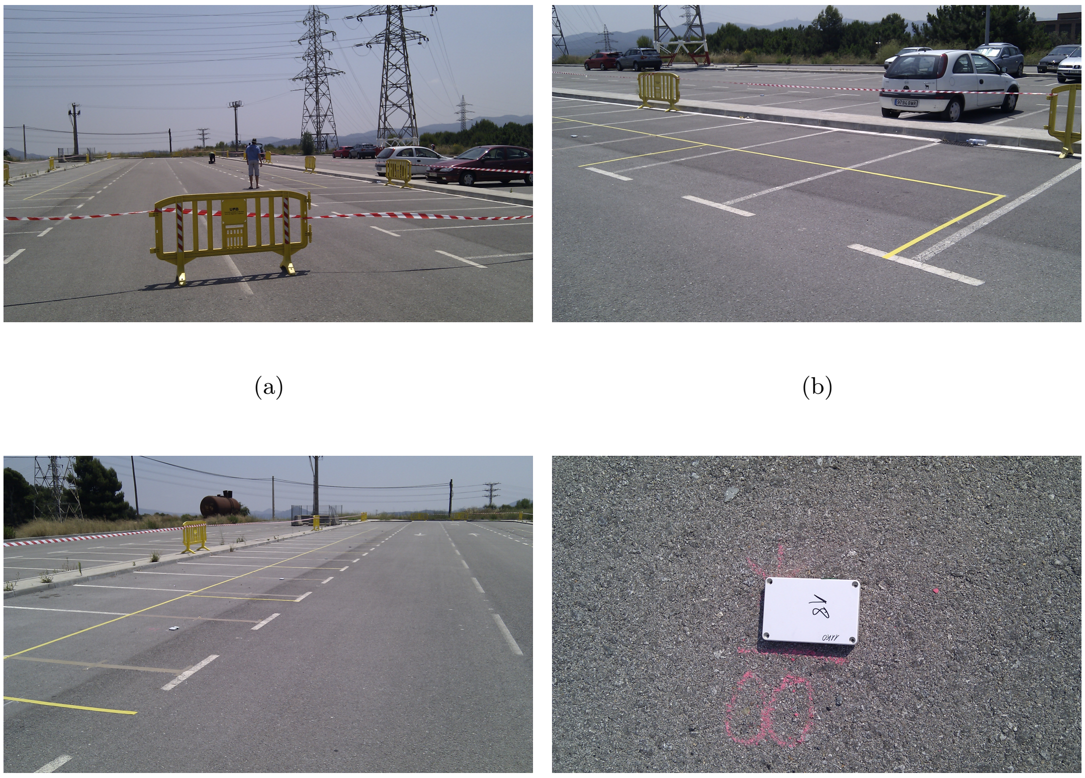


Figure 5.3: Scenario map.

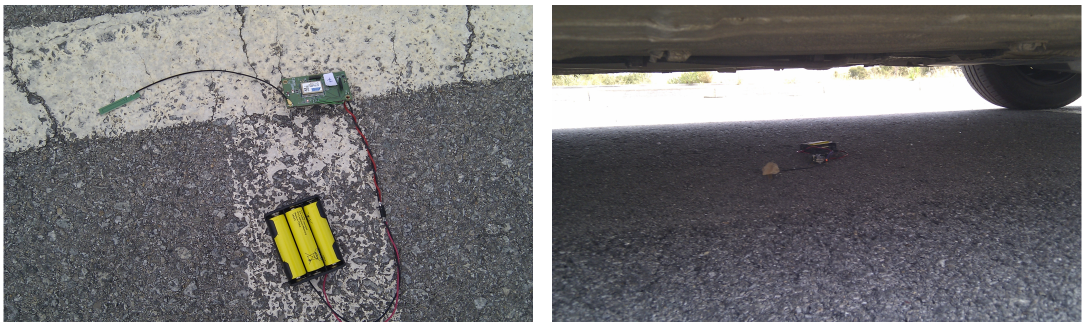


(a) (b)

(c) (d)

Figure 5.4: 5.4(a): Scenario Preparation, 5.4(b) and 5.4(d): online car marking, 5.4(d): Sensor package

.



(a) Photo 1 (b) Photo 2

Figure 5.5: Photo1: Parking sensor, Photo 2: a parking sensor placed below a parked car

.

The RF characteristics of this wireless module is shown in following table 5.1.

Parameters	**Condition**	**Range**	**Unit**
Frequency Band		779 to 787 868 to 868.6 902 to 928	MHz
Number of Channels		15	
Channel Spacing		2	MHz
Transmit Output Power		−11 to +11	dBm
Receiver Sensitivity	AWGN channel, PER=1%		
20 Kbits	PSDU length of 20 octets	-110	dBm
40 Kbits		-108	
100 Kbits		-101	
250 Kbits		-100	
200 Kbits	PSDU length of 127 octets	-97	
400 Kbits		-90	
500 Kbits		-97	
1000 Kbits		-92	
On-Air Data rate	BPSK modulation	20 (at 868 MHz) 40 (at 915 MHz)	Kbps
	O-QPSK modulation	100 (at 868 MHz) 250 (at 915 MHz) and 784 MHz)	
TX Output/RX Input Nominal Impedance	For balanced output	100	Ω
Range, Outdoors	For balanced output	6	Km

Table 5.1: ATZB-900-B0 Sensors. RF Characteristics [16].

5.2 Measurement Campaign

First multilateration is used in a real scenario (outdoor) in order to evaluate its performance in practical scenarios when the driver is moving at a constant speed. The number of anchor nodes used to perform multilateration is three (trilateration). The results of multilateration in a real scenario composed by 6 reference nodes is shown in figures 5.6 and 5.7 for both low and high moving speeds. The reference nodes are shown in blue and the estimates positions in red. The x-dimension corresponds to the UTM Easting whereas the y-dimension corresponds to the UTM Northing. Both are referred in m from the Greenwich meridian and the from the equator.

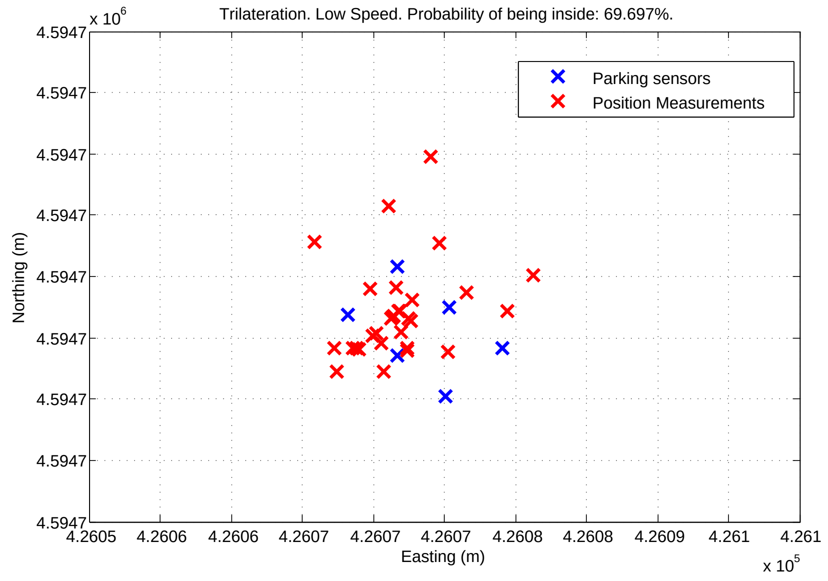


Figure 5.6: Triangulation-based position measurements at low speed

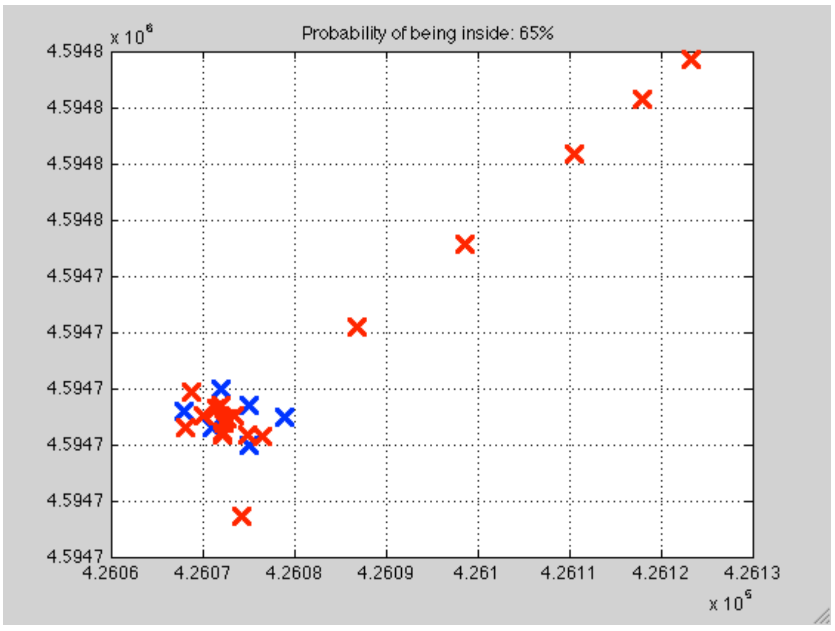
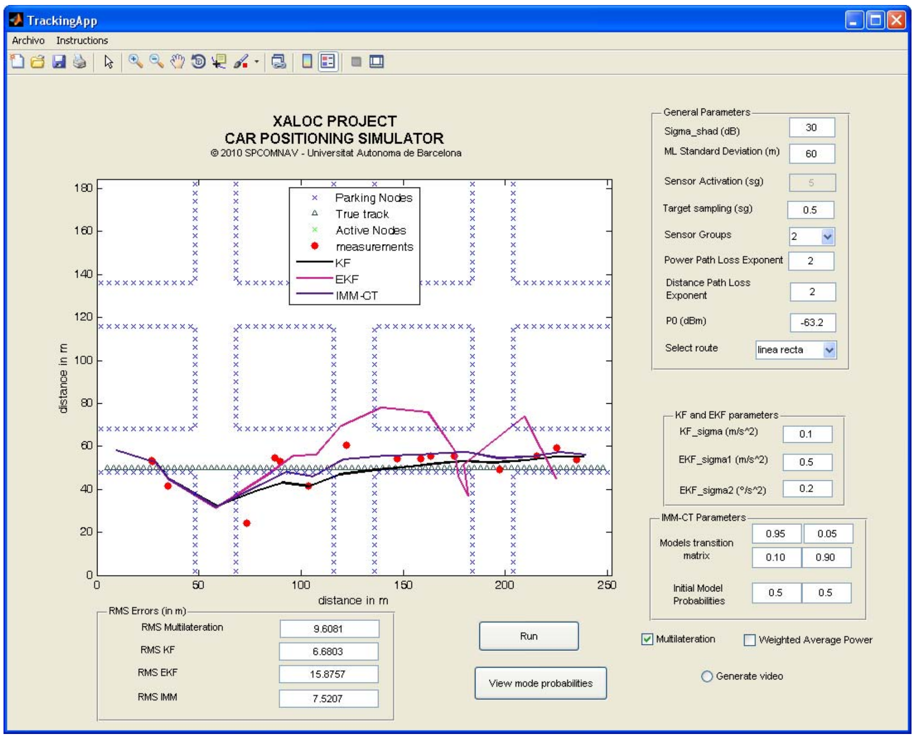


Figure 5.7: Triangulation-based position measurements at high speed

One can see above that the performance of multilateration in real scenarios is really bad giving large position errors as well as position estimates outside the measurement area. One reason of that is due to the multipath effect resulting in large variations on the RSSI and therefore giving bad position estimations. Another effect that can happen in wireless channels is the random temporal variation of the path loss exponent for each wireless link. The performance of multilateration when the noise power is high resulting with large position errors ($\sigma_{shad}^2 = 30\,dB, \sigma_z = 60\,m$) is demonstrated using the simulator. The results are shown in figure 5.8.



RMS Errors (in m)
RMS Multilateration 9.6081
RMS KF 6.6803
RMS EKF 15.8757
RMS IMM 7.5207

General Parameters
Sigma_shad (dB) 30
ML Standard Deviation (m) 60
Sensor Activation (sg) 5
Target sampling (sg) 0.5
Sensor Groups 2
Power Path Loss Exponent 2
Distance Path Loss Exponent 2
P0 (dBm) -63.2
Select route linea recta

KF and EKF parameters
KF_sigma (m/s^2) 0.1
EKF_sigma1 (m/s^2) 0.5
EKF_sigma2 (°/s^2) 0.2

IMM-CT Parameters
Models transition matrix 0.95 0.05 / 0.10 0.90
Initial Model Probabilities 0.5 0.5

[✓ Multilateration] Weighted Average Power
Generate video

Figure 5.8: Multilateration based positioning

It can be seen that some position estimates appear outside the road bounds specified by the anchors positions. Notice that the simulator does not consider all the effects that appear together in a real wireless channel. In fact higher position errors than 9 m appear in practice. For this reason another positioning technique is developed to have better position estimates. This developed method is called as WAPM(**Weighted Average Power Method**). Using WAPM the estimated positions appear always inside the scenario field. It is achieved by assigning weights for each of the coordinates of those anchors that participate in the positioning. These weights are proportional to the RSSI and averaged over all the RSSIs. Thus we can assure that the position of the driver will be allays inside the measurement area and close to that anchor with highest RSSI, i.e. highest weight.

Again the WAPM is formulated in (5.1) as in the section 4.1.

$$(\hat{x},\hat{y}) = \mathbf{A}\cdot\alpha$$
$$\alpha = \frac{\mathrm{RSSI}_n}{\sum_{i=1}^{N}\mathrm{RSSI}_i}, \quad n = 1,\cdots N_{anchors}, \tag{5.1}$$

where $\mathbf{A}_{2\times N}$ is a matrix containing the (x,y) coordinates of N selected reference nodes with the highest RSSI that participate in the positioning.

The performance of this method is demonstrated in practice as shown in figures 5.9 and 5.10 for both low and high speed, providing higher accuracy in the order of few meters. Furthermore the probability that all position estimates appear inside the area is higher than in multilateration.

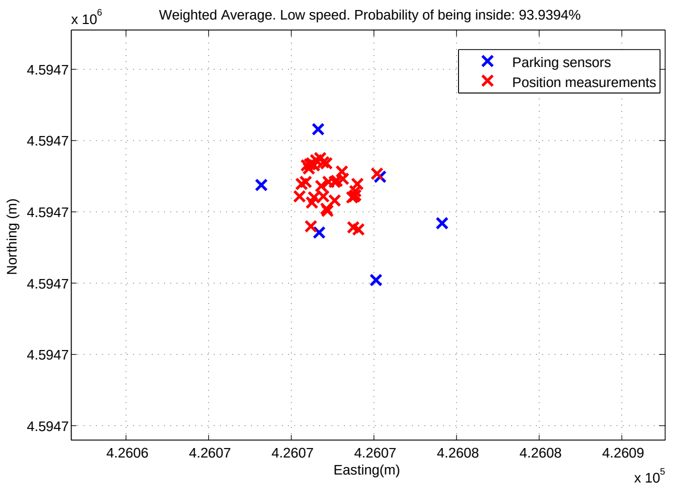


Figure 5.9: WAPM-based position measurements at low speed

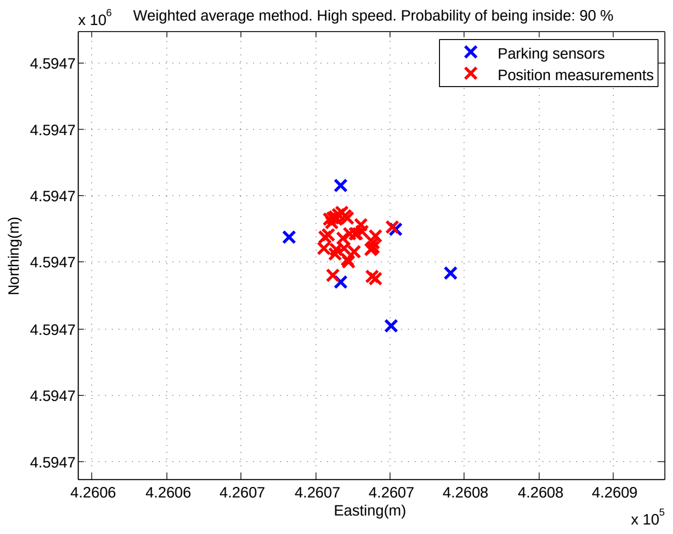
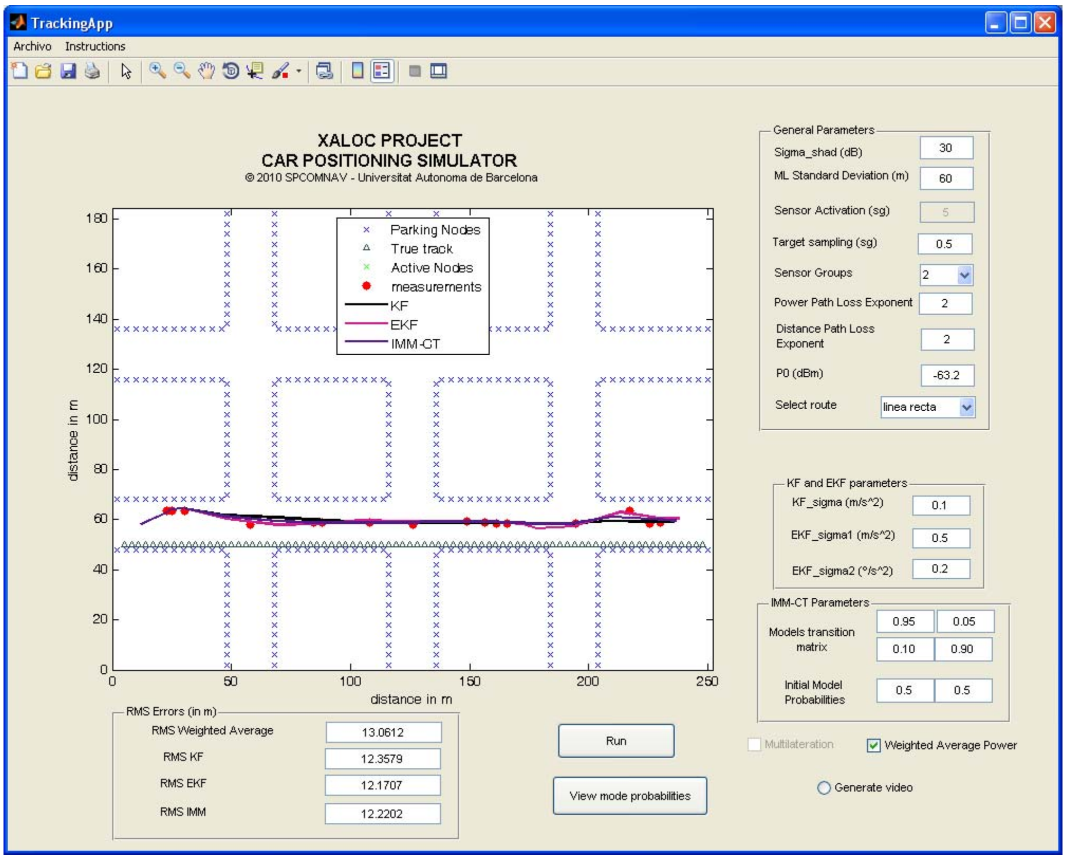


Figure 5.10: WAPM-based position measurements at high speed

A theoretical validation of WAPM can be demonstrated with the simulator. One can see in figure 5.11 that all the position estimates appear inside the road bounded by the parking sensors even for high fadings such as $\sigma^2_{shad} = 30\,dB$ and $\sigma^2_z = 60\,m$.



General Parameters

Sigma_shad (dB)	30
ML Standard Deviation (m)	60
Sensor Activation (sg)	5
Target sampling (sg)	0.5
Sensor Groups	2
Power Path Loss Exponent	2
Distance Path Loss Exponent	2
P0 (dBm)	-63.2
Select route	linea recta

KF and EKF parameters

KF_sigma (m/s^2)	0.1
EKF_sigma1 (m/s^2)	0.5
EKF_sigma2 (°/s^2)	0.2

IMM-CT Parameters

Models transition matrix	0.95	0.05
	0.10	0.90
Initial Model Probabilities	0.5	0.5

RMS Errors (in m)

RMS Weighted Average	13.0612
RMS KF	12.3579
RMS EKF	12.1707
RMS IMM	12.2202

☐ Multilateration ☑ Weighted Average Power

○ Generate video

Figure 5.11: Weighted Average Power based positioning with 4 reference nodes as in 5.8

Although WPAM guarantees that the user position is inside the measurement region, it does not guarantees the true user position. Due to high multipath interference (constructive/destructive) the position measure makes jumps forward and backward. Initially this jumps were very high in order between $10\,m - 50\,m$. In order to overcome this problem a one dimensional Kalman filter was introduced into the localization system. The reason to use one dimension instead of two dimensions as in chapter 4 is because the chosen scenario for the XALOC live demonstration is a straight road with no curves. Therefore the Kalman equations used for the navigator are the following:

State vector: The first element of the state vector is not the x-dimension because the target is not moving through the longitudinal plane of the map but is moving along a diagonal projection straight line since the scenario road is diagonal. Therefore the value d is the distance between an initial point (x_0, y_0) in the projection straight line and the estimated position of the driver.

$$\mathbf{x} = \begin{bmatrix} d \\ v \end{bmatrix} \tag{5.2}$$

The projection straight line contains the projected estimated positions. It is formulated in (5.3).

$$\begin{pmatrix} x_p \\ y_p \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \left(\frac{(b - a_1 x - a_2 y)}{a_1^2 + a_2^2} \right) \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}, \tag{5.3}$$

where

$$m = \frac{y_1 - y_0}{x_1 - x_0} \tag{5.4}$$
$$b = y_0 - x_0 m \tag{5.5}$$
$$a_1 = -m \tag{5.6}$$
$$a_2 = 1, \tag{5.7}$$

being (x_0, y_0) and (x_1, y_1) the coordinates of initial and final points that define the projected straight line, (x, y) the coordinates of the estimate and (x_p, y_p) the coordinates of the projected estimate. The motivation to apply the projection is the same as most of GPS navigators apply, i.e. the position of the driver is located in the middle of the road.

State equation:

$$\mathbf{x}(k+1) = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} \mathbf{v}(k) \quad n = 1, 2, \ldots \tag{5.8}$$

Observation equation:

$$\mathbf{z}(k+1) = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}(k+1) \tag{5.9}$$

As the model is one dimension the covariance of the process acceleration noise is also a unidimensional matrix: $\mathbf{Q}_{1\times1} = \sigma_a^2$. As well the covariance of the measurements is $\mathbf{R}_{1\times1} = \sigma_z^2$. The initialization of the process covariance matrix is found in section 3.3. The optimal values of the KF that have been found after a lot of campaign measurements are the ones shown in (5.10).

$$\sigma_a^2 = 3\,m/s^2 \tag{5.10}$$
$$\mathbf{R} = 25\,m \tag{5.11}$$
$$T = 2.5\,s \tag{5.12}$$

The interval of the measurements in the final demonstration is 2.5 s thus avoiding two groups of activated sensors.

5.3 Implementation of the navigator

The car carries the base node connected to the tablet netbook via USB interface that receives the broadcast messages from all parking sensors. The navigator implemented for vehicle localization and guidance includes the following components:

- User Terminal: netbook+sensor node+3G USB dongle.
- Application: JAVA programming language+tinyOS

A block diagram of the navigator implementation is shown in figure 5.12.

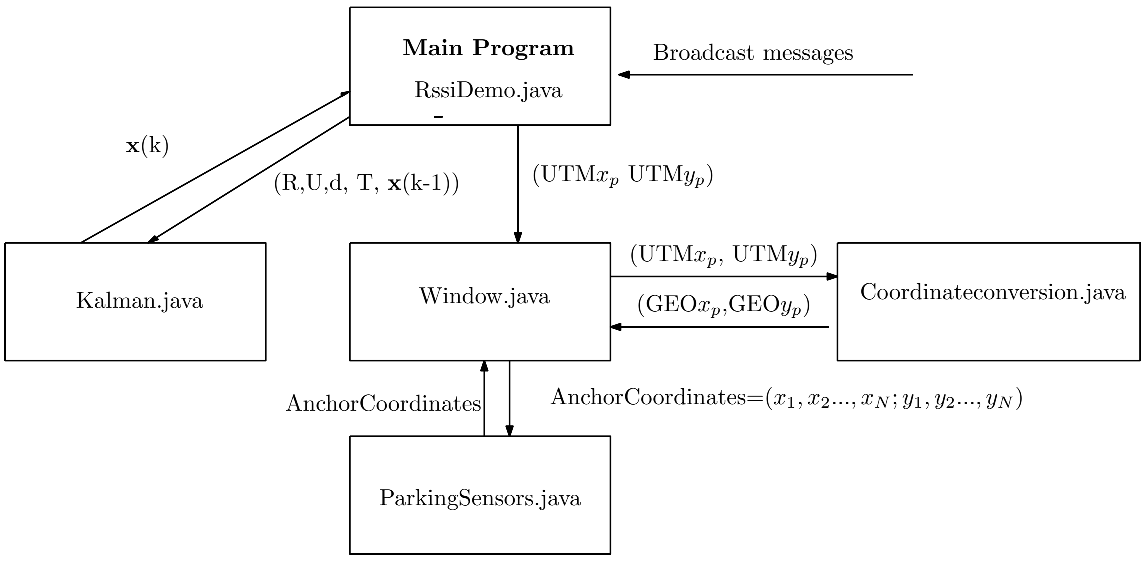


Figure 5.12: Navigator block diagram

The navigator is developed in Java using the Java IDE(*Integrated Development Environment*) Netbeans. Java is an object oriented language which is based on several interconnected classes and objects. An object is an instance of a class and a Java class stores information for several objects. An example: the student class stores objects with the following information: name, surname, address, age, student ID and course level.

Below is given a description of the set of interconnected classes used for the navigator implementation are:

- Rssidemo.java: this is the main class and the main program. This code is able to recollect the frames coming from the base sensor node via serial USB port. Then the received power source is read with the TinyOS function *msg.get_rssi()* from the received packet *msg*. This class uses TinyOS Java libraries provided to interact with the received packets from the WSN through the base node. TinyOS is a free and open source component-based operating system developed by the University of Berkeley. TinyOS is an embedded operating system optimized for low-power energy constrained devices and written in the nesC programming language as a set of cooperating tasks and processes interconnected. TinyOS is based in event-driven handles.

