Kernel-based machine learning for tracking and environmental monitoring in wireless sensor networkds

Sandy Mahfouz

Thèse
de doctorat
de l'UTT

Sandy MAHFOUZ

Kernel-based Machine Learning for Tracking and Environmental Monitoring in Wireless Sensor Networks

Spécialité :
Optimisation et Sûreté des Systèmes

2015TROY0025

Année 2015

utt
université de technologie
Troyes

THESE

pour l'obtention du grade de

DOCTEUR de l'UNIVERSITE DE TECHNOLOGIE DE TROYES

Spécialité : OPTIMISATION ET SURETE DES SYSTEMES

présentée et soutenue par

Sandy MAHFOUZ

le 14 octobre 2015

Kernel-based Machine Learning for Tracking and Environmental Monitoring in Wireless Sensor Networks

JURY

M. C. RICHARD	PROFESSEUR DES UNIVERSITES	Président (Rapporteur)
Mme J. FARAH	PROFESSOR	Examinateur
M. P. HONEINE	PROFESSEUR DES UNIVERSITES	Directeur de thèse
Mme F. MOURAD-CHEHADE	MAITRE DE CONFERENCES	Directrice de thèse
M. F. SEPTIER	MAITRE DE CONFERENCES	Examinateur
M. J.-Y. TOURNERET	PROFESSEUR DES UNIVERSITES	Rapporteur

Personnalité invitée

M. H. SNOUSSI	PROFESSEUR DES UNIVERSITES

Acknowledgements

First and foremost, I would like to gratefully and sincerely thank my advisors, Farah Mourad-Chehade and Paul Honeine, for their excellent guidance, patience, and most importantly, their friendship during these last three years. Without them, this thesis would not have been successfully completed. One simply could not wish for better or more devoted advisors.

Next, I would like to thank the members of my defense committee for their time and dedication. I thank the reviewers, Cédric Richard and Jean-Yves Tourneret, who patiently read the manuscript and provided detailed comments and insightful suggestions. I also thank the examiners Joumana Farah and François Septier, as well as Hichem Snoussi for accepting to serve on my defense committee. Their helpful feedback greatly improved the contents of this manuscript.

I particularly thank Joumana Farah and Hichem Snoussi for their support and assistance during this thesis. They contributed in its success by closely following up my work during these three years.

I also would like to thank all the members of the LM2S Laboratory for their support and friendship. My gratitude extends to Bernadette André and Véronique Banse for their availability and all the help they provided me with. I also thank the UTT doctoral school, in particular the director Régis Lengellé, and the secretaries Pascale Denis, Isabelle Leclercq and Thérèse Kazarian.

My thanks go to all my colleagues and all the friends I made during these last three years, in particular, to Elie, Nathalie, Nisrine, Fei, Yuan and Long. A special thanks goes to my greatly supportive university friends Pauline, Cédric and Rodrigue. Words cannot express my infinite love and gratitude for all of you. Thank you for your patience and encouragement throughout this thesis!

I greatly thank my mother Samira and my father Simon for their unfailing support and their faith in me. I am indebted to them for teaching me dedication and discipline to do whatever I undertake well. I also thank my sister Cynthia and my brother Charbel for always managing to put a smile on my face. Summer 2015 was a tough one for us, but we managed to get through all the hard times together.

Finally, I would like to express my deepest love and thanks to my dear husband Patrick. The last eight years of my life were amazing because of you. We went through engineering school together and both managed to complete our PhD theses without going crazy! And together, we shall continue our life journey through the good times and bad. Thank you for everything...

Dedicated to my parents, my sister and my brother
Dedicated to Patrick, my partner in crime

Abstract

This thesis focuses on the problems of localization and gas field monitoring using wireless sensor networks. First, we focus on the geolocalization of sensors and target tracking. Using the powers of the signals exchanged between sensors, we propose a localization method combining radio-location fingerprinting and kernel methods from statistical machine learning. Based on this localization method, we develop a target tracking method that enhances the estimated position of the target by combining it to acceleration information using the Kalman filter. We also provide a semi-parametric model that estimates the distances separating sensors based on the powers of the signals exchanged between them. This semi-parametric model is a combination of the well-known log-distance propagation model with a non-linear fluctuation term estimated within the framework of kernel methods. The target's position is estimated by incorporating acceleration information to the distances separating the target from the sensors, using either the Kalman filter or the particle filter. In another context, we study gas diffusions in wireless sensor networks, using also machine learning. We propose a method that allows the detection of multiple gas diffusions based on concentration measures regularly collected from the studied region. The method estimates then the parameters of the multiple gas sources, including the sources' locations and their release rates.

Keywords:

- Sensor networks
- Machine learning
- Non-linear models
- Signal processing
- Kalman filtering

Résumé

Cette thèse porte sur les problèmes de localisation et de surveillance de champ de gaz à l'aide de réseaux de capteurs sans fil. Nous nous intéressons d'abord à la géolocalisation des capteurs et au suivi de cibles. Nous proposons ainsi une approche exploitant la puissance des signaux échangés entre les capteurs et appliquant les méthodes à noyaux avec la technique de radio-cartographie dite fingerprinting. Nous élaborons ensuite une méthode de suivi de cibles, en se basant sur l'approche de localisation proposée. Cette méthode permet d'améliorer la position estimée de la cible en tenant compte de ses accélérations, et cela à l'aide du filtre de Kalman. Nous proposons également un modèle semi-paramétrique estimant les distances inter-capteurs en se basant sur les puissances des signaux échangés entre ces capteurs. Ce modèle est une combinaison du modèle physique de propagation avec un terme non linéaire estimé par les méthodes à noyaux. Les données d'accélérations sont également utilisées ici avec les distances, pour estimer la position de la cible, en s'appuyant sur un filtrage de Kalman et un filtrage particulaire. Dans un autre contexte, nous proposons une méthode pour la surveillance de la diffusion d'un gaz dans une zone d'intérêt, basée sur l'apprentissage par noyaux. Cette méthode permet de détecter la diffusion d'un gaz en utilisant des concentrations relevées régulièrement par des capteurs déployés dans la zone. Les concentrations mesurées sont ensuite traitées pour estimer les paramètres de la source de gaz, notamment sa position et la quantité du gaz libéré.

Mots-clés :

- Réseaux de capteurs (technologie)
- Apprentissage automatique
- Modèles non linéaires (statistique)
- Traitement du signal
- Kalman, Filtrage de

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

Wireless sensor networks (WSNs) have become a major research field during the last years. They are composed of a large number of tiny autonomous wireless sensors that are spatially distributed and can communicate and exchange data. The sensors are mostly battery powered; therefore, they have a limited lifetime, and hence all built-in algorithms must focus on reducing energy consumption. Nowadays, WSNs are rapidly gaining importance in a wide range of applications, especially in military, environmental, and healthcare domains. This chapter first introduces wireless sensor networks, their characteristics, limitations and applications. Next, the localization and tracking issues in WSNs are discussed. The problem of detection and localization of a gas diffusion is then presented as well. Finally, the main contributions of this thesis and the organization of the manuscript are outlined.

1.1 Wireless Sensor Networks

Recent advances in electronics and wireless communication technologies, coupled with the need to continuously monitor physical phenomena, have led to the emergence of wireless sensor networks (WSNs). According to the MIT Technology Review, 2003, WSNs are classified as one of the top 10 emerging technologies that will change the world. Indeed, WSNs are likely to be transformative, compared to the traditional wired networks. According to Freedonia Group report on sensors, 2002, the wireless sensor market in 2001 was approximated to $11 Billion, while the wiring installation costs were more than $100 Billion. In this section, we first describe the architecture of the elements of a WSN, i.e., the architecture of the smart wireless sensors; then, we proceed to the description of the characteristics and limitations of WSNs. Finally, we give some examples of applications based on WSNs.

1.1.1 Smart sensors

Wireless sensor networks are networks of compact smart sensors with wireless communication capability [Akyildiz et al., 2002; Vieira et al., 2003; Sohraby et al., 2007]. These small devices are relatively cheap and have the potential to be disseminated in large quantities. Each sensor is capable of acquiring and processing data, and communicating with the neighboring sensors in order to constitute a network. A sensor is hence composed of four major units, as follows:

- **The power supply:** it is usually a non-renewable battery, whose purpose is to power all the other blocks. If feasible, energy harvesters such as solar powered

chargeable batteries are used to minimize the failure of sensors and maintenance cost.

- **The sensing unit:** it consists in the physical sensors, capable of sensing the changes in a physical condition, such as temperature or pressure variations, humidity, light, gas concentrations, etc. It also has an analog-to-digital converter (ADC), that converts the measured analog data into digital data for processing.
- **The processing unit:** this unit consists of a microprocessor, that is able to process the digital data, and a memory for algorithms and data storage.
- **The communication unit:** this unit allows the sensor to communicate with its neighbors; it is usually a radio transceiver, used especially for short-range communications.

Figure 1.1 shows the architecture of a smart sensor.

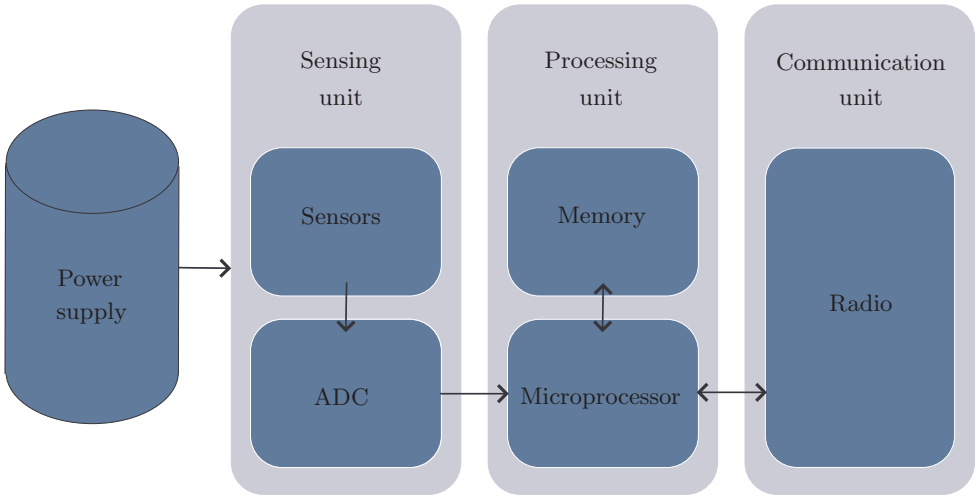


Figure 1.1: Smart sensor architecture

Various kinds of sensors have been developed and made available by different companies over the years. There are mainly three generations of sensors [Chong and Kumar, 2003]. The first generation was introduced in the 1980's, mainly for military purposes. Even though early researchers on sensor networks had in mind large numbers of small sensors, the technology for small sensors was not quite ready. However, in the second generation, that was introduced in the early 2000's, the sensors were more commercialized, and their sizes became a lot smaller than their antecedents. The third and last generation of sensors was introduced in the late 2000's, bringing important improvements to the size, weight, and lifetime of the sensors. Table 1.1 shows the evolution of the sensors' generations over the years.