  The positioning computation as well projection is done in this code including the call to the Kalman filter. The projected computed position is send to the Window class.

- Window.java: this class is the application window of the navigator having all the swing Java objects (buttons, checkboxes, layeredpanes and so on). The origin of Window.java is the library jposition[60] developed as an application for handling maps using Google Maps from Java. The input of this application are the site coordinates in the GPS format. The output is a map downloaded from Google Maps server as well as a marker icon over the site coordinates.

  The navigator requirements were to show both the real-time user's position and the real-time parkings state with different markers (or icons). The problem we found with the jposition application is that the request to Google Maps can not be done every time the position is computed. It is because during the downloading time of the map the parkings states and/or the user's position can be changed since the driver is moving. Therefore the objective was to draw in the correct positions on map the parkings icons as well as the user's position in real-time. The map can be either downloaded from internet or loaded locally depending of which of both provided buttons are selected: cold start and warm start. If the cold start button is clicked then the map is downloaded from Google Maps centred at the first computed user's position. The advantage of the warm start button is that a connection to the Google Maps server is not needed. The user's position icon as well as the parkings icons are drawn using the jLayeredPane class (an available Java class) which allows to draw objects at different independent layers. At the layer 1 we have the icons showing the parkings such us their state with two icons: blue for not busy and red for busy. At the layer 2 we have the driver position icon shown in green.

- ParkingSensor.java: this class is used to create objects that store the UTM(*Universal Traverse Mercator*) coordinates of every parking sensor. In the UTM definition, the world is divided in zones along longitudinal plane and in letters along the latitudinal plane. The UTM coordinates have the following form:

$$zone \quad letter \quad Northing \quad Easting \tag{5.13}$$

  where Easting and Northing are expressed in m from the reference meridian Greenwich and from the equator. In Spain the UTM zone is 31 and the UTM letter T.

- KalmanFilter.java: this class implements the one dimension Kalman filter. As the kalman is of one dimension both the measurements covariance matrix and the process acceleration noise covariance matrix have dimensions 1×1 being thus the variance. The input to this class is: the measurement d, the measurements variance, the acceleration noise variance, the sampling time and the previous state vector.

- CoordinateConversion.java: this class is used to transform from geographical coordinates to UTM coordinates and vice-versa.

The navigator is called **ARID NAVIGATOR** and is shown in figure 5.13. We can see the base node connected on the left to the netbook.

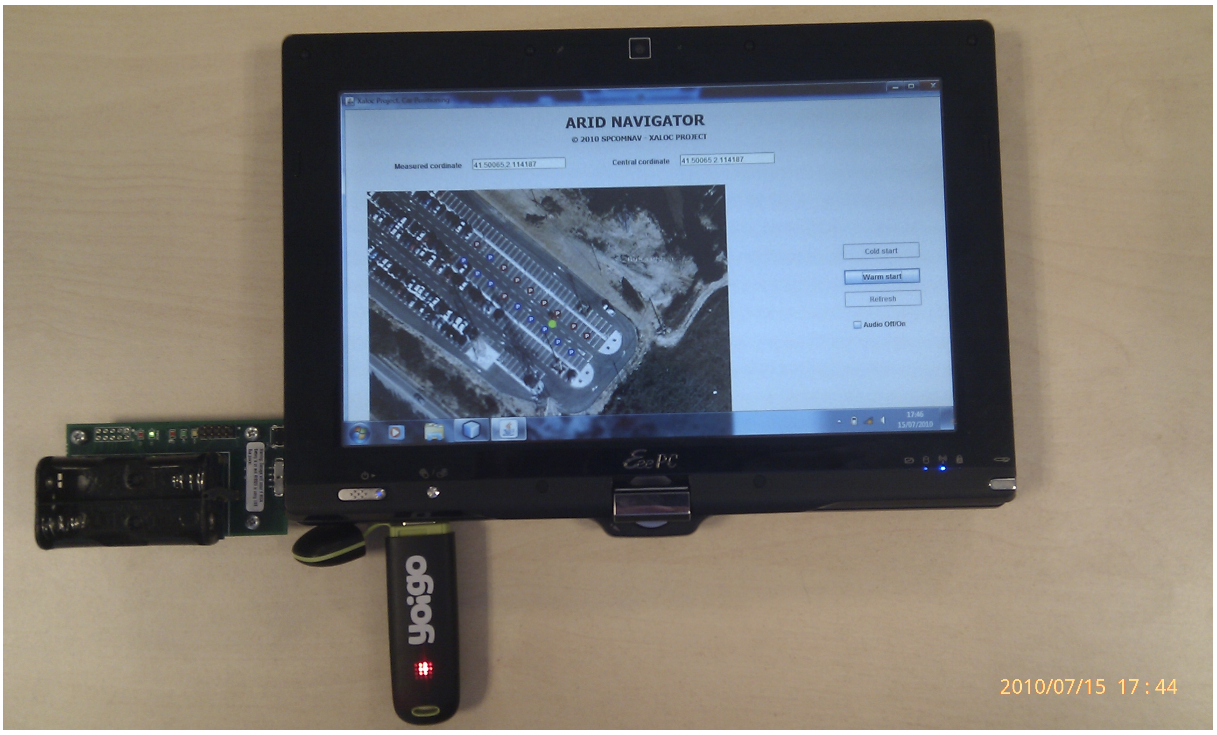
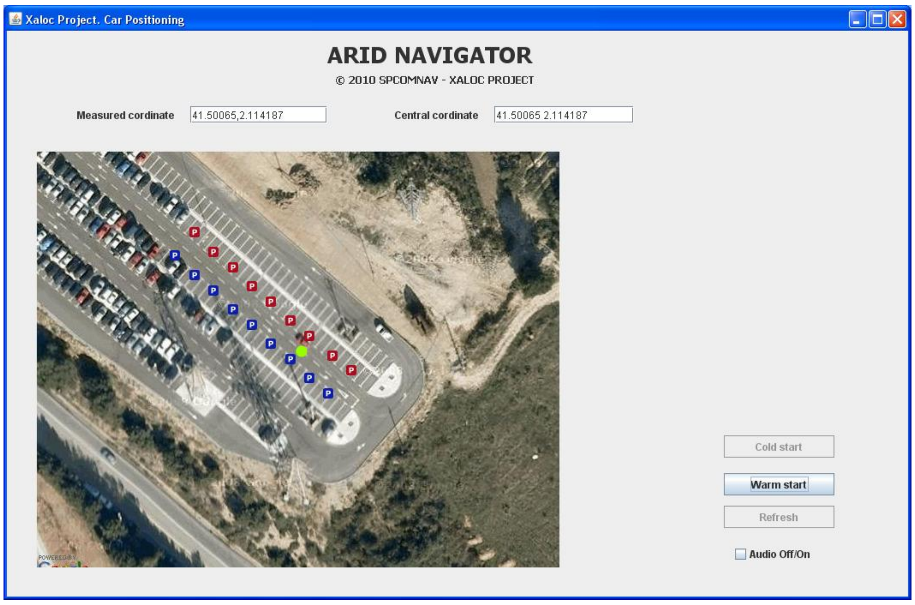


Figure 5.13: ARID Navigator based on Java on a tablet netbook.

The navigator Java application has the following elements:

1. Central coordinate: it gives the central GEO(*Geographical*) coordinate of the map.
2. Measured coordinate: it is the GEO coordinate obtained with trilateration taking the RSSI's of the received broadcasts messages. A coordinate transformation from UTM to GEO is taken into account.
3. Cold start button: it allows to download a map centred at first computed user's position.
4. Warm start button: this options allows to work without the need of downloading a map of the zone. It loads a locally stored map of the zone.
5. Refresh button: this option available only if cold start button is selected. This button refresh the map and downloads another map centred at the driver position.
6. Audio On/Off: if selected, an audio message appears every 10 s to announce the number of available parking slots.

The navigator shows the status of each parking slot with blue and red icons drawn over the map. It has been proved that when a car enters and leaves a certain parking slot, the parking icon on the map is updated with the corresponding one and also an audio message says the number of free parking slots. Figure 5.14 shows the Java application of the navigator:


Xaloc Project. Car Positioning
ARID NAVIGATOR
© 2010 SPCOMNAV - XALOC PROJECT
Cold start
Warm start
Refresh
Audio Off/On

Measured cordinate 41.50065,2.114187

Central cordinate 41.50065 2.114187

Figure 5.14: ARID Navigator

The green point seen on the map in figure 5.14 is the driver's position. It moves exactly in the middle between the parking sensors due to the projection.

5.4 Experimental Validation

The scenario is composed by 18 sensors nodes located in 18 parkings in diagonal(9 parking slots per site) in an area of dimensions $80 \times 70\,m^2$. Thus the experimental validation is realized through a diagonal projection straight line along the diagonal road. In order to demonstrate that the localization system works even with different sensors from other manufacturers several campaign-measurements are carried taking into account the following scenarios:

1. Estimation of the car position at several known positions: The car position estimates are obtained at several known positions along the road between the parking slots. It is not considered tracking.

2. Tracking of the car's position that is moving at a constant speed of $10Km/h$.

3. Tracking of the car's position that is moving at a constant speed of $20Km/h$.

For all the scenarios the following Kalman parameters:

$$\sigma_u^2 = 3\,m/s^2 \tag{5.14}$$

$$R = 10\,m \tag{5.15}$$

The results of the three scenarios are shown in figures 5.15, 5.16 and 5.17 respectively.

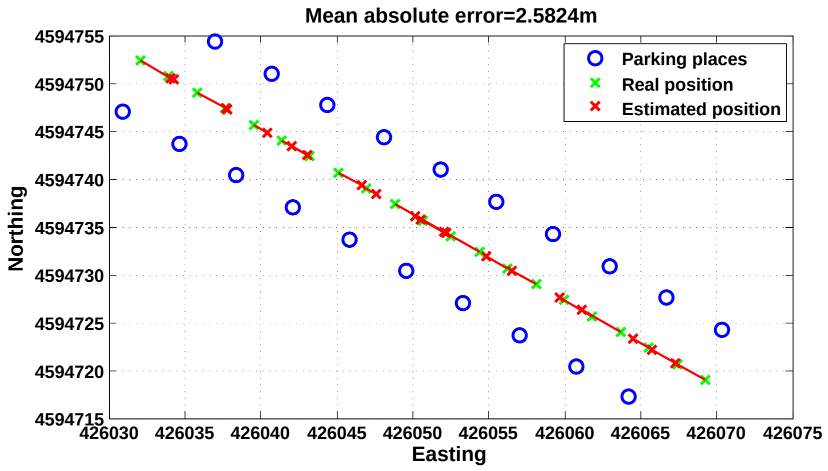
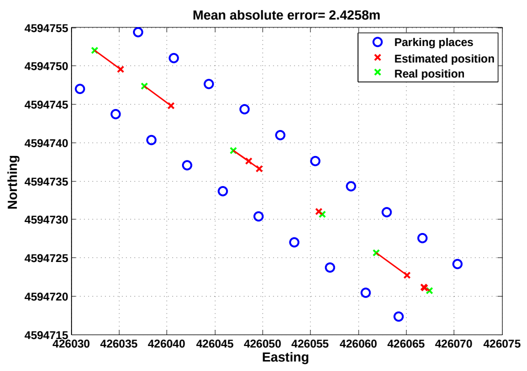


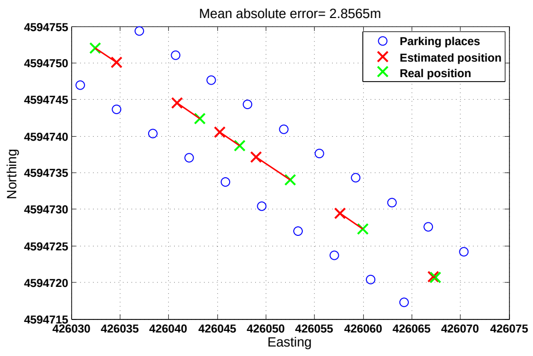
Figure 5.15: Scenario 1: One realization of the estimation of the car at several fixed locations

.

Figure 5.15 shows a mean average error of 2.5824 m. The result is considered valid because as a reference GPS achieves location precisions between $2.5 - 3\,m$.



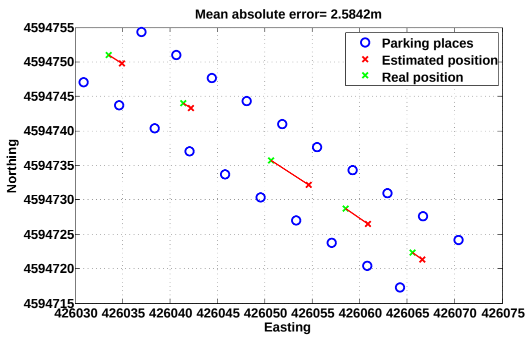
(a)



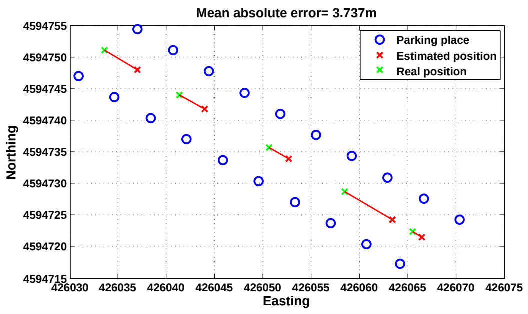
(b)

Figure 5.16: Scenario 2: Two realizations of the car moving at a constant speed of 10 Km/h

.



(a)



(b)

Figure 5.17: Scenario 3: Two realizations of the car moving at a constant speed of 20 Km/h

.

The different realizations in figure when the car moves at a constant speed of 10 Km/h gives a mean error of 2.6412 m whereas when the car moves a constant higher speed of 20 Km/h a mean error of 3.1606 m. Both are greater than the first scenario. Furthermore the error increases with the car speed which means that for speeds higher than 20 Km/h the position accuracy decreases achieving values greater than 3.1606 m. The parameter's navigator have been adjusted to allow the tracking at low considerable speed. Speeds around 20 Km/h are common when the driver is paying attention to the parking panels indications or the provided online navigator.

5.5 Summary

One the one hand, this chapter has shown a developed localization system based on a Wireless Sensor Network. First multilateration is used in order to validate its performance in a real scenario when the driver is moving at a constant speed. It is seen that the multilateration position estimates have large position errors higher thant $5\,m$ most of them appear outside the scenario region. Therefore a new positioning method called **Weighted Average Power Method** is found such that it concentrates all the position estimates always inside the scenario field. The last demonstrates position errors in the average of around $2-3\,m$ and thus it can be as an appropriate positioning technique for practical scenarios.

On the other hand a Java-based navigator called *ARID NAVIGATOR* has been developed to show to the driver its real-time estimated position over a map. In addition to the position the navigator informs the number of free parking slots either by audio or graphically showing their location on the map. The map can be downloaded first from Google Maps server or loaded locally depending on the selected button provided by the navigator. cold start or warm start. The navigator is validated and it was demonstrated during the live demonstration carried on July 7^{th} 2010.

Chapter 6

Conclusions and Future work

This master thesis is sponsored by the regional XALOC project in the framework of the INFOREGIÓ program (INFOREGIO/AJUTS 2009) funded by the Autonomous Government of Catalonia. The carried work has dealt with the theoretical validation as well as the experimental development of a centralized positioning and tracking in a Wireless Sensor Network.

Concerning the theoretical validation, it involves vehicle localization and tracking based on a novel tracking algorithm called IMM (**Interacting-Multiple-Model**) which uses one Kalman filter for uniform motion tracking and one Extended Kalman filter for the coordinated turns. The theoretical analysis demonstrates that the performance of IMM algorithm is better than having one KF and another EKF running in parallel independently because IMM compares the corresponding probabilities of the measurement evaluated in the probability density function of every model. In other words, IMM gives a mixed estimation based on the models probabilities.

The experimental development deals with the implementation of a Java-based navigator on a tablet netbook. The navigator informs the driver of its real-time position on the map and also the number of available parking slots. To validate the implemented navigator, a live demonstration using real sensors is carried out. These sensors send information messages of the parking state to a superNode that is connected with a database server. The experimental development demonstrates that multilateration technique using distances from each parking sensor's RSSI is not accurate when the target is moving. For that reason, a new positioning method called Weighted Average Power Method is proposed. Besides, a Kalman filter is adapted to track the driver.

The XALOC project has had a large impact in the in the press. During the demonstration day tens of producers came to record the live demonstration. The next day a lot news were published. Some of the recorded newspapers are found in the appendix D.

Future work will be based on distributed approaches such as:

- Distributed position
- Distributed tracking

Also a practical implementation of a distributed/centralized IMM algorithm in order to track the driver through the straight lines and through the turns.

Moreover, collaborative localization will be implemented which in WSN with a few number of reference nodes. Thus every unknown sensor will be able to obtain its coordinates from its neighbour's coordinates by means of a collaborative positioning algorithm.

Appendices

Appendix A

Scenario definition code

This appendix deals to show the involved code to create the street scenario as well as the target routes and the anchors positions. The set of matlab .m files shown here are the following:

- GeneralScenario.m : this is the main file. In order to generate the true track a vector called *order* contains the order to follow in each crossing which can be either straight on, turn left or turn right.
- AnchorsPosDefinition.m: this code is intended to create a matrix with positions of the anchors. The third row of this matrix is the ID of the street where those anchors belongs. The street ID's are numbered like a 3×3 (for vertical streets) matrix: $111213; 212223; 313233$ and like a 2×3 for the horizontal streets: $41, 42, 43; 51, 52, 53$.
- changeDirection.m: this code which is executed when the target is inside in a crossing and change the direction accordingly to the route vector and a direction vector. Route vector only has the following values: 1 for going straight on, 2 for turn left and 3 for turn right. The direction vector helps to indicate whether in which direction the target moves either in x or in y axis. It can have the following values: 1 whenever the target moves with positive velocity and -1 whenever the target moves with negative velocity.
- maneuver.m: this code performs nearly coordinated turn model inside the crossings when needed.
- controlTurnRight.m: This function tells with the variables *true* and *false* if the target is located at the border of the crossings to avoid that the target follows its turn right. The meaning of the variables "*true*" and "*false*" are the following:
  - true=1: If the target is located at the borders.
  - false=0:If the target is inside the crossing

Listing A.1: MATLAB code of GeneralScenario.m

```matlab
 clear all;
 %close all;
 format short;

%----------------------------STREET PARAMETERS-----------------------------
Anchors_Street = 24; %Number of total Anchors (12 per side)
distAnch = 4; %distance between anchors
WidthStreet=20;
start_x = 2; %this would be the distance from the reference where
%the parking lots begin
lengthStreet = 2*start_x +distAnch*((Anchors_Street/2)-1);
%--------------------------------------------------------------------------

%------------------------SCENARIO AREA -----------------------------------
block_horiz = 4; %total of horiz "block"
block_vert = 3;  %total of vert "block"
cr=(block_horiz-1)*(block_vert-1); %number of croisses
streets=cr;%Number of streets
Scenario_Width = block_horiz*lengthStreet+WidthStreet*(block_horiz-1);
Scenario_Height = block_vert*lengthStreet+WidthStreet*(block_vert-1);
%-------------------------------------------------------------------------

% %--------------------DEFINE THE POSITIONS OF THE ANCHORS -----------------
% %We associate the set of anchors of one street to a number that identifies
% %this street.
%
 AnchPos=AnchorsPosDefinition(Anchors_Street,block_vert,block_horiz,...
     lengthStreet,distAnch,WidthStreet,start_x);

%---------------------DEFINE THE CROSSINGS IN MATRICES--------------------
%As we have 6 crs in a 4x3 blocks:
cr1=[lengthStreet lengthStreet+WidthStreet;...
    2*(lengthStreet+WidthStreet) 2*lengthStreet+WidthStreet];
cr2=[2*lengthStreet+WidthStreet 2*(lengthStreet+WidthStreet);...
    2*(lengthStreet+WidthStreet) 2*lengthStreet+WidthStreet];
cr3=[3*lengthStreet+2*WidthStreet 3*(lengthStreet+WidthStreet);...
    2*(lengthStreet+WidthStreet) 2*lengthStreet+WidthStreet];

cr4=[lengthStreet lengthStreet+WidthStreet;...
    lengthStreet+WidthStreet lengthStreet];
cr5=[2*lengthStreet+WidthStreet 2*(lengthStreet+WidthStreet);...
    lengthStreet+WidthStreet lengthStreet];
cr6=[3*lengthStreet+2*WidthStreet 3*(lengthStreet+WidthStreet);...
    lengthStreet+WidthStreet lengthStreet];

%--------------------------TARGET PARAMETERS------------------------------
```

```
%Now let's start assuming that a mobile node M travels along a certain path
%with constant velocity. The following parameters are required:

T_target = 0.01;%sampling time in seconds of the target. Every T_target,
%the target has moved v*T_target m.
T_motes=5; %sampling time in seconds of the active motes.
%Every T_motes sg, all active anchors send a message to the target.
anchors_ML = 4; %number of considered anchors for multilateration computation
order=[1,1,1]; %1: go ahead, 2: turn left, 3:turn right
path=[51,52,53,54];
direction=[1;0]; %this variable helps to indicate whether in which
%direction the target moves either in x or in y axis:
%1: the target moves with positive velocity
%-1: the target moves with negative velocity
c=1; %to check the order vector
v=20*1000/3600; %velocity of the target in m/s
distFromAnchors =2; %Distance of the target with respect to the anchors
target_state=[2;v;50;0;path(1)]; %Initial target_state.
%It is in the form of [x,vx,y,vy,streetID].The last row identifies the target
%to that street where it is located
r1=WidthStreet-distFromAnchors;
r2=distFromAnchors;
yaw_rate_left=v/r1; %turning rate in rad/sg. At the beginning we assume
%a constant turning rate
yaw_rate_right=-v/r2; %turning rate in rad/sg. At the beginning we assume
%a constant turning rate
dt_right=(pi/2)*r2/v; %Need time to turn (pi/2) driving at a constant velocity v

%transition matrix
F=[1 T_target  0 0;0 1 0 0;0 0 1 T_target ;0 0 0 1];

%---------------------Next Define the turning matrix ---------------------
%We will have two turning matrices: one to turn left and the other to turn
%right
%--------Turn left matrix---------
coswt_left = cos(yaw_rate_left*T_target );
coswto_left = cos(yaw_rate_left*T_target )-1;
coswtopw_left = coswto_left/yaw_rate_left;
sinwt_left = sin(yaw_rate_left*T_target );
sinwtpw_left = sinwt_left/yaw_rate_left;

Turning_left =   [1 sinwtpw_left   0 coswtopw_left;...
                   0 coswt_left     0 -sinwt_left  ;...
                   0 -coswtopw_left 1 sinwtpw_left ;...
                   0 sinwt_left     0 coswt_left   ];

% %--------Turn right matrix---------
```

```
coswt_right = cos(yaw_rate_right*dt_right);
coswto_right = cos(yaw_rate_right*dt_right)-1;
coswtopw_right = coswto_right/yaw_rate_right;
sinwt_right = sin(yaw_rate_right*dt_right);
sinwtpw_right = sinwt_right/yaw_rate_right;


 Turning_right =   [1 sinwtpw_right   0 coswtopw_right;...
                    0 coswt_right     0 -sinwt_right  ;...
                    0 -coswtopw_right 1 sinwtpw_right ;...
                    0 sinwt_right     0 coswt_right   ];
%-------------------------PATH LOSS MODEL PARAMETERS------------------------
sigma_shad = 2; %shadow fading variance in dB
P0 = -63.2; %Received power at 1 m used for the received power model in dBm
gamma=2;
%------------------------------------------------------------------------
x=1; %Target Samples
ml=1; %Target estimated samples
count=0;
%----------------------------BEGIN THE TRACKING ---------------------------
while 0≤target_state(1,x) && target_state(1,x)≤ max(AnchPos(1,:))...
        && 0≤target_state(3,x) && target_state(3,x)≤ max(AnchPos(2,:))
     count=count+T_target;
     count=roundn(count);
    %Now all the received powers of all the anchors are gathered by the
    %target. We use the simple path loss model:
    %Pr(dBm)=P0(dBm)-10·gamma·log10(d_i), where d_i is the euclidean
    %distance from the mobile target to anchor i

    %Below we check if the target is inside a cr

   if ((cr1(1,1)≤target_state(1,x) && target_state(1,x)≤cr1(1,2)) && ...
 (cr1(2,2)≤target_state(3,x) && target_state(3,x)≤cr1(2,1)) || ...
   (cr2(1,1)≤target_state(1,x) && target_state(1,x)≤cr2(1,2)) && ...
 (cr2(2,2)≤target_state(3,x) && target_state(3,x)≤cr2(2,1))  || ...
   (cr3(1,1)≤target_state(1,x) && target_state(1,x)≤cr3(1,2)) && ...
 (cr3(2,2)≤target_state(3,x) && target_state(3,x)≤cr3(2,1))  || ...
   (cr4(1,1)≤target_state(1,x) && target_state(1,x)≤cr4(1,2)) && ...
 (cr4(2,2)≤target_state(3,x) && target_state(3,x)≤cr4(2,1))  || ...
   (cr5(1,1)≤target_state(1,x) && target_state(1,x)≤cr5(1,2)) && ...
 (cr5(2,2)≤target_state(3,x) && target_state(3,x)≤cr5(2,1))  || ...
  (cr6(1,1)≤target_state(1,x) && target_state(1,x)≤cr6(1,2)) && ...
 (cr6(2,2)≤target_state(3,x) && target_state(3,x)≤cr6(2,1)))

%Next we need to check wether in which cr is located the target
%-----------------if the target is cr the cr 1 ---------------
  if ((cr1(1,1)≤target_state(1,x)) && (target_state(1,x)≤cr1(1,2)) && ...
 (cr1(2,2)≤target_state(3,x)) && (target_state(3,x)≤cr1(2,1)))
 [true,false,dir,target,x1]=controlTurnRight(direction,v,cr1,T_target,...
     target_state,x,distFromAnchors);
 if (true==0 && false==1)
    [y,x]=maneuver(F,order,c,Turning_left,Turning_right,target_state,x);
    target_state(1:4,x)=y;
 else
    direction=dir;
    target_state=target;
    x=x1;
  end
   end

%----------------------if the target is cr the cr 2-------
   if ((cr2(1,1)≤target_state(1,x) && target_state(1,x)≤cr2(1,2)) && ...
 (cr2(2,2)≤target_state(3,x) && target_state(3,x)≤cr2(2,1)))
 [true,false,dir,target,x1]=controlTurnRight(direction,v,cr2,T_target,...
     target_state,x,distFromAnchors);
 if (true==0 && false==1)
   [y,x]=maneuver(F,order,c,Turning_left,Turning_right,target_state,x);
    target_state(1:4,x)=y;
  else
    direction=dir;
    target_state=target;
    x=x1;
 end
  end

 %----------------------if the target is cr the cr 3----------
   if ((cr3(1,1)≤target_state(1,x) && target_state(1,x)≤cr3(1,2)) && ...
 (cr3(2,2)≤target_state(3,x) && target_state(3,x)≤cr3(2,1)))
 [true,false,dir,target,x1]=controlTurnRight(direction,v,cr3,T_target,...
     target_state,x,distFromAnchors);
if (true==0 && false==1)
   [y,x]=maneuver(F,order,c,Turning_left,Turning_right,target_state,x);
    target_state(1:4,x)=y;
else
    direction=dir;
    target_state=target;
    x=x1;
end
   end
 %----------------------if the target is cr the cr 4-------
 if ((cr4(1,1)≤target_state(1,x) && target_state(1,x)≤cr4(1,2)) && ...
 (cr4(2,2)≤target_state(3,x) && target_state(3,x)≤cr4(2,1)))
 [true,false,dir,target,x1]=controlTurnRight(direction,v,cr4,T_target,...
     target_state,x,distFromAnchors);
```

```
if (true==0 && false==1)
   [y,x]=maneuver(F,order,c,Turning_left,Turning_right,target_state,x);
    target_state(1:4,x)=y;
else
    direction=dir;
    target_state=target;
    x=x1;
end
end
%----------------------if the target is cr the cr 5-----------
    if ((cr5(1,1)≤target_state(1,x) && target_state(1,x)≤cr5(1,2)) && ...
 (cr5(2,2)≤target_state(3,x) && target_state(3,x)≤cr5(2,1)))
 [true,false,dir,target,x1]=controlTurnRight(direction,v,cr5,T_target,...
     target_state,x,distFromAnchors);
if (true==0 && false==1)
   [y,x]=maneuver(F,order,c,Turning_left,Turning_right,target_state,x);
    target_state(1:4,x)=y;
else
 direction=dir;
 target_state=target;
 x=x1;
end
    end
%----------------------if the target is cr the cr 6-------

 if ((cr6(1,1)≤target_state(1,x) && target_state(1,x)≤cr6(1,2)) && ...
 (cr6(2,2)≤target_state(3,x) && target_state(3,x)≤cr6(2,1)))
 [true,false,dir,target,x1]=controlTurnRight(direction,v,cr6,T_target,...
     target_state,x,distFromAnchors);
if (true==0 && false==1)
   [y,x]=maneuver(F,order,c,Turning_left,Turning_right,target_state,x);
    target_state(1:4,x)=y;
 else
 direction=dir;
 target_state=target;
 x=x1;
end
end

  else %If the target is not in a cr

    %Before it must check if the target has crossed one of the possible
    %crs. We must do it for each cr.

%-------------------%If target has crossed cr 1---------------------
if x>1
 if ((cr1(1,1)≤target_state(1,x-1))&&(target_state(1,x-1)≤cr1(1,2))&&...
 (cr1(2,2)≤target_state(3,x-1)) && (target_state(3,x-1)≤cr1(2,1)))
   [target_state,direction,c]=changeDirection(direction,order,c,v,cr1,...
       distFromAnchors,start_x,target_state,x);
  end
 %-------------------%If target has crossed cr 2---------------------

if ((cr2(1,1)≤target_state(1,x-1) && target_state(1,x-1)≤cr2(1,2)) && ...
 (cr2(2,2)≤target_state(3,x-1) && target_state(3,x-1)≤cr2(2,1)))
   [target_state,direction,c]=changeDirection(direction,order,c,v,cr2,...
       distFromAnchors,start_x,target_state,x);
end

 %-------------------%If target has crossed cr 3---------------------
  if ((cr3(1,1)≤target_state(1,x-1) && target_state(1,x-1)≤cr3(1,2)) && ...
 (cr3(2,2)≤target_state(3,x-1) && target_state(3,x-1)≤cr3(2,1)))
   [target_state,direction,c]=changeDirection(direction,order,c,v,cr3,...
       distFromAnchors,start_x,target_state,x);
  end
  %-------------------%If target has crossed cr 4---------------------
 if ((cr4(1,1)≤target_state(1,x-1) && target_state(1,x-1)≤cr4(1,2)) && ...
 (cr4(2,2)≤target_state(3,x-1) && target_state(3,x-1)≤cr4(2,1)))
   [target_state,direction,c]=changeDirection(direction,order,c,v,cr4,...
       distFromAnchors,start_x,target_state,x);
 end

   %-------------------%If target has crossed cr 5---------------------
if ((cr5(1,1)≤target_state(1,x-1) && target_state(1,x-1)≤cr5(1,2)) && ...
 (cr5(2,2)≤target_state(3,x-1) && target_state(3,x-1)≤cr5(2,1)))
   [target_state,direction,c]=changeDirection(direction,order,c,v,cr5,...
       distFromAnchors,start_x,target_state,x);
end
%-------------------%If target has crossed cr 6---------------------
if ((cr6(1,1)≤target_state(1,x-1) && target_state(1,x-1)≤cr6(1,2)) && ...
 (cr6(2,2)≤target_state(3,x-1) && target_state(3,x-1)≤cr6(2,1)))
   [target_state,direction,c]=changeDirection(direction,order,c,v,cr6,...
       distFromAnchors,start_x,target_state,x);
end
 end
     target_state(5,x)=path(c);
     x=x+1;

     target_state(1:4,x)=F*target_state(1:4,x-1);
   end
   target_state=roundn(target_state);
end
save linea_recta.mat target_state
```

```
figure(1), plot(AnchPos(1,1:length(AnchPos)),AnchPos(2,1:length(AnchPos)),...
    'x','markersize',5);
hold on
axis([0 4*lengthStreet+3*WidthStreet 0 3*lengthStreet+2*WidthStreet]);
plot(target_state(1,:),target_state(3,:),'b','markersize',5);
legend('Reference Nodes','real track', ...
    'estimated track with multilateration','Location','NorthEast');

%-------TO MAKE A MPG MOVIE: is a set of image frames -------
%   nframes=length(target_state)-1;
%  M=moviein(nframes);
%  for it=1:nframes
% figure(2), plot(AnchPos(1,1:length(AnchPos)),...
%AnchPos(2,1:length(AnchPos)),'x','markersize',5);
%  axis([0 4*lengthStreet+3*WidthStreet 0 3*lengthStreet+2*WidthStreet]);
%  hold on
%  plot(target_state(1,it),target_state(3,it),'b*','markersize',5);
%   plot( estimated_pos_target(1,it), estimated_pos_target(2,it),'r*',...
%'markersize',10);
%   if it>1
%  plot(target_state(1,1:it-1),target_state(3,1:it-1),'*','Color',...
%[135;206;250]/255,'markersize',5);
%   plot( estimated_pos_target(1,1:it-1), estimated_pos_target(2,1:it-1),...
%'*','Color',[255;192;203]/255,'markersize',10);
%  end
%  legend('Reference Nodes','real track',...
%'estimated track with multilateration','Location','NorthEast');
%  M(:,it)=getframe;
%  close all;
%  end
%  movie(M,1);
%  save trackingML2_withoutnoise.mat M
%convert the movie to mpeg format to play the mpeg file: unix('trackingML.mpg')
% mpgwrite(M,jet,'trackingML2_withoutnoise.mpg');
```