Table 1.1: Three generations of sensors

	1980's-1990's	early 2000's	late 2000's-present
Manufacturers examples	custom contractors, such as TRSS	commercial, such as Crossbow Technology, Sensoria, Ember	Dust Inc., CubeWorks Inc.
Size	large shoe box and up	pack of cards to small shoe box	dust particle to a few square millimeters
Weight	kilograms	grams	negligible to grams
Sensor architecture	separate sensing, processing and communication	integrated sensing, processing and communication	integrated sensing, processing and communication
Power supply	large batteries	AA batteries	solar
Lifetime	hours, days or longer	days to weeks	months to years

Sensors are highly communication-intensive systems, since they must interact with the physical world and communicate with each other. However, they have limited resources in terms of processing, memory, communication bandwidth, and energy. Therefore, an operating system must be designed in a way to efficiently address the problems of resource management and resource allocation. Several operating systems and routing protocols have been developed for wireless sensors [Al-Karaki and Kamal, 2004; Demirkol et al., 2006]. The most two well-known operating systems are TinyOS and Contiki [Dutta and Dunkels, 2012]. TinyOS was developed at the University of Berkeley [Levis et al., 2004; Levis, 2006], and it has its own programming language called NesC [Gay et al., 2003], which is an extension to the C programming language. As for Contiki, it was developed at the Swedish Institute of Computer Science by Dunkels et al. [2004] and is written in the C programming language. A detailed comparison of both operating systems in terms of power consumption, task management, resource allocation and flexibility is provided in [Reusing, 2012].

1.1.2 Topologies of WSNs

In WSNs, the topology plays an important role on several levels. Choosing the right topology can reduce the amount of communication overhead needed to exchange information between sensors, and thus save energy. It can also help reduce radio interferences and the probability of losing information during the communication process. Therefore, the choice of topology highly depends on the specific application and on the constraints. There are three main types of topologies in wireless sensor networks [Akyildiz et al.,

2002; Bandyopadhyay and Coyle, 2003; Jardosh and Ranjan, 2008; Mourad, 2010; Mamun, 2012], summarized by the following:

- **The centralized topology:** In such a topology, sensors collect measurements and transmit them to a fusion center for processing. Indeed, sensors in this kind of networks are not required to perform complex computations and data processing. The main advantage of the centralized topology is that it can provide high quality processing; however, the transmission of all measurements to the fusion center often results in unnecessary wasteful energy costs and bandwidth consumption, and thereby it reduces the lifetime and the utility of the network.
- **The distributed topology:** Also known as the mesh or flat topology, it treats equally all the sensors. In such a topology, the sensors perform computations and exchange data with their neighboring sensors, located within their communication range. Since information processing is no longer limited to a single fusion center, the network is more robust to failures. However, developing relevant distributed algorithms remains a challenging issue.
- **The hybrid topology:** Also known as the cluster-based topology, it is a combination of the centralized and distributed topologies; thus, it takes advantage of the properties of both previous topologies. Indeed, when adopting such a topology, the network is partitioned into several clusters, each having its own cluster head, that acts like a local fusion center. Information processing is then distributed among several cluster heads, instead of a single fusion center. Note that such a topology is particularly useful for applications that require scalability to hundreds or thousands of sensors.

The three main types of topology are illustrated in Figure 1.2.

1.1.3 Applications

Recently, wireless sensor networks are beginning to be deployed at an accelerated pace in a broad range of applications, ranging from home monitoring [Gomez and Paradells, 2010] to industrial monitoring [Ramamurthy et al., 2007; Salvadori et al., 2009], environmental monitoring [Yu et al., 2005; S. De Vito, 2012], health-care applications [Cote et al., 2003; Honeine et al., 2011], and military ones [Lee et al., 2009]. An overview of the role played by WSNs in such applications is given in the following:

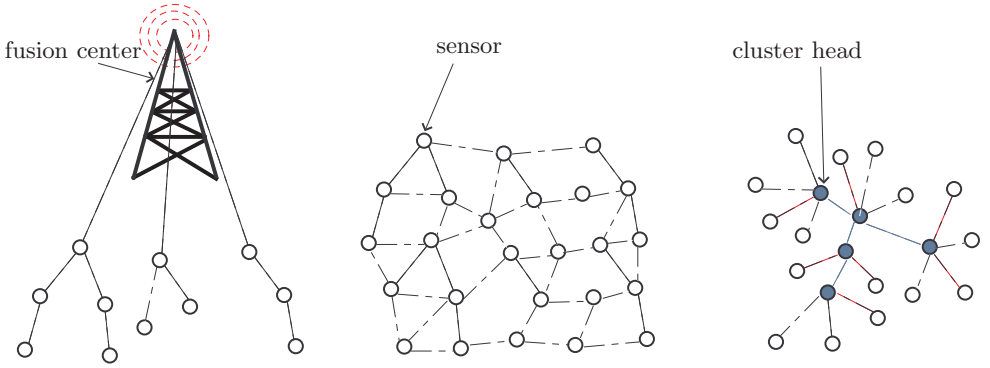


Figure 1.2: Three main topologies of wireless sensor networks

- **Home monitoring:** WSNs play a crucial role in securing and monitoring homes. For example, sensors with motion sensing capabilities have been deployed at windows and doors openings to detect intrusions [Chen and Wang, 2006]. Sensors have been also used to monitor energy and water supply consumption in the aim of saving costs and resources [Trinchero et al., 2011; Maqbool and Chandra, 2013].

- **Industrial monitoring:** WSNs have got significant impacts on the automation and control of industrial processes. They have allowed an increase in production efficiencies by helping for instance the machines to auto-diagnose a malfunctioning [Hou and Bergmann, 2012]. They were also used for monitoring oxygen levels and the leakage of toxic gases, ensuring thus the workers and goods safety [Saeed et al., 2014]. They were also employed to control the temperature inside industrial and medical fridges with sensitive merchandise. An additional application consists in intrusion detection [Shin et al., 2010].

- **Environmental monitoring:** A tremendous progress has been made in terms of environmental monitoring thanks to WSNs. For instance, they have been used for air pollution monitoring [S. De Vito, 2012], and for identifying potential threats such as chemical contamination of water [Chaamwe, 2010]. They were also employed for real-time forest fire detection [Yu et al., 2005], earthquake early detection and volcano monitoring [Lara-Cueva et al., 2014].

- **Health-care applications:** WSNs are revolutionizing the medical domain by providing essential services that can make things easier for the medical staff and improve the life quality of patients [Honeine et al., 2011]. Indeed, body area networks are composed of either wearable sensors or sets of sensors implanted into the human body to measure vital parameters, and thus detect anomalies, such as

heart attack, diabetes and asthma. The information can then be communicated to a device carried by a family member or a medical doctor. The advantage is that even though patients are under continuous supervision, they can move around freely and independently [Zakrzewski et al., 2009]. Such systems are also very useful for tracking the activities of people with Alzheimer disease and for elder-care [Suryadevara and Mukhopadhyay, 2012].

- **Military applications:** WSNs can be efficiently used in battlefields. They can ensure surveillance of regions, assets, perimeters, and borders. They allow tracking of supplies and vehicles, and most importantly tracking of enemies which is helpful for predicting their next movements [Lee et al., 2009; Watthanawisuth et al., 2011]. In addition, sensors could sense polluted areas in case of a chemical, biological, radiological or nuclear (CBRN) attack, thus preventing human exposure in the contaminated areas.

- **Structure health monitoring:** WSNs have been used to monitor vibrations and material conditions in buildings, bridges, and historical monuments [Ong et al., 2008; Bhuiyan et al., 2012]. They can continuously evaluate the health of the structures using several types of sensors capable of collecting different measurements, such as temperature, humidity, corrosion, etc.

This brief list of services that can be provided by WSNs is summarized by Figure 1.3; nevertheless, there are many more applications that have not been mentioned here.

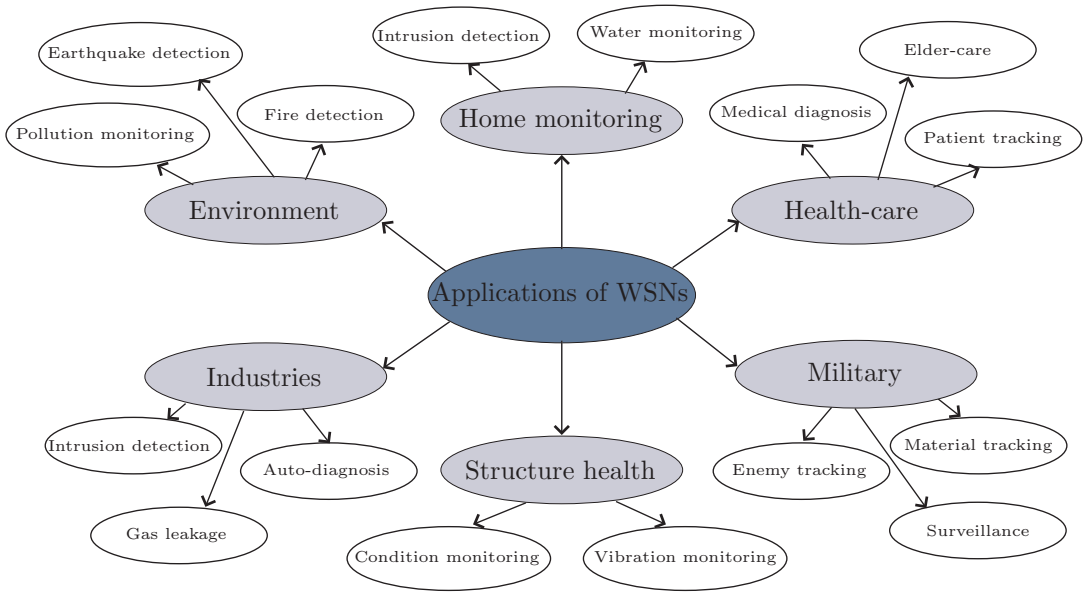


Figure 1.3: Some applications of WSNs

1.2 Localization in Wireless Sensor Networks

In most cited applications, awareness of location information is fundamental since collected data are meaningless without any geographical context. For example, if a fire forest is detected, we need to know where the detection occurred in order to intervene. Therefore, sensor localization has become a fundamental issue, especially in wireless sensor networks, where sensors lack for a fixed infrastructure and are able to move in an uncontrollable manner.