Listing A.2: MATLAB code of AnchorsPosDefinition.m

```
%--------------------DEFINE THE POSITIONS OF THE ANCHORS -----------------
%We associate the set of anchors of one street to a number that identifies
%this street. The function returns a matrix with
%[x,y,streetID,gamma_near,gamma_far,sigma_shad_near,sigma_shad_far]

function [AnchPos]=AnchorsPosDefinition(Anchors_Street,block_vert,block_horiz,...
    lengthStreet,distAnch,WidthStreet,start_x)

Total_Anchors=Anchors_Street*(block_vert-1)*(block_horiz)...
```

```
    +Anchors_Street*(block_horiz-1)*(block_vert);
AnchPos = zeros(3,Total_Anchors); %Anchors Positions.
%AnchPos(1:2,j) = (x,y) coordinates for the anchor j

N=Anchors_Street;
a=1;
z=a;
j=2;
q=11;
s=(lengthStreet*block_vert+WidthStreet*(block_vert-1))-start_x;
pattern=s:-distAnch:s-(distAnch*(Anchors_Street/2)-1);
A=pattern'*ones(1,block_vert);
A=A(:);
B=((WidthStreet+lengthStreet)*(0:block_vert-1))'*ones(1,Anchors_Street/2);
B=B';
B=B(:);
y=A-B; %This vector contains the y coordinates of all anchors nodes placed
%in vertical streets.

%Below positions of those anchor nodes that are placed along the vert
%streets are filled
for i=1:block_horiz-1
  %For x coordinates of the left side nodes
 x_left=lengthStreet*i+WidthStreet*(i-1);
 %For x coordinates of the right side nodes
 x_right=lengthStreet*i+WidthStreet*i;
AnchPos(1,a:2:(a-1)+N*block_vert)= x_left;
AnchPos(1,j:2:(a-1)+N*block_vert)= x_right;

%For y coordinates of the left side nodes
AnchPos(2,a:2:(a-1)+N*block_vert)= y;
%For y coordinates of the right side nodes
AnchPos(2,j:2:(a-1)+N*block_vert)= y;

for n=1:block_vert
AnchPos(3,z:(z-1)+N)=q+(n-1);
z=z+N;
end
a=a+N*block_vert;
z=a;
j=a+1;
q=11;
q=q+10*i;
end

m=block_vert-1;
j=a+1;
```

```
n=1;
q=41;
k=0;
%Below positions of those anchor nodes that are placed along the horiz
%streets are filled
pattern=start_x:distAnch:start_x+(distAnch*(Anchors_Street/2)-1);
A=pattern'*ones(1,block_horiz);
A=A(:);
B=((WidthStreet+lengthStreet)*(0:block_horiz-1))'*ones(1,Anchors_Street/2);
B=B';
B=B(:);
x=A+B; %This vector contains the x coordinates of all anchors nodes placed
%in horizontal streets.

for p=1:block_vert-1
y_left=lengthStreet*m+WidthStreet*m;
y_right=lengthStreet*m+WidthStreet*(m-1);
%For y coordinates of the left side nodes
AnchPos(2,a:2:(a-1)+N*block_horiz)= y_left;
%For y coordinates of the right side nodes
AnchPos(2,j:2:(a-1)+N*block_horiz)= y_right;

%For x coordinates of the left side nodes
AnchPos(1,a:2:(a-1)+N*block_horiz)= x;
%For x coordinates of the right side nodes
AnchPos(1,j:2:(a-1)+N*block_horiz)= x;

for n=1:block_horiz
AnchPos(3,z:(z-1)+N)=q+(n-1);
z=z+N;
end
a=a+N*block_horiz;
m=m-1;
n=n+1;
j=a+1;
q=41;
q=q+10*p;
end

%-------------------------END OF ANCHOR POSITIONS-------------------------
```

Listing A.3: MATLAB code of changeDirection.m

```
function [target_state,direction,c]=changeDirection(dir,ord,c1,v,targetState,x)
target_state=targetState;
direction=dir;
```

```
if dir(1,1) ==1 && ord(c1)==2 %if target turn left it changes the direction
    direction(2,1)=1;
    direction(1,1)=0;
    target_state(2,x)=0;
    target_state(4,x)=v;
 elseif  dir(1,1)==-1 && ord(c1)==2
    direction(2,1)=-1;
    direction(1,1)=0;
    target_state(2,x)=0;
    target_state(4,x)=-v;
 elseif dir(1,1)==1 && ord(c1)==3
    direction(2,1)=-1;
    direction(1,1)=0;
    target_state(2,x)=0;
    target_state(4,x)=-v;
 elseif dir(1,1)== -1 && ord(c1)==3
    direction(2,1)=1;
    direction(1,1)=0;
    target_state(2,x)=0;
    target_state(4,x)=v;
 elseif dir(2,1)==1 && ord(c1)==2
    direction(1,1)=-1;
    direction(2,1)=0;
    target_state(2,x)=-v;
    target_state(4,x)=0;
 elseif dir(2,1)==-1 && ord(c1)==2
    direction(1,1)=1;
    direction(2,1)=0;
    target_state(2,x)=v;
    target_state(4,x)=0;
 elseif dir(2,1)== 1 && ord(c1)==3
    direction(1,1)=1;
    direction(2,1)=0;
    target_state(2,x)=v;
    target_state(4,x)=0;
 elseif dir(2,1)==-1 && ord(c1)==3
    direction(1,1)=-1;
    direction(2,1)=0;
    target_state(2,x)=-v;
    target_state(4,x)=0;
end
   c=c1+1;
```

Listing A.4: MATLAB code of maneuver.m

```
function [y,x]=maneuver(F,order,c,Turning_left,Turning_right,target_state,xin)
```

```
y=target_state;
    if order(c)==1 %straight on
        x=xin+1;
        y(1:4,x)=F*target_state(1:4,x-1);
       % target_state(5,x)=path(c);
     elseif order(c)==2 %turn left
         x=xin+1;
        y(1:4,x)= Turning_left* target_state(1:4,x-1);

    else %turn right
         x=xin+1;
         y(1:4,x)= Turning_right* target_state(1:4,x-1);
    end
   y=y(1:4,x);
```

Listing A.5: MATLAB code of controlTurnRight.m

```
 function [true,false,dir,targetState,x1]=controlTurnRight(direction,v,...
     cr,T_target,target_state,x,distFromAnchors)
 targetState=target_state;
  dir=direction;
 true=0;
 false=1;
 x1=x;
v=v*1000/3600;
%This function tells if the target is located at the border of the
%crs to avoid that the target follows its turn right.
%true=1: If the target is located at the borders
%false=0:If the target is inside the cr

if (direction(1,1)==1 && direction(2,1)==0 && ...
        ((target_state(3,x)==cr(2,2)) ||...
        ((cr(2,2)≤target_state(3,x)) && (target_state(3,x)≤cr(2,2)+1))))
    x1=x+1;
    targetState(1:4,x1)=[cr(1,1)+distFromAnchors;0;...
        target_state(3,x)-v*T_target;-v];
    true=1;
    false=0;
elseif (direction(1,1)==-1 && direction(2,1)==0 && ...
        ((target_state(3,x)==cr(2,1)) ||...
        ((cr(2,1)-1≤target_state(3,x)) && (target_state(3,x)≤cr(2,1)))))
     x1=x+1;
    targetState(1:4,x1)=[cr(1,2)-distFromAnchors;0;cr(2,1)+v*T_target;v];
    true=1;
    false=0;
elseif (direction(1,1)==0 && direction(2,1)==1 &&...
        ((target_state(1,x)==cr(1,2)) ||...
        ((cr(1,2)-1<=target_state(1,x)) && (target_state(1,x)<=cr(1,2)))))
    x1=x+1;
    targetState(1:4,x1)=[cr(1,2)+v*T_target;0;cr(2,2)+distFromAnchors;-v];
    true=1;
    false=0;
elseif (direction(1,1)==0 && direction(2,1)==-1 && ...
        ((target_state(1,x)==cr(1,1)) ||...
        ((cr(1,1)<=target_state(1,x)) && (target_state(1,x)<=cr(1,1)+1))))
    x1=x+1;
    targetState(1:4,x1)=[cr(1,1)-v*T_target;0;cr(2,1)-distFromAnchors;-v];
    true=1;
    false=0;
end
```

Appendix B

Matlab GUI Code

This appendix deals to show the involved code to create the tracking simulator. The set of files "*.m" are the following:

- TrackingApp.m: This is the Matlab GUI tracking application.
- TrackingIMMKFandEKF.m: This code implemments the simulator, that is, when user click over the button run that appears in the application. This function calls all the tracking estimators.
- Multilateration.m: This code performs multilateration with those four active anchor nodes with maximum received power or with those anchors located at the crossing borders. If the option weighted average is specified in the application, instead of doing multilateration, the weighted average method is used.
- weightedAverage.m: This code generates the coefficients used in the weighted average method to calculate the target position.
- posfun.m: It is the cost function $C(x, y)$ appeared in 2.9. This cost function is the euclidean distance from the target to each of the anchors. The function fminunc from matlab optimization toolbox finds the values (x, y) that minimize $C(x, y)$.
- Kalman.m: It implements the Kalman Filter.
- genKalmancoefskf.m: This code computes the kalman coefficients from the state covariance, measurement covariance, transition matrix, observation matrix, sampling time and acceleration noise variance.
- ExtendedKalman.m: It implements the Extended Kalman Filter

- genKalmancoefsekf.m: This code computes the kalman coefficients from the state covariance, measurement covariance, transition matrix, observation matrix, sampling time and acceleration noise variance.
- chooseTransitionMatrix.m: This function choose the transition matrix depending on the target behaviour (if it is turning on or it is going straight on).
- immct: This code implements the IMM estimator.It requires the functions Kalman.m, ExtendedKalman.m and chooseTransitionMatrix.m.

Listing B.1: MATLAB code of TrackingApp.m

```
function varargout = TrackingApp(varargin)

% TRACKINGAPP M-file for TrackingApp.fig
%      TRACKINGAPP, by itself, creates a new TRACKINGAPP or raises the existing
%      singleton*.
%
%      H = TRACKINGAPP returns the handle to a new TRACKINGAPP or the handle to
%      the existing singleton*.
%
%      TRACKINGAPP('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in TRACKINGAPP.M with the given input arguments.
%
%      TRACKINGAPP('Property','Value',...) creates a new TRACKINGAPP or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before TrackingApp_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to TrackingApp_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help TrackingApp

% Last Modified by GUIDE v2.5 09-Jul-2010 11:16:45

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @TrackingApp_OpeningFcn, ...
                   'gui_OutputFcn',  @TrackingApp_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
```

```
                    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
muProb=[];

% --- Executes just before TrackingApp is made visible.
function TrackingApp_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to TrackingApp (see VARARGIN)

% Choose default command line output for TrackingApp
cla(handles.axes1,'reset');
set(handles.edit23,'String','');
set(handles.edit24,'String','');
set(handles.edit25,'String','');
set(handles.edit26,'String','');
set(handles.edit23,'enable','off');
set(handles.edit24,'enable','off');
set(handles.edit25,'enable','off');
set(handles.edit26,'enable','off');
set(handles.Multilateration,'Value',1);
set(handles.radiobutton1,'Value',0);
handles.output = hObject;

set(handles.pushbutton2,'enable','off');
set(hObject,'toolbar','figure');


% Update handles structure
guidata(hObject, handles);

% UIWAIT makes TrackingApp wait for user response (see UIRESUME)
% uiwait(handles.figure1);

%----------------------------STREET PARAMETERS-----------------------------
Anchors_Street = 24; %Number of total Anchors (12 per side)
```

```
distAnch = 4; %distance between anchors
WidthStreet=20;
start_x = 2; %this would be the distance from the reference where
%the parking lots begin
lengthStreet = 2*start_x +distAnch*((Anchors_Street/2)-1);
%-----------------------------------------------------------------------
%------------------------SCENARIO AREA -----------------------------------
block_horiz = 4; %total of horiz "block"
block_vert = 3;  %total of vert "block"
%----------------------------------------------------------------------
%-------------------DEFINE THE POSITIONS OF THE ANCHORS -----------------

 AnchPos=AnchorsPosDefinition(Anchors_Street,block_vert,block_horiz,...
     lengthStreet,distAnch,WidthStreet,start_x);

 axes(handles.axes1);
 plot(AnchPos(1,1:length(AnchPos)),AnchPos(2,1:length(AnchPos)),'x',...
     'markersize',5);
 set(handles.axes1, 'XLim', [0 4*lengthStreet+3*WidthStreet]);
 set (handles.axes1, 'YLim', [0 3*lengthStreet+2*WidthStreet]);
 xlabel('distance in m');
 ylabel('distance in m');
video=0;
% --- Outputs from this function are returned to the command line.
function varargout = TrackingApp_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object deletion, before destroying properties.
function figure1_DeleteFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Archivo_Callback(hObject, eventdata, handles)
% hObject    handle to Archivo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Instructions_Callback(hObject, eventdata, handles)
% hObject    handle to Instructions (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

open Instructions.fig

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double
sigma_shad=str2double(get(handles.edit1,'string'));
if isnan(sigma_shad)
    errordlg('You must enter a numeric value to Sigma_shad','Bad Input','modal')
    set(handles.edit1,'String',2);
end
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double

std_ml=str2double(get(handles.edit2,'string'));
if isnan(std_ml)
    errordlg('You must enter a numeric value to ML Standard Deviation',...
        'Bad Input','modal')
    set(handles.edit2,'String',3);
```

```matlab

end
% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double

T_motes=str2double(get(handles.edit3,'string'));
if isnan(T_motes)
    errordlg('You must enter a numeric value to Sensor Activation',...
        'Bad Input','modal')
    set(handles.edit3,'String',0.5);

end

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double

pathloss=str2double(get(handles.edit4,'string'));
if isnan(pathloss)
    errordlg('You must enter a numeric value to path loss exponent',...
        'Bad Input','modal')
    set(handles.edit4,'String',2);

end

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double

P0=str2double(get(handles.edit5,'string'));
if isnan(P0)
    errordlg('You must enter a numeric value to P0',...
        'Bad Input','modal')
    set(handles.edit5,'String',-63.2);

end

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double

sigma_kf=str2double(get(handles.edit6,'string'));
if isnan(sigma_kf)
    errordlg('You must enter a numeric value to KF_sigma',...
        'Bad Input','modal')
    set(handles.edit6,'String',0.1);

end

% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double

sigma_ekf1=str2double(get(handles.edit7,'string'));
if isnan(sigma_ekf1)
```

```
    errordlg('You must enter a numeric value to EKF_sigma1',...
        'Bad Input','modal')
    set(handles.edit7,'String',0.5);

end
% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double
sigma_ekf2=str2double(get(handles.edit8,'string'));
if isnan(sigma_ekf2)
    errordlg('You must enter a numeric value to EKF_sigma2',...
        'Bad Input','modal')
     set(handles.edit8,'String',0.2);

end

% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


```

```
function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double

p11=str2double(get(handles.edit9,'string'));
if isnan(p11)
    errordlg('Enter a numeric value to (1,1) of the transition matrix',...
        'Bad Input','modal')
    set(handles.edit9,'String',0.95);

end
% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a double

p12=str2double(get(handles.edit10,'string'));
if isnan(p12)
    errordlg('Enter a numeric value to (1,2) of the transition matrix',...
        'Bad Input','modal')
    set(handles.edit10,'String',0.05);

end
% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double
p21=str2double(get(handles.edit11,'string'));
if isnan(p21)
    errordlg('Enter numeric value to (2,1) of the transition matrix',...
        'Bad Input','modal')
    set(handles.edit11,'String',0.10);

end

% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double
p22=str2double(get(handles.edit12,'string'));
if isnan(p22)
```

```
    errordlg('Enter a numeric value to (2,2) of the transition matrix',...
        'Bad Input','modal')
    set(handles.edit12,'String',0.90);

end

% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit13_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
%        str2double(get(hObject,'String')) returns contents of edit12 as a double

mu_ij11=str2double(get(handles.edit13,'string'));
if isnan(mu_ij11)
    errordlg('Enter a numeric value to (1,1) of the Model Probabilities',...
        'Bad Input','modal')
    set(handles.edit13,'String',0.5);

end
% --- Executes during object creation, after setting all properties.
function edit13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```
function edit14_Callback(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit13 as text
%        str2double(get(hObject,'String')) returns contents of edit13 as a double
mu_ij12=str2double(get(handles.edit14,'string'));
if isnan(mu_ij12)
    errordlg('Enter a numeric value to (1,2) of the Model Probabilities',...
        'Bad Input','modal')
    set(handles.edit14,'String',0.5);

end

% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit15_Callback(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit15 as text
%        str2double(get(hObject,'String')) returns contents of edit15 as a double
Ttarget=str2double(get(handles.edit15,'string'));
if isnan(Ttarget)
    errordlg('You must enter a numeric value to Target sampling',...
        'Bad Input','modal')
    set(handles.edit15,'String',0.5);

end

% --- Executes during object creation, after setting all properties.
function edit15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in Multilateration.
function Multilateration_Callback(hObject, eventdata, handles)
% hObject    handle to Multilateration (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if (get(handles.Multilateration,'Value')==1)
set(handles.WeightedAv,'enable','off');
set(handles.WeightedAv,'Value',0);
set(handles.text17,'String', 'RMS Multilateration');
else
set(handles.WeightedAv,'enable','on');
set(handles.WeightedAv,'Value',1);
set(handles.text17,'String', 'RMS Weighted Average');
end
% Hint: get(hObject,'Value') returns toggle state of Multilateration


% --- Executes on button press in WeightedAv.
function WeightedAv_Callback(hObject, eventdata, handles)
% hObject    handle to WeightedAv (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if (get(handles.WeightedAv,'Value')==1)
set(handles.Multilateration,'enable','off');
set(handles.Multilateration,'Value',0);
set(handles.text17,'String', 'RMS Weighted Average');
else
    set(handles.Multilateration,'enable','on');
    set(handles.Multilateration,'Value',1);
    set(handles.text17,'String', 'RMS Multilateration');

end
% Hint: get(hObject,'Value') returns toggle state of WeightedAv

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.pushbutton2,'enable','off');
set(handles.edit23,'enable','on');
set(handles.edit24,'enable','on');
set(handles.edit25,'enable','on');
set(handles.edit26,'enable','on');
sigma_shad1=str2double(get(handles.edit1,'string'));
std_ml=str2double(get(handles.edit2,'string'));
T_motes=str2double(get(handles.edit3,'string'));
Ttarget=str2double(get(handles.edit15,'string'));
Groups=get(handles.popupmenu2,'value');
pathloss=str2double(get(handles.edit4,'string'));
P0=str2double(get(handles.edit5,'string'));
route=get(handles.popupmenu3,'value');
sigma_kf=str2double(get(handles.edit6,'string'));
sigma_ekf1=str2double(get(handles.edit7,'string'));
sigma_ekf2=str2double(get(handles.edit8,'string'));
p11=str2double(get(handles.edit9,'string'));
p12=str2double(get(handles.edit10,'string'));
p21=str2double(get(handles.edit11,'string'));
p22=str2double(get(handles.edit12,'string'));
mu_ij11=str2double(get(handles.edit13,'string'));
mu_ij12=str2double(get(handles.edit14,'string'));
pathloss_dist=str2double(get(handles.edit27,'string'));


pij=[p11 p12;p21 p22];
muij=[mu_ij11 mu_ij12];

%while get(handles.pushbutton1,'UserData')==1;
cla(handles.axes1,'reset');
guidata(hObject,handles);

if (get(handles.Multilateration,'Value')==1)
Multilat=1;
else
    Multilat=0;
end
if get(handles.radiobutton1,'Value')==1
    video=1;
else
    video=0;
end

[mu,finish]=TrackingIMMKFandEKF(sigma_shad1,std_ml,T_motes,Ttarget,...
    Groups,pathloss,pathloss_dist, P0,route,sigma_kf,sigma_ekf1,...
    sigma_ekf2,pij,muij,Multilat, video, hObject, eventdata, handles);
handles.muProb=mu;
if finish ==1
    set(handles.pushbutton2,'enable','on');
end
%end
guidata(hObject, handles);

% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu2 contents as
%cell array. contents{get(hObject,'Value')} returns selected item
%from popupmenu2
if get(handles.popupmenu2,'value')==2
    set(handles.edit3,'String',5);
    set(handles.edit3,'enable','off');
else
    set(handles.edit3,'String','0.5');
    set(handles.edit3,'enable','on');
end

% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu3.
function popupmenu3_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu3 contents as
% cell array. contents{get(hObject,'Value')} returns selected item from popupmenu3


% --- Executes during object creation, after setting all properties.
function popupmenu3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
modeProb();


function edit23_Callback(hObject, eventdata, handles)
% hObject    handle to edit23 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit23 as text
%        str2double(get(hObject,'String')) returns contents of edit23 as a double


% --- Executes during object creation, after setting all properties.
function edit23_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit23 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit24_Callback(hObject, eventdata, handles)
% hObject    handle to edit24 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit24 as text
%        str2double(get(hObject,'String')) returns contents of edit24 as a double


% --- Executes during object creation, after setting all properties.
function edit24_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit24 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit25_Callback(hObject, eventdata, handles)
% hObject    handle to edit25 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit25 as text
%        str2double(get(hObject,'String')) returns contents of edit25 as a double

% --- Executes during object creation, after setting all properties.
function edit25_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit25 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit26_Callback(hObject, eventdata, handles)
% hObject    handle to edit26 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit26 as text
%        str2double(get(hObject,'String')) returns contents of edit26 as a double


% --- Executes during object creation, after setting all properties.
function edit26_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit26 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes when figure1 is resized.
function figure1_ResizeFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton1



function edit27_Callback(hObject, eventdata, handles)
% hObject    handle to edit27 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit27 as text
%        str2double(get(hObject,'String')) returns contents of edit27 as a double
pathloss_dist=str2double(get(handles.edit27,'string'));
if isnan(pathloss_dist)
    errordlg('You must enter a numeric value to path loss exponent',...
        'Bad Input','modal')
    set(handles.edit27,'String',3);

```

```
end

% --- Executes during object creation, after setting all properties.
function edit27_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit27 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),...
        get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Listing B.2: MATLAB code of TrackingIMMKFandEKF.m

```
function [MU,finish] = Tracking_IMM_KF_and_EKF(sigma_shad1,std_ML,...
    Tmotes,Ttarget,Groups,pathloss,pathloss_dist,P_0,route,sigma_kf,...
    sigma_ekf1,sigma_ekf2,pij,muij,Multilat,video,hObject,eventdata, handles)

%-----------------------Tracking simulator -------------------------------
%--------------------------------General scenario-------------------------
%In this simulator we track a car using multilateration technique, that is
%moving between different streets in a urban area.