1.2.1 Problem description

Localization plays a key role in many WSNs applications; however, it is often really challenging [Sayed et al., 2005; Patwari et al., 2005], because of the increasing requirements for low cost and high energy efficiency at the sensors side, as well as practical issues associated with network deployments. An intuitive solution to locate the sensors is the use of GPS (Global Position System) devices [B. Hofmann-Wellenhof and Collins, 2004]; however, this solution is impractical for several reasons. Even though GPS is the most widely used technology for localization, it can hardly be applicable to every sensor in the network [Benbadis et al., 2005; Liu et al., 2010]. First, the price and size of a GPS receiver may be prohibitive for many applications, especially for large-scale networks, where the number of sensors may be in the order of thousands. In addition, GPS receivers consume power excessively. The third drawback for using the GPS technology is that it can only be used for outdoor environments, with sufficient sky visibility. Indeed, until now, the GPS service is unavailable or of low quality indoors, underground, such as in tunnels, caves, parkings, under water, and in sky-obstructed outdoor environments, such as forests and urban canyons.

An alternative solution consists in considering two types of sensors, *anchors* and *non-anchor nodes*. Anchors have known locations, whereas non-anchors nodes, or simply *nodes*, have unknown positions and thus need to be localized. In a general setting, we consider the case of moving nodes. The locations of anchors can be either obtained by using GPS devices if possible, or they can be installed at points with known coordinates. The idea is to estimate the locations of nodes by using internodes communications, collected measurements, and/or the information exchanged with the anchors. However, the most important issue remains in choosing what measurements to use, since their accuracy highly affects the localization algorithm's precision. In the following subsection, we examine different types of measurements that can be used for localization, as well as some well-known state-of-the-art localization methods.

1.2.2 Localization measurements and methods

Localization methods can be divided into two categories: range-based and range-free [He et al., 2003]. Range-based methods use angle estimates or distance estimates separating anchors from the sensor nodes to estimate the nodes' locations, while range-free methods make no assumption about the availability or validity of such information. Since sensor nodes are equipped with radios to communicate with each others and with the anchors, locating a node by exploiting radio signals exchanged in the network is becoming of a great interest nowadays. Several types of measurements can be considered for localization in a range-based localization context, such as the received signal strength indicator, the time of arrival, the time difference of arrival, the angle of arrival, etc.

- **Received signal strength indicator (RSSI):** RSSI-based techniques exploit the attenuation of the signal strength with the traveled distance to estimate the distances separating the emitters from the receivers. Typically, anchors broadcast signals in the network, while nodes detect the broadcasted signals and measure their RSSIs. The distances separating nodes from anchors are then estimated using the measured RSSIs and the path-loss model [Medeisis and Kajackas, 2000; Kaemarungsi and Krishnamurthy, 2004; Patwari et al., 2005; Zanella and Bardella, 2014]. RSSI-based techniques exhibit favorable properties with respect to power consumption, size, and cost, since no additional hardware is needed. However, distance estimation using RSSI is really challenging, since the measurements of signals' powers could be significantly altered by the presence of additive noise, multipath fading, shadowing, and other interferences.

- **Time of arrival (ToA):** ToA-based techniques measure the distance between anchors and nodes using signal propagation [Gezici et al., 2005]. In fact, knowing the signal propagation speed, one can directly translate the propagation time into distance. Different signals could be used, such as radio frequency, acoustic, infrared, ultrasound, etc [Pal, 2010]. However, in all cases, the nodes must either have a common clock (one-way ranging approach) or exchange timing information by certain protocols such as a two-way ranging protocol. Note that maintaining time synchronization adds cost and complexity to the network. Moreover, one can get inaccurate distance estimates in the case of non-line-of-sight (NLOS) conditions, i.e., in the case of obstructions between the transmitter and the receiver. Errors in the estimation of the signals' arrival times might also be induced by noise, interferences, clock drifts, and other sources.

- **Time difference of arrival (TDoA):** TDoA-based techniques use the time difference of arrival of signals traveling between two sensors [Okello et al., 2011].

Two different kinds of signals, having each its own propagation speed, are sent at the same time from an anchor. The signals are received at the node, who then measures the difference of their two times of arrival. Using this difference, along with the propagation speeds of the signals, one can easily estimate the distance separating the sensors. However, like the ToA techniques, these techniques yield inaccurate distance estimates in NLOS conditions and in case of imperfect time synchronization, interferences, and noise.

- **Angle of arrival (AoA):** AoA-based techniques measure the angles of arrival of signals exchanged between anchors and nodes [Niculescu and Nath, 2003; Rong and Sichitiu, 2006]. The advantage of such techniques is that they do not require any clock synchronizations. Unfortunately, AoA hardware tends to be more voluminous and more expensive than the hardware needed for the above techniques.

Several geometric methods have been proposed to estimate the node's location using the above measurements. One of the most intuitive and basic methods is the *trilateration* [Manolakis, 1996]. It consists in determining the position of the node based on simultaneous range measurements from three anchors located at known positions as shown in Figure 1.4(a). The node's position is then estimated at the intersection of the three circles having the anchors as centers and the estimated distances as radii. However, since distance measurements are often imperfect, the intersection of these circles does not always result in a single point [Colistra and Atzori, 2012]. To overcome these imperfections, distance measurements from more than three anchors can be used, resulting in a *multilateration* problem [Capkun and Hubaux, 2005; Wang et al., 2009]. As one can see in Figure 1.4(b), where four anchors are considered, the circles intersect at more than one point. These points identify an area of solution, i.e., the area within which the sensor node is expected to be located. Another localization method is the *triangulation* [Esteves et al., 2003, 2006], illustrated in Figure 1.4(c). The triangulation is used when the direction of the node is estimated instead of the distance; in other words, this technique is used when the available measurements are the angles of arrival of signals exchanged in the network. Then, simple geometric relationships from trigonometry are used to estimate the nodes' positions.