%The scenario is a set of streets of 20m of width, 4m between anchor nodes
%(x) and 12 parking slots per side

%------------------------------------------------------------------------

%----------------------------STREET PARAMETERS----------------------------
Anchors_Street = 24; %Number of total Anchors (12 per side)
distAnch = 4; %distance between anchors
WidthStreet=20;
start_x = 2; %this would be the distance from the reference where
%the parking lots begin
lengthStreet = 2*start_x +distAnch*((Anchors_Street/2)-1);
%-------------------------------------------------------------------------

%-------------------------SCENARIO AREA -----------------------------------
block_horiz = 4; %total of horiz "block"
block_vert = 3;  %total of vert "block"
cr=(block_horiz-1)*(block_vert-1); %number of croisses
streets=cr;%Number of streets
Scenario_Width = block_horiz*lengthStreet+WidthStreet*(block_horiz-1);
```

```
Scenario_Height = block_vert*lengthStreet+WidthStreet*(block_vert-1);
%------------------------------------------------------------------------

%-------------------------ANCHORS PARAMETERS-----------------------------
Ngroups=Groups; %number of groups of activated intercalated Anchors

% %--------------------DEFINE THE POSITIONS OF THE ANCHORS ----------------
% %We associate the set of anchors of one street to a number that identifies
% %this street.
%
 AnchPos=AnchorsPosDefinition(Anchors_Street,block_vert,block_horiz,...
     lengthStreet,distAnch,WidthStreet,start_x);

%---------------------DEFINE THE CROSSINGS IN MATRICES--------------------
%As we have 6 crs in a 4x3 blocks:
cr1=[lengthStreet lengthStreet+WidthStreet;...
    2*(lengthStreet+WidthStreet) 2*lengthStreet+WidthStreet];
cr2=[2*lengthStreet+WidthStreet 2*(lengthStreet+WidthStreet);...
    2*(lengthStreet+WidthStreet) 2*lengthStreet+WidthStreet];
cr3=[3*lengthStreet+2*WidthStreet 3*(lengthStreet+WidthStreet);...
    2*(lengthStreet+WidthStreet) 2*lengthStreet+WidthStreet];

cr4=[lengthStreet lengthStreet+WidthStreet;...
    lengthStreet+WidthStreet lengthStreet];
cr5=[2*lengthStreet+WidthStreet 2*(lengthStreet+WidthStreet);...
    lengthStreet+WidthStreet lengthStreet];
cr6=[3*lengthStreet+2*WidthStreet 3*(lengthStreet+WidthStreet);...
    lengthStreet+WidthStreet lengthStreet];

%-------------------------TARGET PARAMETERS------------------------------
%Now let's start assuming that a mobile node M travels along a certain path
%with constant velocity. The following parameters are required:
switch route
    case 1
load 'route1_target_route'
    case 2
load 'route2_target_route'
    case 3
load 'route3_target_route'
    case 4
load 'route4_target_route'
    case 5
load 'route5_target_route'
    case 6
load 'route6_target_route'
end

```

```
%The target route is previously computed with a sampling period of
%T_target=0.01. Then some samples are taken from all the route:
T=Ttarget;
Target_samples=T/0.01;
T_motes=Tmotes; %sampling time in seconds of the active motes.
%Every T_motes sg, all active anchors send a message to the target.
T_shift = T_motes/Ngroups; %Time shift between the first and second group
%of a total of Ngroups
s=(0:Ngroups-1)*T_shift; %This vector contains all the shifts between
%the group1 and the other anchors groups
s=roundn(s,-1);
anchors_ML = 4; %number of considered anchors for multilateration computation

%-------------------------PATH LOSS MODEL PARAMETERS-----------------------
sigma_shad = sigma_shad1; %shadow fading variance in dB as the power of
%the noise added to the received power
P0 = P_0; %Received power at 1 m used for the received power model in dBm
gamma=pathloss;
gamma_dist=pathloss_dist;
%------------------------------------------------------------------------

%---------------------KALMAN PARAMETERS AND COEFFICIENTS-----------------
std_ml=std_ML; %standard deviation in m due to the multilateration computation
var_ml=std_ml^2;
models=2;
V=cell(1,models); %covariance of the white-gaussian acceleration noise
%of the state model
sigma_a=sigma_kf; %acceleration variance in velocity for KF
sigma_a1=sigma_ekf1; %acceleration variance in velocity for EKF
sigma_a2=sigma_a1; %acceleration variance in turn rate for EKF
sigma_a3=sigma_ekf2;
V{1}=diag([sigma_a,sigma_a]);
V{2}=diag([sigma_a1,sigma_a2,sigma_a3]);
R=[var_ml 0;0 var_ml]; %measurement noise covariance matrix
x_jk=cell(1,models);%This cell used in the IMM algorithm contains
%the state vectors of both KF and EKF which are updated in each time step
%using the IMM
P_jk=cell(1,models);%This cell used in the IMM algorithm contains
%the state covariance matrices for both KF and EKF which are updated in each time step
%using the IMM
P_jk{1}=[var_ml var_ml/T_shift 0 0; var_ml/T_shift 2*var_ml/T_shift^2 ...
    0 0; 0 0 var_ml var_ml/T_shift; 0 0 var_ml/T_shift 2*var_ml/T_shift^2];
P_jk{2}=[var_ml var_ml/T_shift 0 0 0; var_ml/T_shift 2*var_ml/T_shift^2 ...
    0 0 0; 0 0 var_ml var_ml/T_shift 0; 0 0 var_ml/T_shift ...
    2*var_ml/T_shift^2 0;0 0 0 0 0.01];
%Below the initial state covariance matrix for the KF
P_KF=P_jk{1};
```

```
%Below the initial state covariance matrix for the EKF
P_EKF=P_jk{2};
H=cell(1,models);
H{1}=[1 0 0 0;0 0 1 0];
H{2}=[1 0 0 0 0;0 0 1 0 0];
mu_ij=muij;
p_ij=pij;
F=cell(1,models);
F{1}=[1 T_shift  0 0;0 1 0 0;0 0 1 T_shift ;0 0 0 1]; %KF transition matrix
norm_ML=[];
norm_KF=[];
norm_EKF=[];
norm_IMM=[];
%-------------------------------------------------------------------------

ml=1; %Target estimated samples
count=0;
n=1;

%------------PARAMETERS FOR GENERATING THE MOVIE ------
if (video==1)
fps=3;
 fn = strcat('Tmotes_',num2str(T_motes),'sg_',num2str(Ngroups),...
     'group_route',num2str(route),'Sigma_shad',num2str(sigma_shad)','dB.mov');
 MakeQTMovie('start',fn);
 MakeQTMovie('quality', 1);
end


%----------------------------BEGIN THE TRACKING ---------------------------

for x=1:Target_samples:length(target_state)
   %Now all the received powers of all the anchors are gathered by the
    %target. We use the simple path loss model:
    %Pr(dBm)=P0(dBm)-10·gamma·log10(d_i), where d_i is the euclidean
    %distance from the mobile target to anchor i

    %Below we check if the target is inside a cr

 if ((cr1(1,1)≤target_state(1,x) && target_state(1,x)≤cr1(1,2)) && ...
 (cr1(2,2)≤target_state(3,x) && target_state(3,x)≤cr1(2,1)) || ...
   (cr2(1,1)≤target_state(1,x) && target_state(1,x)≤cr2(1,2)) && ...
 (cr2(2,2)≤target_state(3,x) && target_state(3,x)≤cr2(2,1))  || ...
   (cr3(1,1)≤target_state(1,x) && target_state(1,x)≤cr3(1,2)) && ...
 (cr3(2,2)≤target_state(3,x) && target_state(3,x)≤cr3(2,1))  || ...
   (cr4(1,1)≤target_state(1,x) && target_state(1,x)≤cr4(1,2)) && ...
 (cr4(2,2)≤target_state(3,x) && target_state(3,x)≤cr4(2,1))  || ...
```

```
   (cr5(1,1)≤target_state(1,x) && target_state(1,x)≤cr5(1,2)) && ...
 (cr5(2,2)≤target_state(3,x) && target_state(3,x)≤cr5(2,1))  || ...
  (cr6(1,1)≤target_state(1,x) && target_state(1,x)≤cr6(1,2)) && ...
 (cr6(2,2)≤target_state(3,x) && target_state(3,x)≤cr6(2,1)))

%Next we need to check wether in which cr is located the target
%------------------if the target is cr the cr 1 ----------------
  if ((cr1(1,1)≤target_state(1,x)) && (target_state(1,x)≤cr1(1,2)) && ...
 (cr1(2,2)≤target_state(3,x)) && (target_state(3,x)≤cr1(2,1)))

  a=find(abs(s-count)<1/1000);

  if (length(a)>0) %when count has a value of one of the activation times
      %of the different set of Anchors
     Pos_Anchors=[cr1(1,1)-start_x cr1(1,2)+start_x cr1(1,1)-start_x ...
     cr1(1,2)+start_x; cr1(2,1) cr1(2,1) cr1(2,2) ...
     cr1(2,2)];

  [estimated_pos_target(:,ml),ActiveAnch]= Multilateration(target_state(:,x),...
      Pos_Anchors,AnchPos,P0,gamma,gamma_dist,sigma_shad,anchors_ML,...
      Anchors_Street,1,a,Ngroups,distAnch,Multilat);
   activated=1;
  else
   activated=0;
   end
 end

%----------------------if the target is cr the cr 2-------
   if ((cr2(1,1)≤target_state(1,x) && target_state(1,x)≤cr2(1,2)) && ...
 (cr2(2,2)≤target_state(3,x) && target_state(3,x)≤cr2(2,1)))

  a=find(abs(s-count)<1/1000);

  if (length(a)>0) %when count has a value of one of the activation times
      %of the different set of Anchors
     Pos_Anchors=[cr2(1,1)-start_x cr2(1,2)+start_x cr2(1,1)-start_x ...
     cr2(1,2)+start_x; cr2(2,1) cr2(2,1) cr2(2,2) ...
     cr2(2,2)];

  [estimated_pos_target(:,ml),ActiveAnch]= Multilateration(target_state(:,x),...
      Pos_Anchors,AnchPos,P0,gamma,gamma_dist,sigma_shad,anchors_ML,...
      Anchors_Street,1,a,Ngroups,distAnch, Multilat);
   activated=1;
  else
   activated=0;
 end
  end

 %----------------------if the target is cr the cr 3----------
   if ((cr3(1,1)≤target_state(1,x) && target_state(1,x)≤cr3(1,2)) && ...
 (cr3(2,2)≤target_state(3,x) && target_state(3,x)≤cr3(2,1)))
    a=find(abs(s-count)<1/1000);

  if (length(a)>0) %when count has a value of one of the activation times
      %of the different set of Anchors
  Pos_Anchors=[cr3(1,1)-start_x cr3(1,2)+start_x cr3(1,1)-start_x ...
  cr3(1,2)+start_x; cr3(2,1) cr3(2,1) cr3(2,2) ...
  cr3(2,2)];

  [estimated_pos_target(:,ml),ActiveAnch]= Multilateration(target_state(:,x),...
      Pos_Anchors,AnchPos,P0,gamma,gamma_dist,sigma_shad,anchors_ML,...
      Anchors_Street,1,a,Ngroups,distAnch,Multilat);
   activated=1;
  else
   activated=0;
   end
   end
 %----------------------if the target is cr the cr 4-------
 if ((cr4(1,1)≤target_state(1,x) && target_state(1,x)≤cr4(1,2)) && ...
 (cr4(2,2)≤target_state(3,x) && target_state(3,x)≤cr4(2,1)))
  a=find(abs(s-count)<1/1000);

  if (length(a)>0) %when count has a value of one of the activation times
      %of the different set of Anchors
     Pos_Anchors=[cr4(1,1)-start_x cr4(1,2)+start_x cr4(1,1)-start_x ...
      cr4(1,2)+start_x; cr4(2,1) cr4(2,1) cr4(2,2) ...
       cr4(2,2)];

     [estimated_pos_target(:,ml),ActiveAnch]= Multilateration(target_state(:,x),...
         Pos_Anchors, AnchPos,P0,gamma,gamma_dist,sigma_shad,anchors_ML,...
         Anchors_Street,1,a,Ngroups,distAnch,Multilat);
    activated=1;
  else
   activated=0;
 end
end
%----------------------if the target is cr the cr 5-----------
  if ((cr5(1,1)≤target_state(1,x) && target_state(1,x)≤cr5(1,2)) && ...
 (cr5(2,2)≤target_state(3,x) && target_state(3,x)≤cr5(2,1)))
  a=find(abs(s-count)<1/1000);

  if (length(a)>0) %when count has a value of one of the activation times
      %of the different set of Anchors
     Pos_Anchors=[cr5(1,1)-start_x cr5(1,2)+start_x cr5(1,1)-start_x ...
```

```
        cr5(1,2)+start_x; cr5(2,1) cr5(2,1) cr5(2,2) ...
         cr5(2,2)];

      [estimated_pos_target(:,ml),ActiveAnch]= Multilateration(target_state(:,x),...
          Pos_Anchors,AnchPos,P0,gamma,gamma_dist,sigma_shad,anchors_ML,...
          Anchors_Street,1,a,Ngroups,distAnch,Multilat);
     activated=1;
   else
       activated=0;
   end
 end
%----------------------if the target is cr the cr 6-------

 if ((cr6(1,1)≤target_state(1,x) && target_state(1,x)≤cr6(1,2)) && ...
 (cr6(2,2)≤target_state(3,x) && target_state(3,x)≤cr6(2,1)))

  a=find(abs(s-count)<1/1000);

  if (length(a)>0) %when count has a value of one of the activation times
      %of the different set of Anchors
     Pos_Anchors=[cr6(1,1)-start_x cr6(1,2)+start_x cr6(1,1)-start_x ...
      cr6(1,2)+start_x; cr6(2,1) cr6(2,1) cr6(2,2) ...
       cr6(2,2)];

      [estimated_pos_target(:,ml),ActiveAnch]= Multilateration(target_state(:,x),...
          Pos_Anchors,AnchPos,P0,gamma,gamma_dist,sigma_shad,anchors_ML,...
          Anchors_Street,1,a,Ngroups,distAnch,Multilat);
      activated=1;
    else
        activated=0;
 end
end

  else %If the target is not in a cr

  %-------------------PERFORMING MULTILATERATION IN THE STREETS------------
  a=find(abs(s-count)<1/1000);

  if (length(a)>0) %when count has a value of one of the activation times
      %of the different set of Anchors
 [estimated_pos_target(:,ml),ActiveAnch]= Multilateration(target_state(:,x),...
     AnchPos,AnchPos,P0,gamma,gamma_dist,sigma_shad,anchors_ML,...
     Anchors_Street,0,a,Ngroups,distAnch,Multilat);
   activated=1;

  else
    activated=0;
  end
  end

   if size(estimated_pos_target,2)==1
       %KF, EKF and IMM initializations to the first measurement
       KF_estimates=zeros(4,1);
       EKF_estimates=zeros(5,1);
       KF_estimates([1 3],:)=estimated_pos_target;
       EKF_estimates([1 3],:)=estimated_pos_target;
       x_jk{1}=KF_estimates;
       x_jk{2}=EKF_estimates;
       x_combupd1(:,1)=EKF_estimates;
       MU(:,1)=mu_ij;
       norm_ML(ml)=norm([target_state(1,x);target_state(3,x)]-...
           [estimated_pos_target(1,1);estimated_pos_target(2,1)]);
       norm_KF(ml)=norm([target_state(1,x);target_state(3,x)]-...
       [ KF_estimates(1,1);KF_estimates(3,1)]);
       norm_EKF(ml)=norm([target_state(1,x);target_state(3,x)]-...
       [EKF_estimates(1,1) ;EKF_estimates(3,1)]);
       norm_IMM(ml)=norm([target_state(1,x);target_state(3,x)]-...
       [x_combupd1(1,1);x_combupd1(3,1)]);

   end

   if activated==1
   if ml≥2
  %-------------------APPLY KALMAN FILTER--------------------------------
 [x_updKF,P_updKF,x_predKF,P_predKF]=...
     Kalman(KF_estimates(:,ml-1),P_KF,F{1},H{1},V{1},R,...
     estimated_pos_target(:,ml),T_shift);
P_KF=P_updKF;
 KF_estimates(:,ml)=x_updKF;

 %-----------------APPLY EXTENDED KALMAN FILTER---------------------------------
A = chooseTransitionMatrix(EKF_estimates(:,ml-1),T_shift);
F{2}=A;
 [x_updEKF,P_updEKF,x_predEKF,P_predEKF]=...
     ExtendedKalman(EKF_estimates(:,ml-1),P_EKF,F{2},H{2},V{2},R,...
     estimated_pos_target(:,ml),T_shift);
 P_EKF=P_updEKF;
 EKF_estimates(:,ml)=x_updEKF;

 %-----------------APPLY INTERACTIVE MULTIPLE MODEL-CT-------------------
A = chooseTransitionMatrix(x_jk{2},T_shift);
 F{2}=A;
[x_combupd,P_combupd,x_combpred,P_combpred,x_jk1,P_jk1,mu_ijk1]= ...
    immct(mu_ij,p_ij,x_jk,P_jk,F,H,V,R,estimated_pos_target(:,ml),T_shift);
    mu_ij=mu_ijk1;
    x_jk=x_jk1;
    P_jk=P_jk1;
    x_combupd1(:,ml)=x_combupd;
    MU(:,ml)=mu_ij;
 %--------------------------PLOTTING RESULTS ----------------------------
  axes(handles.axes1);
  h1=plot(AnchPos(1,1:length(AnchPos)),...
      AnchPos(2,1:length(AnchPos)),'x','markersize',5);
  hold on
  set(handles.axes1, 'XLim', [0 4*lengthStreet+3*WidthStreet]);
  set (handles.axes1, 'YLim', [0 3*lengthStreet+2*WidthStreet]);
  xlabel('distance in m');
  ylabel('distance in m');
  h2=plot(target_state(1,x),target_state(3,x),'^',...
      'Color',[47;79;79]/255,'markersize',5);
  h3=plot(ActiveAnch(1,1:length(ActiveAnch)),ActiveAnch(2,...
      1:length(ActiveAnch)),'x','Color',[9;249;17]/255,'markersize',5);
  h4=plot( estimated_pos_target(1,ml), estimated_pos_target(2,ml),...
      '.r','markersize',20);
  %'Color',[192;192;192]/255);
  h5=plot(KF_estimates(1,1:ml), KF_estimates(3,1:ml),'k','LineWidth',2);
  h6=plot(EKF_estimates(1,1:ml),EKF_estimates(3,1:ml),...
      'Color',[208;32;144]/255,'LineWidth',2);%[255;130;171]/255);
  h7=plot(x_combupd1(1,1:ml),x_combupd1(3,1:ml),'Color',...
      [85;26;139]/255,'LineWidth',2);%[153;102;204]/255);
  legend([h1 h2 h3 h4 h5 h6 h7],'Parking Nodes','True track',...
   'Active Nodes', 'measurements', 'KF','EKF','IMM-CT','Location','Best');
  MakeQTMovie('addplot');
  norm_ML(ml)=norm_ML(ml-1)+norm([target_state(1,x);target_state(3,x)]-...
  [estimated_pos_target(1,ml);estimated_pos_target(2,ml)]);
  norm_KF(ml)=norm_KF(ml-1)+norm([target_state(1,x);target_state(3,x)]-...
  [KF_estimates(1,ml);KF_estimates(3,ml)]);
  norm_EKF(ml)=norm_EKF(ml-1)+norm([target_state(1,x);target_state(3,x)]-...
  [EKF_estimates(1,ml);EKF_estimates(3,ml)]);
   norm_IMM(ml)=norm_IMM(ml-1)+norm([target_state(1,x);target_state(3,x)]-...
  [x_combupd1(1,ml);x_combupd1(3,ml)]);
    set(handles.edit23,'String',num2str(norm_ML(ml)/ml));
    set(handles.edit24,'String',num2str(norm_KF(ml)/ml));
    set(handles.edit25,'String',num2str(norm_EKF(ml)/ml));
    set(handles.edit26,'String',num2str(norm_IMM(ml)/ml));

  end
   if a==length(s)
    s=s+T_motes;
    s=roundn(s);
  end
  ml=ml+1;
   elseif activated==0
    axes(handles.axes1);
    plot(AnchPos(1,1:length(AnchPos)),AnchPos(2,1:length(AnchPos)),...
        'x','markersize',5);
    hold on
    set(handles.axes1, 'XLim', [0 4*lengthStreet+3*WidthStreet])
    set (handles.axes1, 'YLim', [0 3*lengthStreet+2*WidthStreet]);
    plot(target_state(1,x),target_state(3,x),'^','Color',[47;79;79]/255,...
        'markersize',5);
    if (video==1)
   MakeQTMovie('addplot');
    end
   end
   count=count+T;
   count=roundn(count);
   n=n+1;
end

  MakeQTMovie('framerate', fps);
  MakeQTMovie('finish');
finish = 1;
  RMS_ML=norm_ML/sqrt(ml);
  RMS_KF=norm_KF/sqrt(ml);
  RMS_EKF=norm_EKF/sqrt(ml);
  RMS_IMM=norm_IMM/sqrt(ml);
```

Listing B.3: MATLAB code of Multilateration.m

```
%-----------Performing Multilateration technique out of the crossings------------
function [estimated_pos,AllActive]=Multilateration(target_state,AnchorPos,...
    AllAnchors,P0,gamma,gamma_dist,sigma_shad,anchors_ML,Anchors_Street,...
    incrossing,Ngroup,Ngroups,distAnch,Multilat)

if incrossing ==1 %If the target is in a crossing, compute multilateration
    %with those anchor nodes located at the both sides in x.

 target=[target_state(1);target_state(3)];
 %Next we choose the four neares active anchors of horizontal sides
 AnchPos= [AnchorPos(1,1)-(distAnch*(Ngroups-Ngroup))...
     AnchorPos(1,2)+(distAnch*(Ngroup-1))...
     AnchorPos(1,3)-(distAnch*(Ngroup-1))...
     AnchorPos(1,4)+(distAnch*(Ngroups-Ngroup));...
     AnchorPos(2,1) AnchorPos(2,2) AnchorPos(2,3) AnchorPos(2,4)];

 dist = sqrt((target(1,1)-AnchPos(1,:)).^2+(target(2,1)- AnchPos(2,:)).^2);
 Pr=P0-(10*gamma*log10(dist))+sqrt(sigma_shad)*randn(1,length(dist));

 if Multilat==1
%Variable d_rssi is the estimated distance from RSSI to each of the
%anchors considered to compute multilateration
 d_rssi= 10.^((P0-Pr(1,:))/(10*gamma_dist));
 estimated_pos = fminunc(@(pos_target_est) posfun(pos_target_est,...
     AnchPos,d_rssi),[0;0]);
 elseif Multilat==0
 %WEIGHTED AVERAGE METHOD FOR POSITIONING INSTEAD OF MULTILATERATION.
%IN PRACTICE  WEIGHTED AVERAGE IS BETTER (PRECIS) THAN MULTILATERATIO
 weightedCoeffs = weightedAverage(Pr); %It is a Nx1 vector (N=anchors_ml)
  estimated_pos= [AnchPos(1,:);AnchPos(2,:)]*weightedCoeffs;
 end

elseif incrossing==0 %If the target is in  a certain street

  target=[target_state(1);target_state(3)];
  Anchors_id=find(AnchorPos(3,:)==target_state(5));
  PosAnch=AnchorPos(:,Anchors_id);
  %The following switch defines a matrix with triangle geometry of the
  %Active Anchors Positions
  switch Ngroup
      case 1
c=[Ngroup:2*Ngroups:length(PosAnch) 2*Ngroups:2*Ngroups:length(PosAnch)];
c=sort(c);
SubPosAnch=PosAnch(:,c);
      otherwise
c=[2*Ngroup-1:2*Ngroups:length(PosAnch) 2*Ngroup-2:2*Ngroups:length(PosAnch)];
c=sort(c);
SubPosAnch=PosAnch(:,c);
  end

dist = sqrt((target(1,1)-SubPosAnch(1,:)).^2+(target(2,1)-SubPosAnch(2,:)).^2);
Pr=P0-(10*gamma*log10(dist))+sqrt(sigma_shad)*randn(1,length(dist));

 %We associate in a matrix P an ID to each anchor node and to the
%corresponding received power from that anchor node
P = zeros(2,length(Pr));
P(1,:) = Pr;
P(2,:) = SubPosAnch(3,:);

%Now it is going to compute multilateration with those anchors_ML
%specified with the highest received power:

 Pr_max=zeros(2,anchors_ML); %first row contains
 cont_up=0;
 cont_down=0;
 A=P;
 for kk=1:anchors_ML
   if (cont_down < round(anchors_ML/2))
      [i,j]=max(A(1,:));
      Pr_max(1,kk)=i;
      Pr_max(2,kk)=j;
      A(1,j)=-200;
     cont_down=cont_down+1;
  elseif (cont_up < Anchors_Street-round(anchors_ML/2)) && ...
    (cont_down ≥ round(anchors_ML/2))
    if target_state(4) <0 || target_state(2)<0
        if mod(Ngroup,2)==1
             [a,b]=max(A(1,2:2:length(A)));
             b=2*b;
        elseif mod(Ngroup,2)==0
             [a,b]=max(A(1,1:2:length(A)));
              b=2*b-1;
        end
    else
     if mod(Ngroup,2)==0
       [a,b]=max(A(1,2:2:length(A)));
       b=2*b;
     elseif mod(Ngroup,2)==1
       [a,b]=max(A(1,1:2:length(A)));
        b=2*b-1;
     end
   end
      Pr_max(1,kk)=a;
      Pr_max(2,kk)=b;
      A(1,b)=-200;
      cont_up=cont_up+1;
   end
 end

if Multilat==1
d_rssi= 10.^((P0-Pr_max(1,:))/(10*gamma_dist));
  %----Below multilateration is solved optimizated without constraints  ----
estimated_pos = fminunc(@(pos_target_est) posfun(pos_target_est,...
   SubPosAnch(:,Pr_max(2,:)),d_rssi),[0;0]);
elseif Multilat==0  %WEIGHTED AVERAGE METHOD
SubPosAnch=SubPosAnch(:,Pr_max(2,:));
weightedCoeffs = weightedAverage(Pr_max(1,:)); %It is a Nx1 vector(N=anchors_ml)
estimated_pos= [SubPosAnch(1,:);SubPosAnch(2,:)]*weightedCoeffs;

end
end
```

```

switch Ngroup
      case 1
c=[Ngroup:2*Ngroups:length(AllAnchors) 2*Ngroups:2*Ngroups:length(AllAnchors)];
c=sort(c);
AllActive=AllAnchors(:,c);
      otherwise
c=[2*Ngroup-1:2*Ngroups:length(AllAnchors) 2*Ngroup-2:2*Ngroups:length(AllAnchors)];
c=sort(c);
AllActive=AllAnchors(:,c);
end
```

Listing B.4: MATLAB code of weightedAverage.m

```
%Weighted Average Received power:
%This scripts returns coefficients as the weighted received power defined
%as follows:
% alfa_n = Rssi_n/sum(Rssi(1:N)) where n is in a range between 1 to N and
%N is the number of considered anchors to perform the positioning.
%Rssi is a 1xN vector containing the received powers of all the N anchors.

function weighted_coeffs = weightedAverage(Rssi)

alfas = zeros(1,length(Rssi));

Rssi_sum = sum(Rssi);

for m=1:length(alfas)
    alfas(m)= Rssi(m)/Rssi_sum;
end

weighted_coeffs=alfas';
```

Listing B.5: MATLAB code of posfun.m

```
function F=pos_fun(vec_pos,pos_anclas,medida)

% pos_anclas es la matriz donde cada columna son las componentes x e y de
% los anclas;
% medida es un vector donde cada fila es la medida tomada por cada ancla;


F=0;

for kk=1:size(pos_anclas,2)
```

```

F = F + (sqrt((pos_anclas(1,kk)-vec_pos(1))^2+(pos_anclas(2,kk)-...
    vec_pos(2))^2)-medida(kk))^2;


end
```

Listing B.6: MATLAB code of Kalman.m

```
function [x_updKF,P_updKF,x_predKF,P_predKF,likelihoodKF]=...
    Kalman(x_0j,P_0j,F,H,V,R,z,T)

[k_kf,S,P_predKF,P_updKF]=genKalmancoefskf(R,P_0j,F,H,T,V);
%generate online the Kalman coefficient, the state covariance matrices
%prediction and the update for each received measurement z.

x_predKF=F*x_0j;
error_kf=z-H*x_predKF;
%correction=k_kf(:,:,x)*error_kf;
correction=k_kf*error_kf;
x_updKF=correction+F*x_0j;
if nargout >4
likelihoodKF=(1/((2*pi)^0.5*det(S)^0.5))*...
    exp(-0.5*(z-x_predKF([1 3],:))'*inv(S)*(z-x_predKF([1 3],:)));
end
```

Listing B.7: MATLAB code of genKalmancoefskf.m

```
function [k,S,P_pred,P_upd] = gen_Kalman_coefs_kf(R,P,F,H,T,V)

tau=[(1/2)*T^2 0;T 0;0 (1/2)*T^2;0 T]; %--> This suppose to be the
%known matrix. It takes into account certain variations in the velocity,
%in practice the velocity never is constant. In the signal model it would
%be related to the acceleration vector with the following vector state
%model:
%x(k+1)=Fx(k)+tau*a, where a is a 2x1 white gaussian acceleration noise
%vector, contempling the small changes in velocity in x and in y.
%x is a vector of dimensions 4x1 having both the position and velocity in
%both dimensions, x and y.
%F is the known transition matrix
%The following is the covariance matrix of the state equation, or the also
%known as process noise covariance matrix
Q = tau*V*tau';
%k = zeros(4,2,N);

```

```
%for n=1:N
 P_pred=F*P*F'+Q; %Predicted State covariance
 S=H*P_pred*H'+R; %Residual covariance
 %k(:,:,n)=P*H'*inv(S);
 k=P_pred*H'*inv(S);
 P_upd=P_pred-k*S*k'; %Updated State covariance
%end
```

Listing B.8: MATLAB code of ExtendedKalman.m

```
function [x_updEKF,P_updEKF,x_predEKF,P_predEKF,likelihoodEKF]=...
    ExtendedKalman(x_0j,P_0j,F,H,V,R,z,T)

[k_ekf,S,P_predEKF,P_updEKF]=genKalmancoefsekf(R,x_0j,P_0j,H,T,V);
%generate offline the Kalman coefficients and state covariance matrices
%prediction and updates

x_predEKF=F*x_0j;
error_ekf=z-H*x_predEKF;
%correction=k_ekf(:,:,x)*error_ekf;
correction=k_ekf*error_ekf;
x_updEKF=correction+F*x_0j;

if nargout>4
likelihoodEKF=(1/((2*pi)^0.5*det(S)^0.5))*...
    exp(-0.5*(z-x_predEKF([1 3],:))'*inv(S) * (z-x_predEKF([1 3],:)));
end
```

Listing B.9: MATLAB code of genKalmancoefsekf.m

```
function [k,S,P_pred,P_upd] = gen_Kalman_coefs_ekf(R,x,P,H,T,V)
%-----------------------------------------------------------------------
%The kalman coefficients are 4x1 which corresponds to Kx,Kv_x,Ky,Kv_y
%If we compute them off-line before kalman
%operation we will have a matrix with dimensions 4xN.
%C: Covariance 2x2 matrix of the observation white gaussian noise
%T: sampling time:
%N: number of kalman set of coeficients, one set for each kalman iteration
%sigma_a: variation in the velocity. We assume that between the (k ? 1)th
%and kth timestep the truck undergoes a constant acceleration of ak that
%is normally distributed, with mean 0 and standard deviation ?a
%-----------------------------------------------------------------------


tau=[(1/2)*T^2 0 0;T 0 0;0 (1/2)*T^2 0;0 T 0;0 0 T]; %--> This suppose to be the
```

```
%known matrix. It take into account certain variations in the velocity,
%in practice the velocity never is constant. In the signal model it would
%be related to the acceleration vector with the following vector state
%model:

%x(k+1)=Fx(k)+tau*a, where a is a 2x1 vector, taking into account the
%small acceleration values in x and in y --> a= [a_x;a_y] that makes small
%variations in velocity. x is a vector of dimensions 4x1 having both the
%position and velocity in both dimensions, x and y.
%A is the transition matrix

%Initial covariance matrix of the state equation
Q = tau*V*tau';
w=x(5,1);

if abs(w) ≤ 10^-12
    df=[1 T 0 0 -(1/2)*T^2*x(4);...
        0 1 0 0 -T*x(4);...
        0 0 1 T (1/2)*T^2*x(2);...
        0 0 0 1 T*x(2);...
        0 0 0 0 1];
else
  coswt = cos(w*T);
  coswto = 1-cos(w*T);
  coswtopw = -coswto/w;
  sinwt = sin(w*T );
  sinwtpw = sinwt/w;

  f_omega1=((coswt*T*x(2))/w)-((sinwt*x(2))/w^2)-...
    (sinwt*T*x(4))/w-((-1+coswt)*x(4))/w^2;

  f_omega2=-(sinwt*T*x(2))-(coswt*T*x(4));

  f_omega3=(sinwt*T*x(2))/w-((1-coswt)*x(2))/w^2+...
    (coswt*T*x(4))/w-(sinwt*x(4))/w^2;

  f_omega4=(coswt*T*x(2))-(sinwt*T*x(4));


   df=[1 sinwtpw   0 coswtopw f_omega1 ;...
     0 coswt      0 -sinwt f_omega2;...
     0 -coswtopw 1 sinwtpw f_omega3;...
     0 sinwt      0 coswt f_omega4; ...
     0   0   0   0   1];

end
%df is the jacobian matrix of the coordinated turn model
```

```
P_pred=df*P*df'+Q;  %state prediction covariance
S=H*P_pred*H'+R; %Residual covariance, or innovation covariance
k=P_pred*H'*inv(S); %Filter gain
P_upd=P_pred-k*S*k';%Update state covariance
```

Listing B.10: MATLAB code of immct.m

```
function [x_combupd,P_combupd,x_combpred,P_combpred,x_jk1,P_jk1,...
    mu_ijk1,x_updKF,x_updEKF,x_predKF,x_predEKF]=...
    imm_ct(mu_ij,p_ij,x_jk,P_jk,F,H,V,R,z,T)
%Interactive Multiple Model Estimator for the coordinated turn model
%(IMM-CT)
%We have two models or filters:
%Kalman filter with a process noise covariance V_KF stored in the cell V{1}
%and
%measurement noise covariance stored in the cell R{1}.
% Extended Kalman Filter with a process noise covariance V_EKF stored in
% V{2}and measurement noise covariance stored in the cell R{2}.
%The inputs in the algorithm IMM at time step k are:
%     - the r state vector and r state covariance matrix of previous time
%       step. r is the number of filters. These are x_jk and P_jk as a cell
%       array.
%     - the prior model probabilities for each model. It is a vector
%      with dimensions 1xr.
%     - The models transition probabilities p_ij which reflect the transition
%        probabilities of a Markov Chain. It has the dimensions rxr.
%     - The same inputs than the filters: Transitions matrices, process
%       noise covariance, measurement noise covariance, all of them are
%       cell arrays with dimensions 1xr.

%The outputs of the algorithm at time step k are:
%       - The updated combined state vector and state covariance: x_combupd
%       and P_combupd
%       - the updated state and covariance for each model: x_jk1 and P_jk1.
%       That is, the updated state vector and state covariance computed by
%       each of the different filters. Again they are organized in 1xr cell
%       array.
%       - the updated model probabilities: mu_ijk1 at next time step
%       - Also if it is needed, the combined predicted state vector and
%       state covariance: x_combpred and P_combpred

%We are going to work with 1xr cell arrays, where cell 1x1 we can stored
%whatever we want (vectors or matrices with any dimension).

%The IMM can be divided in three parts:
%   - Interaction/mixing: It computes the mixing probabilities from the
%   model probabilities, and the mixed initial condition for the filter
%   matched.
%   - Filters Prediction and Update (mode matched-filtering)
%   - Mode probability update and mixing probability calculation
%   - State estimate and covariance combination

nmodels = size(mu_ij,2);
dim_KF=4; %the state vector for the KF is 4 (x,vx,y,vy)
dim_EKF=5; %the state vector for the KF is 5 (x,vx,y,vy,w)
%-----------------------INTERACTION/MIXING-----------------------------
   % Normalizing factors for mixing probabilities
   c_j = zeros(1,nmodels);
    for j = 1:nmodels
        for i = 1:nmodels
            c_j(j) = c_j(j) + p_ij(i,j).*mu_ij(i);
        end
    end
   % Mixing probabilities
    MU_ij = zeros(nmodels,nmodels);
    for i = 1:nmodels
        for j = 1:nmodels
            MU_ij(i,j) = p_ij(i,j) * mu_ij(i) / c_j(j); %MU_ij is the
            %updated mixing probability from the previous model
            %probability mu_ij
        end
    end
    % Calculate the mixed state mean for each filter
    x_0j = cell(1,nmodels);
    x_0j{1}=zeros(dim_EKF,1);
    x_0j{2}=zeros(dim_EKF,1);
    for j = 1:nmodels
         ind=[1 2 3 4]';
        for i = 1:nmodels
            x_0j{j}(ind) = x_0j{j}(ind) + x_jk{i}(ind)*MU_ij(i,j);
            ind=[1 2 3 4 5]';
        end
    end

  % Calculate the mixed state covariance for each filter
    P_0j = cell(1,nmodels);
    P_0j{1}=zeros(dim_EKF,dim_EKF);
    P_0j{2}=zeros(dim_EKF,dim_EKF);

    for j = 1:nmodels
        ind=[1 2 3 4]';
         for i = 1:nmodels
P_0j{j}(ind,ind)=P_0j{j}(ind,ind)+ MU_ij(i,j)*(P_jk{i} + ...
    (x_jk{i}(ind)-x_0j{j}(ind))*(x_jk{i}(ind)-x_0j{j}(ind))');
           ind=[1 2 3 4 5]';
        end
    end
%----------------------END INTERACTION/MIXING-----------------------------
%----------------------MODE-MATCHED FILTERING-----------------------------
ind=[1 2 3 4]';
[x_updKF,P_updKF,x_predKF,P_predKF,likelihood_KF]=...
    Kalman(x_0j{1}(ind),P_0j{1}(ind,ind),F{1},H{1},V{1},R,z,T);
 ind=[1 2 3 4 5]';
[x_updEKF,P_updEKF,x_predEKF,P_predEKF,likelihoodEKF]=...
    ExtendedKalman(x_0j{2}(ind),P_0j{2}(ind,ind),F{2},H{2},V{2},R,z,T);

 %----Combined Prediction of the state vector and the state covariance-----
likelihood = [likelihood_KF likelihoodEKF];
  % Predicted state mean
     x_combpred = [x_predKF;0]*mu_ij(1)+x_predEKF*mu_ij(2);
 % Predicted state covariance
 P_predKF1=zeros(dim_EKF,dim_EKF);
 P_predKF1(1:length(P_predKF),1:length(P_predKF))=P_predKF;
 P_predKF= P_predKF1;
 P_combpred=mu_ij(1)*(P_predKF+([x_predKF;0]-x_combpred)*...
     ([x_predKF;0]-x_combpred)')+...
      mu_ij(2)*(P_predEKF+(x_predEKF-x_combpred)*(x_predEKF-x_combpred)');

 c = sum(likelihood.*c_j);
 mu_ijk1 = c_j.*likelihood/c; % Update the model probabilities

 %--------Combined Update of the state vector and the state covariance---
 x_jk1=cell(1,nmodels);
  x_jk1{1}= [x_updKF;0];
 x_jk1{2}=x_updEKF;
 P_jk1=cell(1,nmodels);
 P_jk1{1}=P_updKF;
 P_jk1{2}=P_updEKF;

 x_combupd= zeros(dim_EKF,1);
 P_combupd=zeros(dim_KF,dim_KF);

 % Updated state mean
    for j = 1:nmodels
       x_combupd = x_combupd + mu_ijk1(j)*x_jk1{j};
    end
    x_combupd1=x_combupd(1:dim_KF);
    ind=[1 2 3 4]';
    % Updated state covariance
    for j = 1:nmodels
```

```
P_combupd=P_combupd+ mu_ijk1(j)*(P_jk1{j}+(x_jk1{j}(ind)-x_combupd1)*...
    (x_jk1{j}(ind)-x_combupd1)');
     if j==1
         P_combupd1=zeros(dim_EKF,dim_EKF);
         P_combupd1(1:length(P_combupd),1:length(P_combupd))=P_combupd;
         P_combupd=P_combupd1;
         x_combupd1=x_combupd;
          ind=[1 2 3 4 5]';
     end
    end

%
```

Appendix C

ARID Navigator Java Code

The ARID NAVIGATOR is developed in Java code using the Java IDE Netbeans. The set of Java codes used for the navigator development are:

- Rssidemo.java: this is the main class. This code is able to recollect the frames coming from the base sensor node via serial USB port. Then the received power source is read with a tinyos function from the received packet. Then the positioning computation is done in this code. The position computation is send to the Window class. This code is mainly developed by Albert Bel who also have participated in the XALOC project.
- Window.java: this class is the application window of the navigator having all the swing Java objects (buttons, checkboxes, layeredpanes and so on). The purpose of this class is basically to draw on a map the parkings icons as well as the user's position. The map can be downloaded from internet or loaded locally depending of the cold start and warm start state buttons. To draw the user's position icons and the parkings icons the jLayeredPane class is used.
- ParkingSensor.java: this class is used to create objects that store the UTM coordinates of every parking sensor. This class has been useful to create a list of parkings, with objects instantiated from this class in each position of the parking list.
- KalmanFilter.java: this class implements the one dimension Kalman filter.

Listing C.1: JAVA code of RssiDemo.java

```java

package mymaps;

```

```
/*
 * "Copyright (c) 2005 The Regents of the University  of California.
 * All rights reserved.
 *
 * Permission to use, copy, modify, and distribute this software and
 * its documentation for any purpose, without fee, and without written
 * agreement is hereby granted, provided that the above copyright
 * notice, the following two paragraphs and the author appear in all
 * copies of this software.
 *
 * IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY
 * PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL
 * DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS
 * DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN
 * ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED HEREUNDER IS
 * ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATION TO
 * PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS."
 *
 */

/*
 * This is a modified version of TestSerial.java, from apps/tests/TestSerial
 * from TinyOS 2.x (www.tinyos.net)
 */

/**
 * Java-side application for testing the RSSI demo
 *
 * @author Phil Levis <pal@cs.berkeley.edu>
 * @author Dimas Abreu Dutra <dimas@dcc.ufmb.br>
 * @date April 11 2008
 */


import net.tinyos.message.*;
import net.tinyos.util.*;
import net.tinyos.packet.*;
import java.util.Arrays;
import java.io.*;
import javax.swing.Timer;
import java.awt.event.*;

public class RssiDemo implements MessageListener {

  private MoteIF moteIF;
  public Window win;

  public RssiDemo(MoteIF moteIF) {
    this.moteIF = moteIF;
    this.moteIF.registerListener(new RssiMsg(), this);
     win = new Window();
     win.setVisible(true);
 }
  // variables de control de programa
  public int num_places=18; //nodes totals
  public double dist_threshold=2.000; //threshold distancia
  public String string_pot ="potencia3.txt";
  public String string_coop="valor_coop3.txt";
  public String string_pos="coord_projectades3.txt";
  public String string_pos_klmn="coord_kalman3.txt";
  public int tipus_window=2; // 1-> lectura 2-> dBm 3->lineal
  public double gamma1=5;
  public double gamma2=12;
  public double gamma3 = 18;
  public double velocidadAlta=2; // threshold de velocidad en m/s
  public double velocidadBaja=0.75; // threshold de velocidad en m/s
  public double gamma=0;
  //public double gamma=5;//dbm o lectura P*75%-> 2.25 P*50%-> 3 P*75%-> 6
  //public double gamma=0.277;//factor de reduccio P*75%-> 0.115
  //P*50%-> 0.277 P*25%-> 0.555
  public int tipus_lectura=2; // 1-> lectura directa 2->dBm 3-> watts
  public int tipus_cooperacio=2; //1-> cooperants fixos 2->threshold
  public int tipus_potencia=4; // 1-> lectura 2-> lectura WS 3-> dBm iris 4->dBm
  //WS 5-> linial iris 6-> linial WS
  public double threshold = 86; //threshold potencia
  public double x_mob=0, R=25, V=3,v=(double)0*1000.0/3600.0; //R is the
 // measurement noise covariance and V is the acceleration random value variance
  public int num_coop=4; //numero nodes cooperants
  public int delay_seconds=2500; //every 2.5 sg we perform positioning
  //variables de control del programa

  public double suma=0;
  public double suma_inv=0;
  public int k=0, num_iterations=0;//num_iterations without position measurements
  public double x[] = new double[num_places];
  public double y[] = new double[num_places];
  public double x_mobil=0.0;
  public double y_mobil=0.0;
  public double x0=0.0;
  public double y0=0.0;
  public double x1=0.0;
  public double y1=0.0;
  public double pendent=0.0;
  public double b=0.0;
  public double a1=0.0;
  public double a2=1;
  public double d_kalman = 0.0;
  public double step=-10.0;public int kk=0;public int cont_coop=0;
  public double beta= 0;
  public double posicions_usades[] = new double[num_places];
  public double potencia[] = new double[num_places];
  public double pow_rx[] = new double[num_places];
  public double distancia=0.0;
  public float posCordinates[] = new float[2];
  public double coord_x[] = new double [num_places];
  public double coord_y[] = new double [num_places];
  public int rx_msg=0;

  public int mesura=1, temp=1;

  public double[] updpos; //updated position and velocity in x ith kalman
  public double y_mob=0,x_mob_ant=0,y_mob_ant=0,x_mob_pant=0,y_mob_pant=0;
 public KalmanFilter KF= new KalmanFilter();


  public void messageReceived(int to, Message message) {
   RssiMsg msg = (RssiMsg) message;
   int source = message.getSerialPacket().get_header_src();
   if(source≥1 && source≤18){
   rx_msg++;
    switch(tipus_potencia){
        case 1: //lectura directa
            potencia[source-1]= msg.get_rssi();
            pow_rx[source-1]=msg.get_rssi();
            break;
        case 2: //lectura directa WS
            potencia[source-1]= msg.get_rssi()*3.1;
            pow_rx[source-1]=msg.get_rssi()*3.1;
            break;
        case 3: //dBm iris
            potencia[source-1]= (msg.get_rssi()-91)*(-1);
            pow_rx[source-1]=(msg.get_rssi()-91)*(-1);
            break;
        case 4: //dBm WS
           try{
               potencia[source-1]= (msg.get_rssi()*3.1-98)*(-1);

```

```
                pow_rx[source-1]=(msg.get_rssi()*3.1-98)*(-1);
                 //System.out.println("source good: "+ String.valueOf(source));
                  //  System.out.println("msg type good"+ String.valueOf(msg.AM_TYPE));
                }catch(Exception e){
                    System.out.println("source: "+ String.valueOf(source));
                    System.out.println("msg type"+ String.valueOf(msg.AM_TYPE));
                }
                break;
            case 5: // lineal iris
                potencia[source-1]= Math.pow(10,(((msg.get_rssi()-91)-30)/10.0));
                pow_rx[source-1]=Math.pow(10,(((msg.get_rssi()-91)-30)/10.0));
                break;
            case 6: // lineal WS
                potencia[source-1]= Math.pow(10,(((msg.get_rssi()*3.1-98)-30)/10.0));
                pow_rx[source-1]=Math.pow(10,(((msg.get_rssi()*3.1-98)-30)/10.0));
                break;
        }

        try{
            FileWriter fr = new FileWriter(string_pot,true);
            PrintWriter salida=new PrintWriter(fr);
            salida.println(source + " " + potencia[source-1]);
            salida.close();
        }

        catch(java.io.IOException ioex){}

      }
        coord_x[0]=(double)426064.21;
        coord_y[0]=(double)4594717.32;
        coord_x[1]=(double)426070.39;
        coord_y[1]=(double)4594724.20;
        coord_x[2]=(double)426060.76;
        coord_y[2]=(double)4594720.39;
        coord_x[3]=(double)426066.66;
        coord_y[3]=(double)4594727.57;
        coord_x[4]=(double)426057.03;
        coord_y[4]=(double)4594723.72;
        coord_x[5]=(double)426062.95;
        coord_y[5]=(double)4594730.92;
        coord_x[6]=(double)426053.30;
        coord_y[6]=(double)4594727.05;
        coord_x[7]=(double)426059.24;
        coord_y[7]=(double)4594734.28;
        coord_x[8]=(double)426049.56;
        coord_y[8]=(double)4594730.38;
        coord_x[9]=(double)426055.53;
```