The main disadvantage of range-based methods is that they highly depend on the estimated distances that can be inaccurate. For this reason, alternative RSSI-based methods have been proposed for sensor localization. Such methods are range-free, employing for instance connectivity measurements [Y. Shang and Fromherz, 2003]. Instead of estimating exact distances, connectivity-based techniques compare the RSSIs of a considered node to fixed power thresholds in order to detect all its neighboring anchors. Such

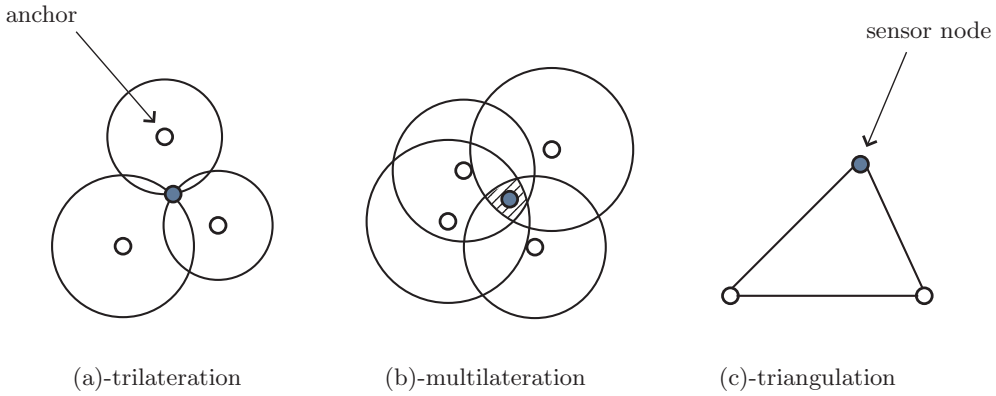


Figure 1.4: Some localization methods

approach assumes that all anchors have ideal circular transmission ranges and that signals' powers decrease monotonically with the increase of the nodes' traveled distances. Moreover, all anchors are considered to have the same transmission range, i.e., an equal transmission radius as shown in Figure 1.5. Localizing a given sensor consists then in computing the intersection of the communication disks of its neighboring anchors. However, this technique can only provide a coarse-grained estimate of each node's location. In addition, the localization error is highly dependent on the node density of the network, the number of anchors, and the network topology [Mao et al., 2007].

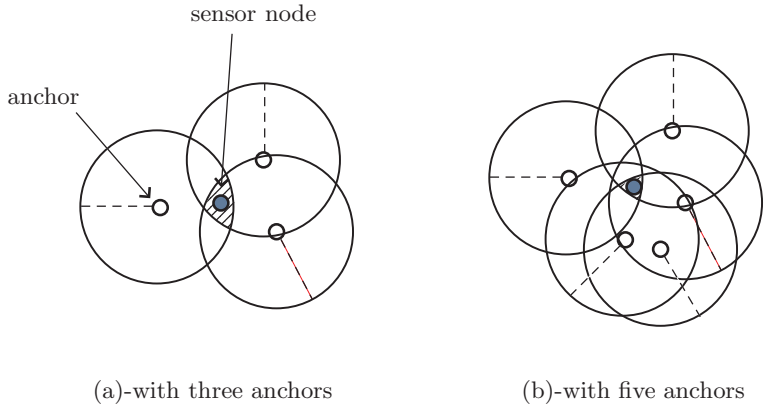


Figure 1.5: Localization by connectivity

Other more reliable techniques for RSSI-based localization rely on (radio-location) fingerprinting [Lin and Lin, 2005; Koyuncu and Yang, 2011]. Compared to connectivity-based methods, these techniques do not make any assumption on the communications in the network. However, they need a pre-configuration phase where a collection of

fingerprints is gathered. Indeed, a typical fingerprinting location sensing system consists of two phases. The first consists in collecting the RSSIs of the signals sent by the anchors and received at several reference locations within the surveillance area. A database of fingerprints is then obtained; it is composed of the set of reference positions with their associated RSSIs. In the second phase, that is the localization phase, nodes move and collect RSSI information, which is then combined with the collected fingerprints database to compute their locations. The advantage of the location fingerprinting technique is that it takes into account the stationary characteristics of the environment, such as multipath propagation, wall attenuation, etc.

1.3 Target Tracking in Wireless Sensor Networks

Target tracking is an interesting research field in WSNs, that consists in estimating instantly the position of a moving target [Li et al., 2002; Dallil et al., 2013]. Nowadays, target tracking is of a great importance for many applications, ranging from traditional ones, such as air traffic control, to emerging applications like supply chain management and wildlife tracking.

1.3.1 Problem description

Target tracking can be viewed as a sequential localization problem, thus requiring recursive location estimation algorithms. Tracking approaches must focus on respecting diverse constraints in wireless sensor networks, such as power-consumption issues, network standards, etc. Note that the tracking problem can be addressed in several ways depending on the application. In fact, tracking systems can be divided into two categories: passive and active. Passive tracking systems store tracking data in memory on the device, and use them later on when needed. These systems are used in cases where immediate knowledge of the target's position or motion is not necessary, whereas active tracking systems are real-time tracking systems that instantly compute the target's position. Here, tracking algorithms must respect additional constraints, such as real-time implementation feasibility. Furthermore, the targets can be either cooperative, i.e., they participate intentionally in the tracking process, or non cooperative, not exchanging data for position estimation.