```
    coord_y[9]=(double)4594737.63;
    coord_x[10]=(double)426045.83;
    coord_y[10]=(double)4594733.70;
    coord_x[11]=(double)426051.82;
    coord_y[11]=(double)4594740.98;
    coord_x[12]=(double)426042.10;
    coord_y[12]=(double)4594737.03;
    coord_x[13]=(double)426048.11;
    coord_y[13]=(double)4594744.33;
    coord_x[14]=(double)426038.37;
    coord_y[14]=(double)4594740.36;
    coord_x[15]=(double)426044.40;
    coord_y[15]=(double)4594747.69;
    coord_x[16]=(double)426034.63;
    coord_y[16]=(double)4594743.68;
    coord_x[17]=(double)426040.69;
    coord_y[17]=(double)4594751.04;
    /*coord_x[18]=(double)426030.9;
    coord_y[18]=(double)4594747.01;
    coord_x[19]=(double)426036.98;
    coord_y[19]=(double)4594754.39;*/
    x1=(double) (coord_x[0]+coord_x[1])/2.0;
    y1=(double) (coord_y[0]+coord_y[1])/2.0;
    x0=(double) (coord_x[16]+coord_x[17])/2.0;
    y0=(double) (coord_y[16]+coord_y[17])/2.0;
    pendent=((double) (y1-y0)/(x1-x0));
    b=y0-(x0*pendent);
    a1=-pendent;
    //System.out.println("pendent: "+pendent+"b: "+b+"a1: "+a1+"y1: "+y1);
    distancia=Math.sqrt((Math.pow((x1-x0),2))+(Math.pow((y1-y0),2)));
     if(source>=1 && source<=18){
    System.out.println("Rssi " + source + ": " + potencia[source-1]);
     }

    ActionListener temporitzador=new ActionListener()
    {
           public void actionPerformed(ActionEvent evento) {
               //System.out.println(" rx_msg: "+rx_msg);
                 Arrays.sort(potencia);
                 cont_coop=0;
                 int cont=0;
                 kk=0;
                 x_mobil=0;
                 y_mobil=0;
                 suma=0;
                 suma_inv=0;
                 for (k=0;k<num_places;k++)
```

```
{
    switch (tipus_lectura){
        case 1: //lectura
            switch (tipus_cooperacio){
                case 1: //cooperants fixos
               if (potencia[num_places-k-1]>0){
                 while (pow_rx[kk]!=potencia[num_places-k-1]){
                            kk++;
                        }
                        x[cont_coop]=coord_x[kk];
                        y[cont_coop]=coord_y[kk];
                        cont_coop++;
                        posicions_usades[kk]=1;
                        kk=0;
                    }break;
                case 2: //power threshold
                    if (potencia[num_places-k-1]>threshold){
                  while (pow_rx[kk]!=potencia[num_places-k-1]){
                            kk++;
                        }
                        x[cont_coop]=coord_x[kk];
                        y[cont_coop]=coord_y[kk];
                        cont_coop++;
                        posicions_usades[kk]=1;
                        kk=0;
                    }break;
            }break;
        case 2: //dBm watts
            if(potencia[k]==0){
                cont++;
            }
            switch (tipus_cooperacio){
                case 1: //cooperants fixos
                    if (potencia[k]>0){
                        while (pow_rx[kk]!=potencia[k]){
                            kk++;
                        }
                        x[cont_coop]=coord_x[kk];
                        y[cont_coop]=coord_y[kk];
                        cont_coop++;
                        posicions_usades[kk]=1;
                        kk=0;
                    }break;
                case 2: //power threshold
                 if (potencia[k]<threshold && potencia[k]>0){
                        while (pow_rx[kk]!=potencia[k]){
                            kk++;
```

```
                                          }
                                          x[cont_coop]=coord_x[kk];
                                          y[cont_coop]=coord_y[kk];
                                          cont_coop++;
                                          posicions_usades[kk]=1;
                                          kk=0;
                                      }break;
                              }break;
                              case 3: //dBm watts
                              if(potencia[k]==0){
                                  cont++;
                              }
                              switch (tipus_cooperacio){
                                  case 1: //cooperants fixos
                                      if (potencia[k]>0){
                                          while (pow_rx[kk]!=potencia[k]){
                                              kk++;
                                          }
                                          x[cont_coop]=coord_x[kk];
                                          y[cont_coop]=coord_y[kk];
                                          cont_coop++;
                                          posicions_usades[kk]=1;
                                          kk=0;
                                      }break;
                                  case 2: //power threshold
                                  if (potencia[k]<threshold && potencia[k]>0){
                                          while (pow_rx[kk]!=potencia[k]){
                                              kk++;
                                          }
                                          x[cont_coop]=coord_x[kk];
                                          y[cont_coop]=coord_y[kk];
                                          cont_coop++;
                                          posicions_usades[kk]=1;
                                          kk=0;
                                      }break;
                              }break;

                         }
                 }

                   try{
                         FileWriter ffr = new FileWriter(string_coop,true);
                         PrintWriter sortida_pos=new PrintWriter(ffr);
     sortida_pos.println(posicions_usades[0]+ " " +posicions_usades[1]+ " " +
             posicions_usades[2]+" " +posicions_usades[3]+" " +posicions_usades[4]+
             " " +posicions_usades[5]+" " +posicions_usades[6]+" " +
             posicions_usades[7]+" " +posicions_usades[8]+" " +posicions_usades[9]+
             " " +posicions_usades[10]+" " +posicions_usades[11]);
                         sortida_pos.close();
                 }
                 catch(java.io.IOException ioex){}

                  if (cont_coop==0 && rx_msg!=0)
                  {
                      switch(tipus_lectura){
                          case 1: //lectura
                              kk=0;
                              while (pow_rx[kk]!=potencia[num_places-1]){
                                  kk++;
                              }
                              x[cont_coop]=coord_x[kk];
                              y[cont_coop]=coord_y[kk];
                              posicions_usades[kk]=1;
                              kk=0;
                              break;
                          case 2: //dBm
                              k=0;
                              while (potencia[k]==0){
                                  k++;
                              }
                              while (potencia[k]!=pow_rx[kk]){
                                  kk++;
                              }
                              x[cont_coop]=coord_x[kk];
                              y[cont_coop]=coord_y[kk];
                              posicions_usades[kk]=1;
                              kk=0;
                              break;

                      }
                      cont_coop=1;
                  }
                  if (cont_coop!=0)
                  {
                      //System.out.println(" cont_coop: "+cont_coop);
                      if (num_coop<cont_coop){
                          cont_coop=num_coop;
                      }

                     switch (tipus_lectura){
                          case 1: //lectura directa
                              for (k=0;k<cont_coop;k++){
                                  suma=suma+potencia[num_places-k-1];
                              }break;
```

```
                        case 2: //dBm watts
                            for (k=cont;k<cont+cont_coop;k++){
                       //System.out.println("k: "+k+" cont_coop: "+cont_coop);
                                suma=suma+potencia[k];
                                suma_inv=suma_inv+(1.0/potencia[k]);
                            }break;
                       case 3: //watts
                           for (k=cont;k<cont+cont_coop;k++){
                       //System.out.println("k: "+k+" cont_coop: "+cont_coop);
                              suma=suma+potencia[k];

                            }break;

                    }

                    switch (tipus_lectura){
                        case 1: //lectura directa
                            for (k=0;k<cont_coop;k++){
                       beta=((double)potencia[num_places-k-1])/((double)suma);
                                x_mobil=(double)x_mobil+beta*x[k];
                                y_mobil=(double)y_mobil+beta*y[k];
                            }break;
                        case 2: //dBm watts
                            for (k=cont;k<cont+cont_coop;k++){
                       beta=((double)suma/potencia[k])/((double)suma*suma_inv);
                                x_mobil=(double)x_mobil+beta*x[k-cont];
                                y_mobil=(double)y_mobil+beta*y[k-cont];

                            }break;
                        case 3:
                            for (k=cont;k<cont+cont_coop;k++){
                                beta=((double)potencia[k])/((double)suma);
                                x_mobil=(double)x_mobil+beta*x[k-cont];
                                y_mobil=(double)y_mobil+beta*y[k-cont];

                            }break;
                    }

                   x_mob=(double) (x_mobil+(((b-(x_mobil*a1)-(y_mobil*a2))/
                           ((Math.pow(a1,2)+Math.pow(a2,2))))*a1));
                   y_mob=(double) (y_mobil+(((b-(x_mobil*a1)-(y_mobil*a2))/
                           (Math.pow(a1,2)+Math.pow(a2,2)))*a2));
                }

            try
            {
                FileWriter fww = new FileWriter(string_pos,true);
```

```
                    PrintWriter sortidab=new PrintWriter(fww);
                    sortidab.println(x_mob + " " + y_mob);
                    sortidab.close();
                }
                catch(java.io.IOException ioex){}
                if(rx_msg==0){ //if there is not measurements every 2.5 sg
                   num_iterations++;
                }
                else{
                    num_iterations=0;
                }
                if (num_iterations>2){
                    x_mob=0;
                    y_mob=0;
                    mesura=1;
                    for (k=0;k<num_places;k++){
                        potencia[k]=0;
                        pow_rx[k]=0;
                    }
                 }
                if (x_mob!=0 & y_mob!=0){
           d_kalman=Math.sqrt(Math.pow((x_mob-x0),2)+(Math.pow((y_mob-y0),2)));
                    if (x_mob<x0)
                    {
                        d_kalman=-d_kalman;
                        v=-v;
                    }
                    else if(x_mob==x0)
                    {
                        if (y_mob<y0)
                        {
                            d_kalman=-d_kalman;
                            v=-v;
                        }
                    }

                    if ((mesura==1) || (num_iterations>2)) {
             KF = new KalmanFilter(d_kalman,v,(double)delay_seconds/1000.0,R);
                        mesura=0;
                        num_iterations=0;
                    }

                    updpos=KF.kalman(KF.KFestimates, V,R,d_kalman);
                    x_mob=x0+updpos[0]*(x1-x0)/distancia;
                    y_mob=y0+updpos[0]*(y1-y0)/distancia;
                }
             if (Math.sqrt(Math.pow((x_mob-x_mob_ant),2)+
                         Math.pow((y_mob-y_mob_ant),2))<dist_threshold)
            {
                x_mob_pant=x_mob_ant;
                y_mob_pant=y_mob_ant;
            }
            else if (x_mob!=0)
            {
                x_mob_pant=x_mob;
                y_mob_pant=y_mob;
            }
            y_mob_ant=y_mob_pant;
            x_mob_ant=x_mob_pant;

            step++;

            System.out.println("Posicion despues KF: coordenada x: " +
                    x_mob  +  "  coordenada y: " + y_mob);
            KF.KFestimates[0]=updpos[0];
            KF.KFestimates[1]=updpos[1];
            if ((Math.abs(updpos[1]) >velocidadBaja)&&
                    (Math.abs(updpos[1]) <velocidadAlta)){
                gamma = gamma2;
            }
            else if (Math.abs(updpos[1]) >velocidadAlta){
                gamma=gamma3;
            }
            else{
                 gamma=gamma1;
            }
            try
            {
                FileWriter fww = new FileWriter(string_pos_klmn,true);
                PrintWriter sortidab=new PrintWriter(fww);
                sortidab.println(x_mob_pant + " " + y_mob_pant);
                sortidab.close();
             }
             catch(java.io.IOException ioex){}

             try
             {
            if ((!win.jButton1.isEnabled()) && (!win.jButton3.isEnabled()))
                 {
              win.startNavigator(x_mob_pant, y_mob_pant, pow_rx, updpos[1]);

                 }
             }catch (Exception e)
             {
```

```
                        System.err.println("objeto window no existe");
                    }
                  switch (tipus_window){
                      case 1: //lectura directa
                          for (k=0;k<num_places;k++)
                          {
                              if (pow_rx[k]!=0){
                                  pow_rx[k]=pow_rx[k]-gamma;
                              }
                              potencia[k]=pow_rx[k];
                              posicions_usades[k]=0;
                          }break;
                      case 2: //dBm
                          for (k=0;k<num_places;k++)
                          {
                              if (pow_rx[k]!=0){
                                  pow_rx[k]=pow_rx[k]+gamma;
                              }
                              potencia[k]=pow_rx[k];
                              posicions_usades[k]=0;
                          }break;
                      case 3: //lineal
                          for (k=0;k<num_places;k++)
                          {
                        pow_rx[k]=pow_rx[k]*Math.exp(-gamma*delay_seconds/1000.0);
                              potencia[k]=pow_rx[k];
                              posicions_usades[k]=0;
                          }break;
                  }
                  rx_msg=0;

              }

         };
              if (temp==1){
                  new Timer(delay_seconds, temporitzador).start();
                  temp=0;
              }

     }
     public float[] getcoordinates(){
         return posCordinates;
     }
     private static void usage() {
       System.err.println("usage: RssiDemo [-comm <source>]");
     }

```

```
  public static void main(String[] args) throws Exception {
    String source = null;
    if (args.length == 2) {
      if (!args[0].equals("-comm")) {
    usage();
    System.exit(1);
      }
      source = args[1];
    }
    else if (args.length != 0) {
      usage();
      System.exit(1);
    }

    PhoenixSource phoenix;

    if (source == null) {
      phoenix = BuildSource.makePhoenix(PrintStreamMessenger.err);
    }
    else {
      phoenix = BuildSource.makePhoenix(source, PrintStreamMessenger.err);
    }

    MoteIF mif = new MoteIF(phoenix);

    RssiDemo serial = new RssiDemo(mif);


  }


}
```

Listing C.2: JAVA code of Window.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package mymaps;

/**
 *
 * @author spcomnav
 */
```

```

import java.awt.event.ActionEvent;
import java.net.URL;
import java.net.MalformedURLException;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;

import org.jposition.CharacterInvalidException;
import org.jposition.Colors;
import org.jposition.Coordinate;
import org.jposition.CoordinateRangeException;
import org.jposition.Dimension;
import org.jposition.DimensionRangeException;
import org.jposition.MapTypeException;
import org.jposition.Marker;
import org.jposition.MarkersMap;
import org.jposition.ZoomRangeException;
import javax.swing.*;
import java.awt.*;
import java.util.List;
import edu.uab.geoloc.DeviceInfo;
import edu.uab.geoloc.ServerWrapper;
import sun.audio.*;
import java.io.*;


public class Window extends javax.swing.JFrame{

   //Variabe declaration
     public String[] MapURL;
   //public String mapurl;
    public MarkersMap mapa;
    public ImageIcon image, marker, freeParking,nonfreeParking;
    public JLabel markerLabel, markerAnchor1,markerAnchor2, markerAnchor3,
            markerAnchor4, markerAnchor5, markerAnchor6;
    public int PosCalculated, refresh, zoom, ParkingIconSizeWidth,
            ParkingIconSizeHeight, ini=0;
    public ArrayList<JLabel> markers;
    public List<ParkingSensors> parkings;
      public List<JLabel> markerAnchors;
    public List<DeviceInfo> li;
    public double centralLatToUTM,centralLonToUTM, centralLonToUTMproof,
            centralLatToUTMproof, x_mob, y_mob,mpp=0,centralLat=0,
            centralLon=0, cpx,cpy, contador=0.0;
    public boolean coldstart, warmstart, audio;
    public InputStream in=null;
```

```java
    public AudioStream as=null;
    public long[] indexes;


       private javax.swing.JTextField NewCord;
    private javax.swing.JTextField centralcord;
    public javax.swing.JButton jButton1;
    public javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLayeredPane jLayeredPane1;
    private javax.swing.JLabel labelImage;
    private javax.swing.JCheckBox jCheckBox1;


    /** Creates new form Window */
    public Window() {
        //initComponents();
        jLabel1 = new javax.swing.JLabel();
        centralcord = new javax.swing.JTextField();
        jButton1 = new javax.swing.JButton();
        jLabel3 = new javax.swing.JLabel();
        NewCord = new javax.swing.JTextField();
        jLayeredPane1 = new javax.swing.JLayeredPane();
        labelImage = new javax.swing.JLabel();
        jButton3 = new javax.swing.JButton();
        jLabel2 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jButton4 = new javax.swing.JButton();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jCheckBox1 = new javax.swing.JCheckBox();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Xaloc Project. Car Positioning ");

        jLabel1.setText("Central cordinate");

        centralcord.setEnabled(false);

        jButton1.setText("Cold start");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jLabel3.setText("Measured cordinate");

        NewCord.setEnabled(false);

        labelImage.setPreferredSize(new java.awt.Dimension(620, 500));
        labelImage.setBounds(0, 10, 620, 500);
        jLayeredPane1.add(labelImage, javax.swing.JLayeredPane.DEFAULT_LAYER);

        jButton3.setText("Warm start");
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });

        jLabel2.setFont(new java.awt.Font("Tahoma", 1, 26));
        jLabel2.setText("ARID NAVIGATOR");

        jLabel4.setFont(new java.awt.Font("Tahoma", 1, 12));
        jLabel4.setText("Â© 2010 SPCOMNAV - XALOC PROJECT");

        jButton4.setText("Refresh");
        jButton4.setEnabled(false);
        jButton4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton4ActionPerformed(evt);
            }
        });
         jLabel5.setText("");
        jLabel7.setText("");
        jLabel6.setText("");
         jCheckBox1.setText("Audio Off/On");
         jCheckBox1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jCheckBox1ActionPerformed(evt);
            }
        });
  javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
  getContentPane().setLayout(layout);
 layout.setHorizontalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup()
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup()
.addContainerGap()
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup()
.addComponent(jLayeredPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 620,
        javax.swing.GroupLayout.PREFERRED_SIZE)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup()
.addGap(70, 70, 70)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
 .addComponent(jLabel7)
 .addComponent(jCheckBox1, javax.swing.GroupLayout.PREFERRED_SIZE, 100,
 javax.swing.GroupLayout.PREFERRED_SIZE)
 .addComponent(jButton4)
 .addComponent(jButton3)
 .addComponent(jButton1)))
 .addGroup(layout.createSequentialGroup()
 .addGap(18, 18, 18)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
 .addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED_SIZE, 343,
        javax.swing.GroupLayout.PREFERRED_SIZE)
 .addComponent(jLabel5, javax.swing.GroupLayout.DEFAULT_SIZE, 343,
 Short.MAX_VALUE)))))
 .addGroup(layout.createSequentialGroup()
 .addGap(48, 48, 48)
 .addComponent(jLabel3)
 .addGap(18, 18, 18)
 .addComponent(NewCord, javax.swing.GroupLayout.PREFERRED_SIZE, 160,
 javax.swing.GroupLayout.PREFERRED_SIZE)
 .addGap(18, 18, 18)
 .addComponent(jLabel1)
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
 .addComponent(centralcord, javax.swing.GroupLayout.PREFERRED_SIZE, 163,
 javax.swing.GroupLayout.PREFERRED_SIZE)))
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
 layout.createSequentialGroup()
  .addContainerGap(415, Short.MAX_VALUE)
  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
  .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 265,
  javax.swing.GroupLayout.PREFERRED_SIZE)
  .addGroup(layout.createSequentialGroup()
  .addGap(10, 10, 10)
  .addComponent(jLabel4)))
  .addGap(311, 311, 311)))
  .addGap(50, 50, 50))
  );
  layout.setVerticalGroup(
  layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
  .addGroup(layout.createSequentialGroup()
  .addContainerGap()
  .addComponent(jLabel2)
  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
  .addComponent(jLabel4)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
  .addGroup(layout.createSequentialGroup()
  .addGap(23, 23, 23)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
  .addComponent(jLabel3)
  .addComponent(NewCord, javax.swing.GroupLayout.PREFERRED_SIZE,
  javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
  .addComponent(jLabel1)
  .addComponent(centralcord, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
   .addGroup(layout.createSequentialGroup()
  .addGap(18, 18, 18)
  .addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
          javax.swing.GroupLayout.PREFERRED_SIZE)))
  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
  .addGroup(layout.createSequentialGroup()
  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
  .addComponent(jLayeredPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 500,
          javax.swing.GroupLayout.PREFERRED_SIZE))
  .addGroup(layout.createSequentialGroup()
  .addGap(18, 18, 18)
  .addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED_SIZE, 28,
          javax.swing.GroupLayout.PREFERRED_SIZE)
  .addGap(158, 158, 158)
  .addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE, 22,
          javax.swing.GroupLayout.PREFERRED_SIZE)
  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
  .addComponent(jButton1)
  .addGap(18, 18, 18)
  .addComponent(jButton3)
  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
  .addComponent(jButton4)
  .addGap(18, 18, 18)
  .addComponent(jCheckBox1)))
  .addContainerGap(28, Short.MAX_VALUE))
   );
```

```java

        jLabel1.getAccessibleContext().setAccessibleName("Direccion");

        pack();
        MyIniComponents();


    }
    public void MyIniComponents(){
          zoom=19;
         PosCalculated=0;
         refresh=0;
         cpx=this.labelImage.getWidth()/2; // central pixel in x
         cpy=this.labelImage.getHeight()/2; //central pixel in y

         //free Parking icon
  freeParking = new ImageIcon(getClass().getResource("/mymaps/pBlau.gif"));
  freeParking = new ImageIcon(freeParking.getImage().getScaledInstance(20,
          20, Image.SCALE_SMOOTH));

        //non free Parking icon
 nonfreeParking = new ImageIcon(getClass().getResource("/mymaps/pVermell.gif"));
       nonfreeParking = new ImageIcon(nonfreeParking.getImage().
               getScaledInstance(20, 20, Image.SCALE_SMOOTH));

       ParkingIconSizeWidth= freeParking.getIconWidth();
       ParkingIconSizeHeight= freeParking.getIconHeight();

     //Adding parking Sensors
        parkings= new ArrayList<ParkingSensors>();
       parkings.add(new ParkingSensors(426064.21, 4594717.32)); //anchor 1
       parkings.add(new ParkingSensors(426070.39, 4594724.20)); //anchor 2
       parkings.add(new ParkingSensors(426060.76, 4594720.39)); //anchor 3
       parkings.add(new ParkingSensors(426066.66, 4594727.57)); //anchor 4
       parkings.add(new ParkingSensors(426057.03, 4594723.72)); //anchor 5
       parkings.add(new ParkingSensors(426062.95, 4594730.92)); //anchor 6
       parkings.add(new ParkingSensors(426053.30, 4594727.05)); //anchor 7
       parkings.add(new ParkingSensors(426059.24, 4594734.28)); //anchor 8
       parkings.add(new ParkingSensors(426049.56, 4594730.38)); //anchor 9
       parkings.add(new ParkingSensors(426055.53, 4594737.63)); //anchor 10
       parkings.add(new ParkingSensors(426045.83, 4594733.70)); //anchor 11
       parkings.add(new ParkingSensors(426051.82, 4594740.98)); //anchor 12
       //added
       parkings.add(new ParkingSensors(426042.10,4594737.03)); //anchor 13
       parkings.add(new ParkingSensors(426048.11,4594744.33)); //anchor 14
       parkings.add(new ParkingSensors(426038.37,4594740.36)); //anchor 15
       parkings.add(new ParkingSensors(426044.40,4594747.69)); //anchor 16
```

```
        parkings.add(new ParkingSensors(426034.63,4594743.68)); //anchor 17
        parkings.add(new ParkingSensors(426040.69,4594751.04));  //anchor 18
        parkings.add(new ParkingSensors(426030.90,4594747.01)); //anchor 19
        parkings.add(new ParkingSensors(426036.98,4594754.39)); //anchor 20

        li = new ArrayList <DeviceInfo>();
        markerAnchors=new ArrayList<JLabel>();
        audio=false;
     }

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
         //cold start button
        coldstart=true;
         warmstart=false;
         this.jButton1.setEnabled(false);
         this.jButton3.setEnabled(false);
         this.jButton4.setEnabled(true);
         if (this.refresh==1){
             this.refresh=0;
         }
            checkParkingState(); //another timer with audio
    }

   private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
         // TODO add your handling code here:
        //Warm Start button
         coldstart=false;
         warmstart=true;
         this.jButton1.setEnabled(false);
         this.jButton3.setEnabled(false);
         checkParkingState(); //with audio
    }
    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
         // TODO add your handling code here:
         //Refresh button
     PosCalculated=0;
    refresh=1;
    jLayeredPane1.removeAll();
    jLayeredPane1.validate();
    repaint();
    this.labelImage.setIcon(image);
    jLayeredPane1.add(this.labelImage);
     }
    private void jCheckBox1ActionPerformed(java.awt.event.ActionEvent evt) {
         // TODO add your handling code here:
         if (!jCheckBox1.isSelected()){
             audio=false;
         }
         else if (jCheckBox1.isSelected()){
             audio=true;
         }
     }
     @SuppressWarnings("static-access")

     public void startNavigator(double UTMEasting, double UTMNorthing,
             double[] potencia, double vkalman){
         //System.out.println("iniciando");
              double diffxx0, diffyy0,measuredLat, measuredLon;
              int pmpx,pmpy;

              double[] centralGeoCordinates, measuredGeoCordinates;

               PosCalculated=PosCalculated+1;
               y_mob=(double)UTMNorthing;
               x_mob=(double)UTMEasting;

           //pmpx: pixels of the measured easting in x
           //pmpy: pixels of the measured northing in y
           //diffxx0: x-x0, diffyy0: y-y0; (x0,y0): are the central cordinates
           //of the map in UTM and (x,y) are the
           //measured coordinates

             String newUTM, mapurl;
            Coordinateconversion cordinates = new Coordinateconversion();
             Coordinate coordenada1 = null;

              if (this.refresh==1 & coldstart){
                  centralLatToUTM=y_mob;
                 centralLonToUTM=x_mob;
             }
             if (PosCalculated==1)
             {
                 if (coldstart) {
                 centralLatToUTM=y_mob;
                 centralLonToUTM=x_mob;
                 }
                 else if (warmstart){
                 //centralLatToUTM=4594716.87;
                 centralLatToUTM=4594716.25;
                 //centralLonToUTM=426068.23;
                 centralLonToUTM=426068.65;
                 }
                 newUTM="31 T "+String.valueOf(centralLonToUTM)+" "+
                         String.valueOf(centralLatToUTM);
                 centralGeoCordinates=cordinates.utm2LatLon(newUTM);
                 centralLat=centralGeoCordinates[0];
                 centralLon=centralGeoCordinates[1];
           this.centralcord.setText(String.valueOf(centralLat).
                 substring(0, 8)+" "+String.valueOf(centralLon).substring(0,8));
                 this.centralcord.setEnabled(true);
                 //mpp: meters per pixel
                  mpp= this.getmpp();

             }
             else if (PosCalculated>1){
               jLayeredPane1.remove(markerLabel);
               jLayeredPane1.validate();
               repaint();
             }

             newUTM="31 T "+String.valueOf(x_mob)+" "+String.valueOf(y_mob);
             measuredGeoCordinates=cordinates.utm2LatLon(newUTM);
             measuredLat= measuredGeoCordinates[0];
             measuredLon= measuredGeoCordinates[1];

              NewCord.setText(String.valueOf(measuredLat).substring(0,8)+","+
                      String.valueOf( measuredLon).substring(0, 8));
              NewCord.setEnabled(true);
             diffxx0=x_mob-centralLonToUTM;
             diffyy0=y_mob-centralLatToUTM;

             pmpx=Math.round((float)(cpx+diffxx0/mpp));
             pmpy=Math.round((float)(cpy-diffyy0/mpp));

           marker = new ImageIcon(getClass().getResource("/mymaps/puntV.gif"));
           //changing the size of the image
           marker = new ImageIcon(marker.getImage().getScaledInstance(20, 20,
                   Image.SCALE_SMOOTH));
         markerLabel = new JLabel(marker);

         // Calculamos el punto X medio del mapa para el icono.
           pmpx = Math.round(pmpx- markerLabel.getIcon().getIconWidth()/3);
         // Calculamos el punto Y medio del mapa para el icono.
         pmpy = Math.round(pmpy+markerLabel.getIcon().getIconHeight()/3);

         markerLabel.setBounds(pmpx,pmpy,
                 markerLabel.getIcon().getIconWidth(),
                 markerLabel.getIcon().getIconHeight());
              // Insertamos el componente en la capa 1 (por encima de la capa 0)
             try {
                coordenada1 = new Coordinate(centralLat,centralLon);

             } catch (CoordinateRangeException ex) {
                 Logger.getLogger(Window.class.getName()).log(Level.SEVERE,
                         null, ex);
             }

     if (coldstart)  {
        // System.out.println("coldstart");
         try {
         ArrayList<Marker> listaPuntos = new ArrayList<Marker>();

             mapa = new MarkersMap("", coordenada1, 17, listaPuntos);
             mapa.setDimmension(new Dimension(610, 500));
             mapa.setMapType(MarkersMap.Satellite);

             mapurl=mapa.getMapURL();
             mapurl=mapurl.replaceAll("zoom=17", "zoom="+String.valueOf(zoom));

             try {

             if (PosCalculated==1)
             {
             System.out.println("Ahora empezamos a descargar la imagen " +
             "servidor");
            long tiempoInicio = System.currentTimeMillis();
            image = new ImageIcon(new URL(mapurl));
             this.labelImage.setIcon(image);
             long TotalTiempo = System.currentTimeMillis()-tiempoInicio;
             System.out.println("El tiempo de demora en obtener la imagen " +
                     "del servidor es:" + TotalTiempo/1000 + "  seg");
             }
         } catch (MalformedURLException ex) {
           Logger.getLogger(Window.class.getName()).log(Level.SEVERE, null, ex);
        }
        }catch (MapTypeException ex) {
           Logger.getLogger(Window.class.getName()).log(Level.SEVERE, null, ex);

         }catch (DimensionRangeException ex) {
           Logger.getLogger(Window.class.getName()).log(Level.SEVERE, null, ex);
         } catch (ZoomRangeException ex) {
           Logger.getLogger(Window.class.getName()).log(Level.SEVERE, null, ex);

         }

     }
     else if (warmstart){
            if (PosCalculated==1)
```

```
                {
              image = new ImageIcon(getClass().getResource("/mymaps/mapa1.jpg"));
                this.labelImage.setIcon(image);

                }

        }
         jLayeredPane1.setLayer(markerLabel, new Integer(2));
              jLayeredPane1.add(markerLabel ,new Integer(2));
        }

     public double getmpp(){
         mpp= Math.cos(centralLat*Math.PI/180)*(1/Math.pow(2,(zoom+8)))*40075017;
             return mpp;
        }
        public void checkParkingState(){
        if (jCheckBox1.isSelected()){
            audio=true;
        }
         int delay=10000; //every 2 sg we wheck the state of each parking
    java.awt.event.ActionListener taskPerformer = new
            java.awt.event.ActionListener() {
              public void actionPerformed(java.awt.event.ActionEvent evt) {
                      int num_free_parkings=0;
                       try{
                          li=ServerWrapper.getData();
                         }catch (Exception e){}
                 try{

                     for (int pos=1; pos≤18; pos++){
                        if (li.get(pos).isOcupat()){
                             drawParkingIcon(pos-1, li.get(pos).isOcupat());
                   }
                        if (!li.get(pos).isOcupat()){
                           num_free_parkings++;
                           drawParkingIcon(pos-1, li.get(pos).isOcupat());
                   }
                  }
                   if (contador %10 == 0 && audio){
                      sound(num_free_parkings);
                  }
                  }catch(Exception e){
                  e.printStackTrace();

                  }
                   contador=contador+10.0;
                   if (ini==0){ini=ini+1;}
         }
        };
           new Timer(delay, taskPerformer).start();
        }

        public void drawParkingIcon(int id, boolean state){
            double metersppx, diffanchxx0, diffanchyy0;
            int pmpanchx, pmpanchy;
            double bb,cc;
            metersppx= this.getmpp();

            bb=parkings.get(id).getUTM_lat();
            cc=parkings.get(id).getUTM_lon();
            diffanchyy0 = parkings.get(id).getUTM_lat()-centralLatToUTM;
            diffanchxx0 = parkings.get(id).getUTM_lon()-centralLonToUTM;

            pmpanchx= Math.round((float)(cpx+diffanchxx0/metersppx));
             pmpanchy= Math.round((float)(cpy-diffanchyy0/metersppx));
            pmpanchx=Math.round(pmpanchx - ParkingIconSizeWidth/3);
            pmpanchy=Math.round(pmpanchy+ ParkingIconSizeHeight/3);

                if (ini!=0){
            jLayeredPane1.remove(markerAnchors.get(17));
             jLayeredPane1.validate();
            jLayeredPane1.repaint();
            }
             if (state){
                  markerAnchor1 = new JLabel(nonfreeParking);
             }else{
                  markerAnchor1 = new JLabel(freeParking);
             }
            markerAnchors.add(id,markerAnchor1);
              markerAnchors.get(id).setBounds(pmpanchx, pmpanchy,
                           freeParking.getIconWidth(),freeParking.getIconHeight());
                jLayeredPane1.setLayer(markerAnchors.get(id), new Integer(1));
                jLayeredPane1.add(markerAnchors.get(id) ,new Integer(1));


        }
              public void sound(int numFreeParkings){
            try{

                String file = "/audio/"+String.valueOf(numFreeParkings)+".wav";
              in = getClass().getResourceAsStream(file);
              as = new AudioStream(in);
              AudioPlayer.player.start(as);
            }catch(Exception e){
                e.printStackTrace();
            }

            }
}
```