1.3.2 Target tracking methods

The issue of tracking mobile targets through WSNs is enthusiastically researched and addressed in several works. Proposed algorithms use different types of measurements, such as received signal strength indicators [Lau and Chung, 2007], angle of arrival [Zhang et al., 2013], time difference of arrival [Wendeberg et al., 2012] and time-of-arrival [Xu et al., 2013].

The foundation of tracking is to recursively estimate the target's positions and speeds, which are often referred to as states [Svensson, 2010]. A first step in performing tracking is to make a prediction of the current value of the state. To do so, a model that describes the target's motion is needed. Given knowledge of the previous state, one can then estimate the actual state using the motion model, also called *the state-space model*. For example, consider the problem of tracking a moving vehicle. By knowing its previous position, speed, and direction, one can use the state-space model to estimate its current position. However, the longer the prediction horizon is, the more uncertain the estimation will be. Therefore, this uncertainty needs to be included in the state-space model. In the second step, the estimation is updated using additional information. This information could be any relevant measurement collected from the network. Figure 1.6 shows a target exchanging information, while moving, with the sensors in the network. A relationship must be found between the needed positions and the collected measurements, i.e., we define a measurement model, also called *observation model*. Of course, one must take into account uncertainties in this model as well.

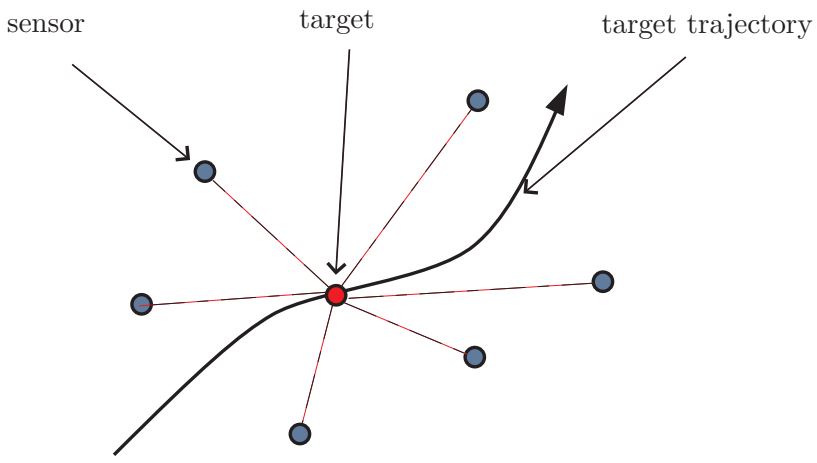


Figure 1.6: Tracking in WSNs

Depending on the type of measurements, one can choose an adequate filter to combine the information from both the state-space model and the observation model. For instance, if the observation model is linear, one can use the Kalman filter [Kalman, 1960; Welch and Bishop, 2001]. This filter first predicts the unknown position using the previous estimated position and the state-space model. Then, in the following step, the predicted position is corrected using the observation model as shown in Figure 1.7. The main limitation of this filter is that it is only reliable for systems that are almost linear. To address the non-linear estimation problem, approaches based on the extended Kalman filter (EKF) [Julier and Uhlmann, 1997] and unscented Kalman filter (UKF) [Julier and Uhlmann, 2004] have been proposed. The main disadvantage of such approaches is that they perform linearization and approximations leading to sub-optimal performance and, sometimes, to a divergence of the filter [UmaMageswari et al., 2012].

Figure 1.7: The Kalman filter

Another well-known filter used for tracking is the particle filter (PF) [Branko Ristic, 2004], also known as the Monte-Carlo filter. The particle filter relies on the minimum mean-square error (MMSE) estimate of the target state given all present and past observations. It seeks to represent the posterior distribution of the hidden states by a properly weighted set of time-varying random samples also called particles. As the number of samples goes to infinity, the weighted average of those samples converges at each time step, in some statistical sense, to the true global MMSE estimate of the current unknown states, given all present and past network measurements [Dias and Bruno, 2012]. For such filters, the observation model can be non-linear, and the initial state and noise distributions can take any required form. Compared to the Kalman-based method, this algorithm needs more computations, due to the generation and resampling of the particles; however, it has more potential in cases where the noise is non-Gaussian and the models are non-linear [Gustafsson et al., 2002].

1.4 Detection and Estimation of a Gas Diffusion

In other contexts, wireless sensor networks are being used for the detection and estimation of pollutions caused by chemical, biological, radiological or nuclear attacks. These pollutions are potent threats to the environment and to human society, wherever they occur. In this thesis, we consider the general case of pollutions caused by gas diffusions. Nevertheless, the proposed solutions could easily be adapted and applied to other types of pollutions. Considering the potentially catastrophic consequences, it is important to detect the gas diffusion and estimate its parameters, including the source's location and the release rate.