Listing C.3: JAVA code of ParkingSensors.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package mymaps;

/**
 *
 * @author Administrador
 */
public class ParkingSensors {
    private double UTM_lat, UTM_lon;

    public ParkingSensors(double UTM_lon,double UTM_lat) {
        //Coordinateconversion con = new Coordinateconversion();
        //String[] utm;
        //utm=con.latLon2UTM(GPS_lon, GPS_lon).split("\\s+");
        this.UTM_lat= UTM_lat;
        this.UTM_lon = UTM_lon;
        }


    public double getUTM_lat() {
        return UTM_lat;
    }

    public double getUTM_lon() {
        return UTM_lon;
    }

    public void setUTM_lat(double UTM_lat) {
        this.UTM_lat = UTM_lat;
    }

    public void setUTM_lon(double UTM_lon) {
        this.UTM_lon = UTM_lon;
    }


}
```

Listing C.4: JAVA code of KalmanFilter.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package mymaps;

/**
 *
 * @author spcomnav
 */
public class Kalman_filter {
double[] KFestimates = new double[2];
double samplingT;
double [][]P=new double[2][2];
double[] k = new double[2];

public Kalman_filter(){};
public Kalman_filter(double x0mob,double v, double T,double R){

KFestimates[0]=x0mob;
KFestimates[1]=v;
P[0][0]=R;
P[0][1]=R/T;
P[1][0]= R/T;
P[1][1]=2*R/Math.pow(T, 2);
samplingT=T;
}

public double[] kalman(double statevector[],double V,double R,double z) {
    double posx,vx,T, S,error;
    T=this.samplingT;
    double[] tau, predsv, correction,updsv; //prediction state vector,
    //and updated state vector
    double[][] Q, P_pred,P_upd,P;
    Q= new double [2][2];
    P_pred=new double [2][2];
    P_upd=new double [2][2];
    tau=new double[2];
    predsv=new double[2];
    updsv=new double[2];
    correction=new double[2];
    P=this.P;

    posx=statevector[0];
    vx=statevector[1];
    //BELOW GENERATING KALMAN COEFFICIENTS------------------------------
    tau[0] = (1/2)*Math.pow(T, 2);
    tau[1]=T;

    Q[0][0]=((Math.pow(T, 4))/4)*V;
    Q[0][1]=((Math.pow(T, 3))/2)*V;
     Q[1][0]=Q[0][1];
     Q[1][1]=(Math.pow(T, 2))*V;

     P_pred[0][0]=P[0][0]+P[1][0]*T+P[0][1]*T+P[1][1]
*Math.pow(T, 2)+Q[0][0];
     P_pred[0][1]=P[0][1]+P[1][1]*T+Q[0][1];
      P_pred[1][0]=P[1][0]+P[1][1]*T+Q[1][0];
       P_pred[1][1]=P[1][1]+Q[1][1];

       S=P_pred[0][0]+R;

       k[0]=P_pred[0][0]/S;
       k[1]=P_pred[1][0]/S;

       P_upd[0][0]=P_pred[0][0]-Math.pow(k[0], 2)*S;
       P_upd[0][1]=P_pred[0][1]-k[0]*S*k[1];
        P_upd[1][0]=P_pred[1][0]-k[0]*S*k[1];
         P_upd[1][1]=P_pred[1][1]-Math.pow(k[1], 2)*S;

         this.P=P_upd;
         //---------------------------------------------------------------
         //BELOW RUNNING KALMAN EQUATIONS!!!
         predsv[0]=posx+T*vx;
         predsv[1]=vx;

         error=z-predsv[0];
         correction[0]=k[0]*error;
         correction[1]=k[1]*error;

         updsv[0]=correction[0]+predsv[0];
         updsv[1]=correction[1]+predsv[1];


         return updsv;
  }
```

```
public double[] getcoefficients(){
    return k;
}
}
```

Appendix D

XALOC news

The XALOC project has demonstrated a novel system to automatically guide the drivers to find free parking slots in an urban area. This is done installing one sensor mote in each parking slots. The sensors send to a central server information whether a parking slots is busy or not. On the other hand, all the sensors contribute to perform positioning of the driver using some positioning methods. Thus a navigator implementation is developed to show the real-time position of the driver when he is moving at a low speed looking for an available parking. This navigator shows also the number of free-non free parking slots and has the option to notify the number of available parkings by audio.

The XALOC project was demonstrated in a pilot test in the last INFOREGIO projects announcement (with a reference number 2009INFOREGIO-0016). XALOC project got a large impact to both fields research-technical and the media. The details of the project appeared the following day in TV media (TV,TV3, BTV, ETB,...), radio (RAC1, RNE, COM Ràdio, ...) and press (El periódico de Catalunya, Diari AVUI, La vanguardia, El Punt, ...).

This appendix shows some photos taken during the live demonstration as well as some divulged news from the press obtained on July 07 2010 at the GPS coordinates 41.500848, 2.113959 which is located at the UAB fire department parking.

D.1 Photos of the demonstration day

This section shows some taken photos at the demonstration day with different production companies: TV3, TVE-1, Antena3, Telecinco, BTV noticies, Atlas and eitb.

(a)

(b)

Figure D.1: Photos showing different production companies

.

(a)

(b)

Figure D.2: D.2(a) shows the used car for the demonstration. D.2(b) shows all the used equipment for the demonstration

Figure D.3: They are the BTVNoticies producers.

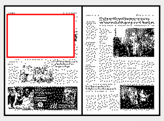
D.2 Live demonstration news

This section shows some recollected news of the live demonstration.

DIARI DE TERRASSA
TERRASSA

08/07/10

Prensa: Diaria
Tirada: 5.998 Ejemplares
Difusión: 4.991 Ejemplares



Sección: LOCAL Valor: 376,00 € Área (cm2): 499,2 Ocupación: 45,55 % Documento: 1/1 Cód: 39572659

Un sistema detecta las plazas libres de parking en la calle

Han participado investigadores de la UAB

El dispositivo es pionero en Europa y funciona a través de internet

Redacción

La mayoría de parkings subterráneos disponen de un panel electrónico que informa de si hay plazas vacías en cada planta. El sistema permite que los conductores vayan directamente hacia la planta donde hay espacios libres. La fórmula evita que los conductores den rodeos por el interior del estacionamientos y que este espacio reduzca el nivel de movilidad por el interior del edificio.

¿Este modelo de gestión podría aplicarse en parkings en superficies como las zonas azules o verdes?. Ésta es la pregunta que se hicieron investigadores de la Universitat Autònoma de Barcelona (UAB) junto con la empresa WorlSensing y el Centre Tecnològic de Telecomunicacions de Catalunya (CTTC). Y ya hay respuesta. El grupo ha desarrollado un dispositivo con tecnología muy avanzada y más precisa que el GPS, capaz de localizar las plazas de aparcamiento libres en la calle y guiar al usuario hasta la más próxima.

El sistema, denominado Xaloc, funciona mediante la instalación de una red de sensores sin hilos sobre las plazas en superficie. Estos sensores detectan si la plaza está ocupada o no y transmiten la información, mediante internet, a un servidor central. El servidor procesa los datos y los envía hacia los paneles indicativos situados en la calle, que muestran la información del estado de ocupación de la zona en tiempo real. Esta información también podría llegar de una forma más personalizada al usuario a través del teléfono móvil o de GPS de última generación.

Ignasi Vilajosana, gerente de WorldSensing, flanqueado por los investigadores Mischa Dohler y José López Vicario.

NOVEDAD Y COSTE

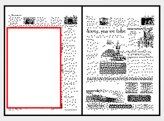
El dispositivo fue presentado ayer por los investigadores José López Vicario y Antoni Morell, del departamento de Telecomunicaciones e Ingeniería de Sistemas de la UAB; Ignasi Vilajosana, gerente de la empresa WorlSensing y Mischa Dohler, investigador del CTTC. El grupo destacó que el sistema es pionero en Europa y que hay algunos ayuntamientos que se han interesado en su aplicación, entre éstos los de Barcelona, Sant Cugat y Sabadell. Todos ellos han mostrado su disposición a probar el invento.

Los responsables prevén que la plataforma sea abierta y de acceso gratuito. Sí que tiene un coste la instalación de los sensores en la calle. Se estima que cada sensor cuesta unos 170 euros, una cantidad que deberían asumir los consistorios. Sin embargo, Vilajosana añadió que el sistema permite adoptar servicios complementarios para amortizar la inversión. Entre éstas señaló la reserva previa de plaza de aparcamiento o la posibilidad de pagar la tarifa de estacionamiento mediante el móvil, según recoge la agencia Europa Press. Para los creadores del invento, el nuevo sistema también es beneficioso para reducir la contaminación de CO2, en la medida que si se conocen las plazas libres y ocupadas hay una circulación más fluida.

QUE! (BARCELONA) 08/07/10
BARCELONA

Prensa: Diaria
Tirada: 200.215 Ejemplares
Difusión: 200.215 Ejemplares

Sección: LOCAL Valor: 5.053,00 € Área (cm2): 608,7 Ocupación: 60,22 % Documento: 1/1 Cód: 39573340



SI NO QUIERES RASCARTE EL BOLSILLO, ESTACIONA POR ENCIMA DE LA RONDA DE DALT

Barcelona pierde plazas de aparcamiento gratis año tras año

El número de espacios para dejar el coche sin pagar descendió en 15.712 en 2009 respecto a 2008. Por contra, el Àrea Verda reservada para los vecinos creció más de un 57%

CARLA MERCADER
carla.mercader@que.es

Estacionar el coche en Barcelona sin rascarse el bolsillo ya es casi imposible en el centro y todo apunta a que se complicará todavía más a medida que pasen los años, ya que que el número de plazas de aparcamiento no reguladas en la calle no para de descender. Según datos municipales, en 2009 eran 121.407 los espacios gratuitos –15.712 menos que en 2008, o lo que es lo mismo, un 11,5% menos–, mientras que en 2007 eran 137.119 y en 2006, 137.326. Las plazas de pago para estacionar en la calzada, por contra, aumentan. Y es que la política municipal de gestión de la movilidad apuesta por este tipo de espacios, que pasaron de ser 43.761 en 2006 a sumar 50.619 en 2009. Los estacionamientos de Àrea Verda Preferent, es decir, exclusiva para los vecinos, son los que más se multiplican. El año pasado eran 35.729, un 57,2% más que en 2008. Este mes de julio el Àrea Verda se ha ampliado en 12.000 plazas en Poblenou, Sarrià, Gràcia, Sant Andreu y Vall d'Hebron.

El Àrea Verda sólo se pone en marcha en los barrios a petición de los vecinos

LOS VECINOS, SATISFECHOS

Jordi Giró, vicepresidente de la Federació d'Associacions de Veïns de Barcelona, afirma que aunque "pagar no gusta" el Àrea Verda "ha descongestionado" la ciudad y "ha mejorado la calidad ambiental y la calidad de vida de los vecinos". "Incluso hay menos coches abandonados o en venta en la calle", apunta. Sin embargo, dice que en algunas zonas, como Gràcia, faltan parkings públicos.

ES MÁS FÁCIL NO PAGAR EN GRÀCIA, HORTA, NOU BARRIS Y SARRIÀ-SANT GERVASI

Más plazas no reguladas en el norte de la ciudad

Desde el Ayuntamiento apuntan que la mayoría de plazas que no son de pago están situadas en las zonas de Sarrià-Sant Gervasi, Gràcia, Horta y Nou Barris que quedan por encima de la Ronda de Dalt.

618.440 plazas en parkings

Según datos del Consistorio de 2009, en Barcelona existen 618.440 plazas de estacionamiento en parkings "fuera de la calzada". Son un 0,9% más que en 2008 y un 4,7% más que en 2006.

El Àrea Verda cuesta hasta 2,94 euros por hora

Los vecinos pagan 1 euro a la semana por el Àrea Verda en el caso de que la usen. Las tarifas para los demás ciudadanos van desde 1,08 hasta 2, 94 euros por hora, en función de la demanda de estacionamiento existente en cada zona.

NACE UN SISTEMA PARA LOCALIZAR SITIOS LIBRES SIN TENER QUE DAR VUELTAS

Investigadores de la UAB, el Centre Tecnològic de Telecomunicacions de Catalunya y la empresa WorldSensing han creado un sistema que localiza plazas de aparcamiento libres en la calle y guía al usuario hasta la más cercana. Se trata de una red de sensores ubicados en las áreas Verda y Blava que detecta el espacio vacío y lo transmite a un servidor central que lo reenvia a dispositivos electrónicos con internet, como teléfonos móviles o GPS.

Reduce las emisiones contaminantes

El sistema llegaría a evitar la emisión de 400 toneladas diarias de CO2, gracias a la reducción del tiempo de búsqueda de plaza, que alcanza los 15,7 minutos de media en Barcelona.

Los consistorios deberían asumir el coste

Los ayuntamientos de Barcelona, Sabadell y Sant Cugat han mostrado su interés por el sistema.

El sistema está pensado para áreas de pago.

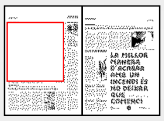
PUBLICO (PUBLIC)
MADRID

08/07/10

Prensa: Diaria
Tirada: 117.459 Ejemplares
Difusión: 74.084 Ejemplares

Página: 6

Sección: SOCIEDAD Valor: 2.306,00 € Área (cm2): 494,5 Ocupación: 47,74 % Documento: 1/1 Cód: 39568717



INNOVACIÓ » Un equip de la UAB idea la manera de reduir temps i contaminació

Un sistema facilita trobar plaça d'aparcament

Trobar plaça d'aparcament pot ser més fàcil, segons asseguren els promotors del nou sistema de localització. ACN

GLÒRIA AYUSO
BARCELONA

Un conductor triga una mitjana de 15,7 minuts a trobar aparcament a Barcelona. Un equip d'investigadors de la Universitat Autònoma de Barcelona (UAB) ha desenvolupat juntament amb l'empresa WorldSensing un sistema que facilita la localització de les places lliures i guia el cotxe fins a la més propera. Els seus promotors asseguren que amb el seu ús es redueix una mitjana de tres minuts la recerca. Tenint en compte que cada dia hi ha més d'un milió de vehicles a la ciutat que aparquen als carrers, això vol dir un estalvi de 3 milions de minuts i una reducció d'emissions de 400 tones de CO_2.

Altres avantatges són la reducció del *trànsit d'agitació*, o trànsit de vehicles que circulen sense rumb específic buscant un lloc on aparcar, de manera que es possibilita una millor fluïdesa.

L'equip, encapçalat per José López Vicario i Antoni Morell, ha treballat durant un any en el desenvolupament del projecte, que ha costat 400.000 euros i ha estat finançat per l'Agència de Gestió d'Ajuts Universitaris i Recerca del Govern. El sistema de localització, anomenat Xaloc, consisteix en la instal·lació de sensors al paviment del carrer, al centre de les zones de les àrees blaves i verdes. Els sensors detecten si la plaça està ocupada i transmeten la informació a través d'internet a un servidor que envia la informació a panells indicatius situats a l'entrada del carrer o a la cruïlla i mostra el seu estat d'ocupació.

El dispositiu estalvia 3 milions de minuts al volant i 400 tones de CO_2

Uns sensors identifiquen places lliures i els cotxes que en busquen

A més, "els sensors del carrer identifiquen els usuaris que busquen plaça", explica Vicario. Això és possible mitjançant la descàrrega del programari d'un navegador portàtil, que els conductors instal·len als seus mòbils o PDA. Els sensors, amb la seva doble funció, possibiliten identificar la ubicació del vehicle. El navegador comunica amb el servidor central i reconeix el nombre de places lliures a tota la ciutat i assenyala les més pròximes al vehicle. L'equip d'investigadors assegura que la tecnologia de posicionament i localització funciona millor que un GPS.

L'empresa WorldSensing té previst comercialitzar el sistema a finals d'any. Per ara s'hi han interessat els ajuntaments de Barcelona i Sabadell, on es portaran a terme proves pilot. El seu responsable, Ignasi Vilajosana, defensa la rendibilitat dels aparells. Cada sensor costa entre 120 i 150 euros. "Si és més fàcil trobar plaça hi ha més rotació a l'àrea blava i verda. En tres anys s'amortitza el preu", explica l'empresari.

Pagament de les àrees

Tot i que destaca que els principals objectius són reduir la contaminació i millorar la fluïdesa del trànsit, el sistema pot tenir una tercera funció gens menyspreable per als consistoris en temps de crisi. Els sensors envien tota la informació sobre la rotació i el temps d'ocupació de les places al servidor, que pot comparar les dades amb el nombre de tiquets que s'han imprès a les màquines d'àrea blava i verda. La Guàrdia Urbana ho tindrà així també més fàcil a l'hora de localitzar els vehicles que no han pagat el tiquet corresponent. ✱

What's your Speedup
7.8x
Accelerate applications on GPUs using PGI compilers and tools
Parallel RNG
www.pgroup.com/accelerator/stats

Scientific COMPUTING INFORMATION TECHNOLOGY FOR SCIENCE

Search
Popular Searches: lims, visualization, chemistry, statistics, hpc

INFORMATICS | HPC | DATA ANALYSIS | DATA SOLUTIONS | LIMS GUIDE | MULTIMEDIA | NEWSLETTERS | NEWS

JOB SEARCH | WHITE PAPERS | SUBSCRIBE | DIGITAL LIBRARY | ADVERTISE | EDITORIAL | CONTACT US | ABOUT US

TOPICS
- Informatics
- HPC
- Data Analysis
- Data Solutions

Home > New System Helps Locate Car Park Spaces

New System Helps Locate Car Park Spaces

By AlphaGalileo

A research group from the Universitat Autònoma de Barcelona Department of Telecommunications and Systems Engineering at the School of Engineering, led by José López Vicario and Antoni Morell, took part in the development of a new system which locates unoccupied car park spaces and guides users to the nearest one. The new network of sensors for the management of public car parks and locations, which researchers have named XALOC (Xarxes de sensors per a la gestió d'Aparcaments públics i LOCalització), was developed by a consortium formed by the firm WorldSensing (consortium leader) and the Centre for Telecommunications Technology of Catalonia (CTTC). The project was financed by Catalan Government's Agency for Administration of University and Research Grants (AGAUR).

The project's consortium developed a platform based on a network of wireless sensors capable of detecting unoccupied spaces outdoors, and on an alternative positioning system with more precision in urban areas than GPS technology. This platform is capable of locating and guiding drivers to car park spaces available in the area.

The network's sensors are located on the ground directly in the middle of the car park space. The sensors detect whether the space is occupied or not and send information via internet to a central station. The server processes this information and sends it to indication panels located in the street which display the information in real time. Advanced communication techniques are used to send guidance data to the network.

The sensor platform at the same time locates users looking to park and thus offers a personalised service. UAB researchers have designed a specific portable navigator for users called ARID Navigator which makes use of communication signals belonging to the network of sensors to position users within their urban surroundings. Once the vehicle is located, the navigator communicates with XALOC's central server and reports to the user the number of available car park spaces in the area and where they are located.

The positioning and location technology used to develop the system is totally new and offers many advantages in comparison to conventional GPS navigators, such as more precise urban location techniques, reduced positioning time and better coverage.

The XALOC system will improve traffic management in urban areas and reduce what is known as "agitated traffic", traffic caused by drivers circulating and looking for a place to park. Reducing the volume of agitated traffic will allow for a substantial improvement in circulation fluidity in urban areas and thus contribute to effective reductions in pollution and an increase in citizen satisfaction.

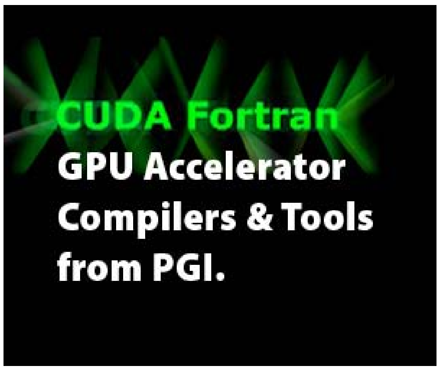
SOURCE

Email for more Information.

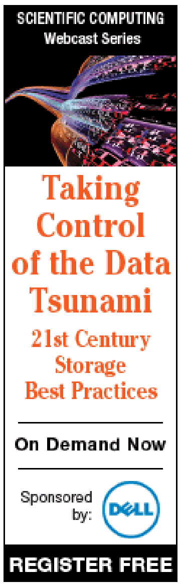
Email Article | Contact the Editor | Printer Friendly

Post to Del.icio.us | Digg This | Post to Slashdot

Click for FREE Trial Download
SIGMAPLOT
Exact Graphs and Data Analysis
Publication Quality Graphs & Statistical Analysis
> Easy to Use
> Regression / Curve Fitting
>Excel Integration
>Statistical Wizard
>And Much More



Most Viewed Content
- Russian Mathematician Rejects $1 Million Millennium Prize
- Famed Hominid Lucy no Longer Alone
- New Hawking Book Due Out this Fall
- Bionic Cat gets Prosthetic Paws
- Programmable Matter Creates Self-folding Sheets
- Pulsars Help Uncover Ripples in Space-Time
- Galactic Archaeologists Find Origin of Milky Way's Stars
- Improved Telescope Peers Clearly Through Atmosphere
- Seeking a Bridge to the Quantum World
- Mass Spectrometry Used in Oil Spill Research Mission

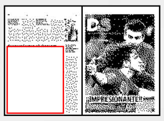

SCIENTIFIC COMPUTING Webcast Series
Taking Control of the Data Tsunami
21st Century Storage Best Practices
On Demand Now
Sponsored by: DELL
REGISTER FREE

SUR
MALAGA

08/07/10

Prensa: Diaria
Tirada: 36.690 Ejemplares
Difusión: 30.438 Ejemplares

Sección: SOCIEDAD Valor: 2.166,00 € Área (cm2): 513,8 Ocupación: 50,13 % Documento: 1/1 Cód: 39582540



Aparcamientos vía Internet

Diseñan un sistema que informa a los conductores de las plazas libres en la calle

La UAB desarrolla unos dispositivos que transmiten a través de la Red y de los GPS la presencia o no de vehículos en la zona

:: EFE

BARCELONA. Dar vueltas por la vías de la ciudad buscando un hueco donde dejar el coche tiene los días contados, si finalmente se comercializa un proyecto desarrollado por un grupo de investigadores de la Universidad Autónoma de Barcelona (UAB) y que actualmente se encuentra en pruebas.

El sistema, que informa a los conductores de los aparcamientos libres de las zonas verdes y azules en superficie, cuenta con un dispositivo emisor de datos que se coloca en las plazas y que detecta la presencia o no de vehículos en el punto delimitado por la pintura.

Estos emisores transmiten los datos a un servidor de Internet, desde donde se distribuye la información a los diferentes dispositivos interconectados con el servicio de información. «Desde Internet los datos se transmiten a los aparatos correspondientes, vía web o a los dispositivos GPS, de modo que el conductor es guiado a las plazas de aparcamiento libres más cercanas», ha explicado el gerente de la empresa WorldSensing, Ignasi Vilajosana. Esta compañía, junto con el Centro Tecnológico de Telecomunicaciones de Cataluña han participado en el proyecto, que fue presentado ayer en el campus de la Autónoma de Barcelona.

El sistema está pensado de momento para plataformas web, es decir, desde ordenadores, «smartphones» o dispositivos GPS que tengan conexión a Internet y que puedan conocer en tiempo real cuál es el estado de las calles más cercanas a su emplazamiento, con una localización del vehículo también en tiempo real. «El principal problema que nos encontramos fue la distancia que teníamos que cubrir entre la plaza de aparcamiento y los dispositivos receptores de los usuarios, y optamos por un sistema que conectaba los emisores de las plazas entre sí para acercar la información al punto de difusión», ha explicado el investigador del Centro Tecnológico de Telecomunicaciones de Catalunya Mischa Dohlet.

La baja conectividad se debe a que los dispositivos emisores se instalan bajo tierra, como las balizas luminosas de las rotondas, y esto disminuye su capacidad de envío de datos.

Se calcula que el sistema puede reducir en al menos cinco minutos el tiempo que invierte un conductor a la hora de buscar aparcamiento, lo que supondría una reducción de unas 400 toneladas de dióxido de carbono anuales.

Aunque en periodo de pruebas, varios ayuntamientos, como el de Barcelona o el de Sabadell (Barcelona), ya han mostrado su interés para llevar a cabo pruebas piloto en sus ciudades.

Único e innovador

Se trata de un sistema innovador que no existe en ningún otro país, y que mejora el que se encuentra en San Francisco, pensado básicamente para el pago del uso de las plazas de aparcamiento al aire libre.

El mecanismo abre nuevas puertas y mercados, como la posibilidad de incorporar el pago directo, mejorando al de Estados Unidos a través de la conexión vía operador móvil, que requeriría un acuerdo con algunas de las empresas que actualmente operan en el país.

El proyecto ha costado 400.000 euros y su instalación supone una inversión de unos 170 euros por plaza de aparcamiento, que por otra parte los ayuntamientos podrían recuperar en dos años gracias a un incremento de la estancia de los vehículos en las plazas de pago al tener que invertir menos tiempo buscando aparcamiento.

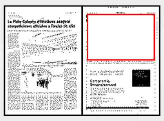
El sistema detecta los aparcamientos libres en superficie. :: SUR

DIARI DE SABADELL
SABADELL

08/07/10

Prensa: Diaria
Tirada: 5.659 Ejemplares
Difusión: 4.438 Ejemplares

Sección: LOCAL Valor: 384,00 € Área (cm2): 436,3 Ocupación: 45,75 % Documento: 1/1 Cód: 39573423



Diseñan un sistema que localiza las plazas de aparcamiento libres

El Ayuntamiento de Sabadell se interesa por este mecanismo de la UAB

Investigadores de la UAB han diseñado un sistema que localiza plazas libres de aparcamiento en la calle y guía al usuario hasta la próxima. El Ayuntamiento de Sabadell ya se ha interesado.

I. LOPERA

Un equipo de investigadores del Departament de Telecomunicació i Enginyeria de Sistemes de la UAB, en la Escola d'Enginyeria, encabezados por José López Vicario y Antoni Morell ha participado en el desarrollo de un nuevo sistema que localiza plazas de aparcamiento libres en la calle y guía al usuario hasta la más cercana, reduciendo el tiempo que invierte en buscar estacionamiento.

Está pensado para zonas urbanas y plazas de pago, lo que permitiría amortizar el coste de instalación, presupuestado en 400 mil euros y en unos 170 euros por cada plaza. Así pues, se sufragaría en un par de años. Por ello, ayuntamientos como Sabadell, Barcelona o Mataró se han mostrado interesados en el proyecto, todavía en fase de prueba. Ha sido diseñado por un consorcio donde también participan la empresa World-Sensing (líder del consorcio) y el Centre Tecnològic de Telecomunicacions de Catalunya.

El sistema consiste en una red de sensores sin hilos capaces de detectar plazas de aparcamiento libres en exteriores y labores de localización de vehículos con un sistema alternativo al GPS, y más preciso que éste en zonas urbanas.

Con esta plataforma es posible localizar y guiar a los conductores hacia las plazas de aparcamiento disponibles dentro del área de interés.

Los sensores de la red se sitúan en el pavimento de la calle, justo a centro de las áreas azules y verdes que detectan si está o no ocupada, y transmiten la información, mediante internet, a un servidor central. Este servidor las procesa y las envía a paneles informativos situados en la calle que muestran la información del estado de ocupación de la zona en tiempo real. Se ha utilizado técnicas de comunicación avanzadas.

La plataforma de sensores localiza, a su vez, a los usuarios que buscan aparcamiento, de manera que se puede ofrecer un servicio personalizado. En concreto, los investigadores de la UAB han diseñado un navegador portátil para el usuario, llamado ARID Navigator, que aprovecha las señales de comunicación, propias de la red de sensores, para posicionarse dentro del entorno urbano. Una vez el vehículo es localizado, el navegador se comunica con el servidor central de Xaloc para conocer el número de plazas libres en la zona y su ubicación, y muestra toda la información al usuario.

NORMA VIDAL (ACN

El conductor podrá consultar las plazas libres por internet

Mejor que el GPS

La tecnología de posicionamiento y de localización es totalmente nueva y ofrece grandes ventajas respecto a navegadores convencionales basados en GPS, como son una localización más precisa en áreas urbanas, un tiempo de posicionamiento más reducido y mejor cobertura del servicio.

Xaloc mejorará la gestión del tráfico disminuyendo lo que los expertos llaman «tráfico de agitación», es decir, los vehículos que circulan sin rumbo específico buscando un lugar donde aparcar. Una disminución del volumen del tráfico de agitación mejorará la fluidez de la circulación de forma sustancial en zonas urbanas para contribuir a una reducción efectiva de la contaminación —más de 400 toneladas de dióxido de carbono, sólo en Barcelona— y un aumento de la satisfacción del conductor ■

|

ECTO | XALOC

Jueves 08/07/2010. Actualizado **16:18h.**

Su aparcamiento libre, tras el segundo cruce a la derecha

- En Barcelona, los coches en busca de plaza emiten hasta 15 toneladas de CO^2
- El XALOC instala un sensor en cada aparcamiento, que avisa si está libre
- La información se transmite a una pantalla o a un navegador en el vehículo

Eva Belmonte | Barcelona

Actualizado **jueves 08/07/2010 16:18 horas**

En una ciudad como Barcelona, los coches en busca de aparcamiento provocan diez toneladas al día de CO^2. Si al perjuicio ecológico le sumamos el impacto en la salud mental de los conductores, que dedican entre 10 y 15 minutos al día a esta engorrosa tarea, las rondas en busca de plaza se convierten en un mal endémico de las grandes ciudades. Un proyecto de la **Universitat Autònoma de Barcelona** (UAB) le hace frente presentando el primer sistema para localizar aparcamientos en plena calle.

Desarrrollado por la UAB, la empresa WorldSensing y el Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), el XALOC está compuesto por una red de sensores que se colocan en las plazas de parking. **Cada uno de ellos envía una señal -libre/ocupado-** a un servidor que coloca los inputs en un mapa de la ciudad. Esta valiosa información se visualizaría en pantallas luminosas que indiquen el camino a seguir a los conductores.

Como complemento, los investigadores de la UAB han diseñado un navegador portátil para el coche -ARID-, que indica al conductor, tras lanzar al servidor su posición, la plaza libre más cercana. "**La idea es que ARID se pueda usar en cualquier terminal conectada a internet, como un iPad o una Blackberry**, por ejemplo", adelanta José López Vicario, uno de los investigadores que ha liderado el proyecto.

El sistema de sensores funciona **como el de los parkings privados** que iluminan en verde las plazas libres. La diferencia sustancial de este proyecto, tal y como la explica López, es que los aparcamientos subterráneos funcionan con ultrasonidos colocados encima de los coches, un sistema imposible de implantar en plena calle. Además de instalar **sensores en la superficie**, justo en el centro de la plaza de aparcamiento, el XALOC permite centralizar toda la información. Esta red de aparcamientos, además, mejora la conectibilidad del GPS, ya que funciona, advierte López, **"en subterráneos y zonas de la ciudad donde el GPS no llega"**.

Aunque por el momento no se aplicará en ninguna ciudad española, los responsables del proyecto **han establecido contactos con ayuntamientos como el de Barcelona**. La capital catalana podría convertirse, así, en un enorme parking puntero al aire libre. Y en un descanso para el conductor de ronda.

universia
red de universidades, red de oportunidades

NOTICIA AMPLIADA

Mischa Dohler, investigador del CTTC; Ignasi Vilajosana gerente de WorldSensing; y José López Vicario, investigador de la UAB

INNOVACIÓN

Desarrollan un sistema para localizar plazas de aparcamiento en la calle

Universitat Autònoma de Barcelona

Investigadores de la UAB, de la empresa WorldSensing y del Centro Tecnológico de Telecomunicaciones de Catalunya (CTTC) han desarrollado un sistema que localiza plazas de aparcamiento libres en la calle y que guía al usuario hasta la más próxima. El sistema, al que han llamado XALOC, está basado en una nueva tecnología de localización más precisa que el GPS en zonas urbanas.

8/7/2010

COMPARTIR

El sistema XALOC mejorará la gestión del tránsito en entornos urbanos, disminuyendo lo que los expertos llaman "tránsito de agitación", es decir, el tránsito de vehículos que circulan sin rumbo específico buscando un lugar donde aparcar

Un equipo de investigadores del Departamento de Telecomunicación e Ingeniería de Sistemas de la UAB , en la Escuela de Ingeniería, dirigido por José López Vicario y Antoni Morell, ha participado en el desarrollo de un nuevo sistema que **localiza plazas de aparcamiento libres en la calle y guía al usuario hasta la más cercana**. El sistema, llamado XALOC (Xarxes de sensors per a la gestió d'Aparcaments públics i LOCalització), ha sido desarrollado por un consorcio en el que también participan la empresa WorldSensing (líder del consorcio), y el Centro Tecnológico de Telecomunicaciones de Catalunya (CTTC), financiado por la Agència de Gestió d'Ajuts Universitaris i de Recerca (AGAUR) de la Generalitat de Catalunya.

El consorcio del proyecto ha desarrollado una plataforma basada en una **red de sensores sin hilos capaz de realizar tareas de detección de plazas libres de aparcamiento en exteriores**, y tareas de localización de vehículos con un sistema alternativo al GPS, y más preciso que éste en zonas urbanas. Con esta plataforma es posible localizar y guiar a los conductores hacia las plazas de aparcamiento disponibles dentro del área de interés.

Los sensores de la red se sitúan en el pavimento de la calle, justo en el centro de las áreas azules y verdes. Estos sensores detectan si la plaza está o no ocupada, y transmiten la información, mediante Internet, a un servidor central. Este servidor las procesa y las envía a paneles indicativos situados en la calle que muestran la información del estado de ocupación de la zona en tiempo real. Se han utilizado técnicas de comunicación avanzadas para llevar a cabo el guiado de los datos de la red.

Al mismo tiempo, la plataforma de sensores localiza a los usuarios que buscan aparcamiento, de modo que se puede ofrecer un servicio personalizado. En concreto, los investigadores de la UAB han diseñado un **navegador portátil para el usuario**, llamado ARID Navigator, que aprovecha las señales de comunicaciones, propias de la red de sensores, para posicionarse en el entorno urbano. Una vez se localiza el vehículo, el navegador se comunica con el servidor central de XALOC para conocer el número de aparcamientos libres en la zona y su ubicación, y muestra toda esta información al usuario.

La tecnología de posicionamiento y de localización es totalmente nueva y ofrece **grandes ventajas respecto a los navegadores convencionales**, basados en GPS, como son una localización más precisa en entornos urbanos, un tiempo de posicionamiento más reducido y mejor cobertura del servicio.

El sistema XALOC mejorará la gestión del tránsito en entornos urbanos, disminuyendo lo que los expertos llaman **"tránsito de agitación"**, es decir, el tránsito de vehículos que circulan sin rumbo específico buscando un lugar donde aparcar. Una disminución del volumen del tránsito de agitación permitirá mejorar la fluidez de la circulación de manera substancial en entornos urbanos, para contribuir a una reducción efectiva de la contaminación y a un aumento de la satisfacción del ciudadano.

Bibliography

[1] Y. I.F.Akyildiz, S.Weilian and E.Cayirci, “A survey on sensor networks,” *Communications Magazine, IEEE*, vol. 40, no. 8, pp. 102 – 114, Aug 2002.

[2] G. S.Evripidis and T.Zahariadis, “AWSN Protocol Stack Specification and High Level Node Architecture,” Telecommunications System Institute, Tech. Rep., Oct 2008. [Online]. Available: ”http://www.awissenet.eu/downloadfile.aspx?fid=11501ftype=d”

[3] [Online]. Available: ”http://www.xbow.com/Products/productdetails.aspx?sid=264”

[4] A.Savvides, “Introduction to location discovery, lecture 4.”

[5] S. A. I. R. N.Patwari, J.N.Ash and N.S.Correal, “Locating the nodes: cooperative localization in wireless sensor networks,” *Signal Processing Magazine, IEEE*, vol. 22, no. 4, pp. 54 – 69, July 2005.

[6] J. N.Bulusu and D.Estrin, “Gps-less low-cost outdoor localization for very small devices,” *Personal Communications, IEEE*, vol. 7, no. 5, pp. 28 –34, Oct 2000.

[7] D.Niculescu and B.Nath, “DV based positioning in ad hoc networks,” *Telecommunication Systems*, vol. 22, no. 1, pp. 267–280, 2003.

[8] C. A.Savvides and M.B.Srivastava, “Dynamic fine-grained localization in ad-hoc wireless sensor networks,” *UC Los Angeles: Center for Embedded Network Sensing. Retrieved from: http://www. escholarship. org/uc/item/6xs0j41x*, 2001.

[9] “Wireless hybrid enhanced mobile radio estimators.”

[10] M. C.Rohrig, “Localization of sensor nodes in a wireless sensor network using the nanoloc trx transceiver,” April 2009, pp. 1 –5.

[11] X.-h. Z.Ke, L.Yang and S.Heejong, “The application of a wireless sensor network design based on zigbee in petrochemical industry field,” Nov. 2008, pp. 284 –287.

[12] K. Y.Bar-Shalom, L.X.-rong. 605 Third Avenue, New York, NY 10158-0012: Willey-Interscience, 2001, vol. 1.

[13] G. Welch and G. Bishop, “An introduction to the Kalman filter,” *University of North Carolina at Chapel Hill, Chapel Hill, NC*, 1995.

[14] Y.Tai and Y.Bo, “Collaborative target tracking in wireless sensor network,” Aug. 2009, pp. 2–1005 –2–1010.

[15] H.-L. M.E.Farmer and A.K.Jain, "Interacting multiple model (imm) kalman filters for robust high speed human motion tracking," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 2, 2002, pp. 20 – 23 vol.2.

[16] Atmel, "ZigBit$^{\text{tm}}$ 700/800/900 mhz wireless modules, atz-900-b0." Telecommunications System Institute, Tech. Rep., Oct 2008. [Online]. Available: "http://www.atmel.com/dyn/resources/prod_documents/doc8227.pdf"

[17] A. W.R.Heinzelman and H.Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8*. Washington, DC, USA: IEEE Computer Society, 2000, p. 8020.

[18] M.Shemshaki and H.S.Shahhoseini, "Energy efficient clustering algorithm with direct path supports," May 2009, pp. 277 –281.

[19] O.Buyanjargal and K.Youngmi, "An Energy Efficient Clustering Algorithm for Event-Driven Wireless Sensor Networks (EECED)," Aug. 2009, pp. 1758 –1763.

[20] F.G.Mula, "Redes de sensores inalámbricos," Computer and Architecture Department,University of Granada, Tech. Rep., July 2007.

[21] K. Soe, "Increasing lifetime of target tracking wireless sensor networks," *World Academy of Science Engineering and Technology*, vol. 32, no. 32, pp. 440–446, 2008.

[22] S.Suganya, "Cluster-based approach for collaborative target tracking in wireless sensor networks," July 2008, pp. 276 –281.

[23] S. S.Balasubramanian, I.Elangovan and K.R.Namuduri, "Distributed and collaborative tracking for energy-constrained ad-hoc wireless sensor networks," vol. 3, March 2004, pp. 1732 – 1737 Vol.3.

[24] O.-H. S.Kuo-Feng and H.C.Jiau, "Localization with mobile anchor points in wireless sensor networks," *Vehicular Technology, IEEE Transactions on*, vol. 54, no. 3, pp. 1187 – 1197, May 2005.

[25] B.-J. T.He, C.Huang and T.Abdelzaher, "Range-free localization schemes for large scale sensor networks," in *Proceedings of the 9th annual international conference on Mobile computing and networking*. ACM, 2003, p. 95.

[26] P.-P. X. S.Tian, X.Zhang, "A RSSI-based DV-hop algorithm for wireless sensor networks," in *International Conference on Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007*, 2007, pp. 2555–2558.

[27] J. M. C.Savarese and K.Langendoen, "Robust positioning algorithms for distributed ad-hoc wireless sensor networks," in *ATEC '02: Proceedings of the General Track of the annual conference on USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2002, pp. 317–327.

[28] R.Nagpal, "Organizing a global coordinate system from local information on an amorphous computer," 1999.

[29] Y. O.Baala and A.Caminada, "The impact of ap placement in wlan-based indoor positioning system," March 2009, pp. 12 –17.

[30] O.Baala and A.Caminada, "Wlan-based indoor positioning system: experimental results for stationary and tracking ms," Nov. 2006, pp. 1 –4.

[31] A.-J. H.Wang, H.Lenz and U.D.Hanebeck, "Wlan-based pedestrian tracking using particle filters and low-cost mems sensors," March 2007, pp. 1 –7.

[32] T. T.Kitasuka and A. Fukuda, "Design of wips: Wlan-based indoor positioning system," *Korea Multimedia Society*, vol. 7, no. 4, pp. 15–29, 2003.

[33] F.-J. F.Izquierdo, M.Ciurana and E.Zola, "Performance evaluation of a toa-based trilateration method to locate terminals in wlan," Jan. 2006, pp. 1 – 6.

[34] M. U.Grossmann and S.Hakobyan, "Rssi based wlan indoor positioning with personal digital assistants," Sept. 2007, pp. 653 –656.

[35] S. M.Ciurana and F.Barcelo-Arroyo, "Wlan indoor positioning based on toa with two reference points," March 2007, pp. 23 –28.

[36] T. A.Awad and F.Dressier, "Adaptive distance estimation and localization in wsn using rssi measures," Aug. 2007, pp. 471 –478.

[37] M. J.G.Castano and M.Ekstrom, "Local positioning for wireless sensors based on bluetooth trade;," Sept. 2004, pp. 195 – 198.

[38] G. F.Forno and G.Portelli, "Design and implementation of a bluetooth ad hoc network for indoor positioning," *Software, IEE Proceedings -*, vol. 152, no. 5, pp. 223 – 228, Oct. 2005.

[39] J. H.Wymeersch and M.Z.Win, "Cooperative localization in wireless networks," *Proceedings of the IEEE*, vol. 97, no. 2, pp. 427 –450, Feb. 2009.

[40] A.Gopakumar and L.Jacob, "Localization in ultra wideband sensor networks using tabu search," Jan. 2008, pp. 99 –102.

[41] a. Z. Z.Tingting, Z.Qinyu, "A two-step toa estimation method based on energy detection for ir-uwb sensor networks," May 2009, pp. 139 –145.

[42] [Online]. Available: "http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/IRIS_Da

[43] P. F. J.Chen, X.J.Wu and J.W.Liu, "A new distributed localization algorithm for zigbee wireless networks," June 2009, pp. 4451 –4456.

[44] F. F. S.Tennina, M.Di Renzo, "On the distribution of positioning errors in wireless sensor networks: A simulative comparison of optimization algorithms," 31 2008-April 3 2008, pp. 2075 –2080.

[45] L.Xinrong, "Collaborative localization with received-signal strength in wireless sensor networks," *Vehicular Technology, IEEE Transactions on*, vol. 56, no. 6, pp. 3807 – 3817, Nov. 2007.

[46] J.Xiang and Z.Hongyuan, “Sensor positioning in wireless ad-hoc sensor networks using multidimensional scaling,” vol. 4, March 2004, pp. 2652 – 2661 vol.4.

[47] M.Hui and B.W.-H.Ng, “Collaborative data and information processing for target tracking in wireless sensor networks,” Aug. 2006, pp. 647 –652.

[48] E. D.Ma and L. Beng, “A comprehensive study of kalman filter and extended kalman filter for target tracking in wireless sensor networks,” Oct. 2008, pp. 2792 –2797.

[49] C.Mosquera and S.K.Jayaweera, “Entangled kalman filters for cooperative estimation,” July 2008, pp. 279 –283.

[50] L. Boyd and S. Vandenberghe. Upper Saddle River, New Jersey 07458: Prentice Hall PTR, 1993, vol. 1.

[51] R. S.Ahmad and P.Graham, “Design and implementation of a sensor network based location determination service for use in home networks,” Oct. 2006, pp. 622 –626.

[52] K.Lorincz and M.Welsh, “Motetrack: a robust, decentralized approach to rf-based location tracking,” *Personal and Ubiquitous Computing*, vol. 11, no. 6, pp. 489–503, 2007.

[53] B.Zhang and F.Yu, “An energy efficient localization algorithm for wireless sensor networks using a mobile anchor node,” June 2008, pp. 215 –219.

[54] E. M.Di and L.H.Beng, “A comprehensive study of kalman filter and extended kalman filter for target tracking in wireless sensor networks,” Oct. 2008, pp. 2792 –2797.

[55] M. L. H.L.Vu, T.T.Tran, “A simple method for positioning and tracking in wireless sensor networks,” Dec. 2008, pp. 229 –233.

[56] C.Rohrig and M.Muller, “Localization of sensor nodes in a wireless sensor network using the nanoloc trx transceiver,” April 2009, pp. 1 –5.

[57] Y. Kim and K. Hong, “An IMM algorithm for tracking maneuvering vehicles in an adaptive cruise control environment,” *INTERNATIONAL JOURNAL OF CONTROL AUTOMATION AND SYSTEMS*, vol. 2, pp. 310–318, 2004.

[58] C. S.C.Hu, Y.C.Wang and Y.C.Tseng, “A vehicular wireless sensor network for co2 monitoring,” Oct. 2009, pp. 1498 –1501.

[59] Y. V.W.S.Tang and J.Cao, “An intelligent car park management system based on wireless sensor networks,” Aug. 2006, pp. 65 –70.

[60] [Online]. Available: ”http://code.google.com/p/jposition/”