1.4.1 Problem description

Gas releases are becoming an increasing danger in today's world. Gas releases can be divided into two categories: deliberate or accidental. Deliberate releases of hazardous chemicals usually occur in war and terrorist attacks. One of the first major attacks happened in Tokyo, Japan in 1995 [Tu, 1999]. Its objective was to inflict terror amongst the population by releasing a highly lethal nerve agent, designed to paralyse and kill within minutes, inside a crowded train subway. As for accidental gas releases, they usually happen in industrial plants, mines, landfill sites, volcanos, etc. One of the world's worst chemical disasters was the gas disaster in Bhopal, India in 1984 [Kalelkar, 1988]. This disaster involved a catastrophic leakage of methyl isocyanate at the Union Carbide Corporation (UCC), a pesticide manufacturing plant in Bhopal. Another recent disaster occurred in 2010 in the Grand Riviera Princess Hotel at Playa Del Carmen, Mexico. Accumulation of methane from a nearby swamp caused a huge explosion at the basement that rendered many deaths and severe financial damages.

1.4.2 Detection and estimation for extreme environmental conditions

Taking emergency actions in case of an instantaneous gas release needs real-time forecasting of the concentration of gas in the environment. This highly depends on the source's parameters such as the source's location, the release rate, etc. However, estimating the source's parameters requires the collection of gas concentration measurements from the area of explosion. Wireless sensor networks have proven to be useful in such scenarios, where human intervention is risky and expensive because of the necessary protective equipments [Christopoulos and Roumeliotis, 2005].

Several European projects were developed within the context of wireless sensor networks for extreme environmental conditions. One example is the project DORSIVA

(2002-2005), that stands for Development of Optical Remote Sensing Instruments for Volcanological Applications. The major goal of this project was to monitor volcanic gas ratios and fluxes of SO_2, using sensors to acquire gas samples at the crater of the volcano. Hence, the activity inside the volcano is monitored without human intervention. Another example is the WINSOC project (2006-2009), that stands for Wireless Sensor Networks with Self-Organization Capabilities for Critical and Emergency Applications. The primary objective of WINSOC was to develop an innovative sensor network architecture having distributed processing capabilities for environmental applications, with focus on landslides detection, gas leakage detection, and large scale temperature field monitoring.

1.5 Organization of the Manuscript and Contributions

This thesis brings several contributions to the domain of wireless sensor networks. Indeed, different approaches have been developed for different types of applications. First, we introduce a new RSSI-based localization technique, developed within the framework of kernel methods. Second, we propose a novel target tracking technique using the already proposed localization technique, the target's inertial information, and the Kalman filter. We also develop a new RSSI/distance propagation model. We then use this model, along with the Kalman filter or the particle filter, to develop a second target tracking approach, that proves to be even more robust than the first one. Finally, in a different context, we propose an original clusterized framework for the detection and estimation of multiple gas sources in wireless sensor networks.

1.5.1 Organization of the manuscript

The rest of the manuscript is organized as follows:

In the second chapter, we give a brief overview of machine learning and kernel methods. Machine learning explores the construction and study of algorithms that infer models and relationships from collected data, without any knowledge of the underlying system. We give some examples of the most used kernels, and then introduce well-known theorems that are investigated throughout this manuscript.

In the third chapter, we introduce an original centralized method combining radiolocation fingerprinting and machine learning for sensors localization in WSNs. The method consists in defining a model, whose inputs and outputs are respectively the received signal strength indicators and the sensors' locations. To define this model,

several kernel-based machine learning techniques are investigated, such as the kernel ridge regression, the support vector regression, and the vector-output regularized least squares. Then, a clusterized version of this method is proposed. Finally, the performance of both centralized and clusterized methods is illustrated using both simulated and real data.

In the fourth chapter, we propose a new method for target tracking in WSNs that combines machine learning with a Kalman filter to estimate instantaneous positions of a moving target. The target's accelerations, along with information from the network, are used to obtain an accurate estimation of its position. The method consists first in using the model defined in the third chapter to obtain a first position estimate of the target under investigation. The Kalman filter is used afterwards to combine predictions of the target's positions based on acceleration information with the first estimates, leading to more accurate ones. The performance of the method is studied for different scenarios, and a thorough comparison to well-known algorithms is also provided.

In the fifth chapter, we first propose two new RSSI/distance models that characterize the relationship between the distances separating sensors and the RSSIs of the signals exchanged by these sensors in a WSN. The first model is a non-parametric regression model, while the second one is a semi-parametric regression model that combines the well-known log-distance theoretical propagation model with a non-linear fluctuation term. Then, using the proposed RSSI/distance models, we present two new tracking approaches. The target's position is estimated by combining acceleration information and the distances separating the target from sensors, using either the Kalman filter or the particle filter. A fully comprehensive study of the choice of parameters for the proposed distance models, as well as a performance analysis of the two proposed tracking methods. Comparisons to recently proposed methods are also provided.

In the sixth chapter, we introduce an original clusterized approach that allows us to detect and estimate the parameters of multiple gas sources in WSNs, including the sources' locations and their release rates. The method consists in regularly collecting concentration measures from the studied region. Then, using these concentrations, a predefined kernel-based machine detects any gas release. Once an alert is raised, the collected concentrations are processed locally in order to estimate the gas release parameters. First estimates of the parameters are given using a kernel-based model that takes as input the collected concentrations. The estimated parameters and the measured concentrations are then used with a second kernel-based model to provide a more accurate estimate of the source location. An evaluation of the performance of the method is provided in the case of a single source, as well as for multiple sources. Moreover, a comparison to state-of-the-art techniques is also given.

Finally, chapter seven provides concluding remarks, a discussion of limitations, and an outlook on future research directions.

1.5.2 Publications

Peer-reviewed international journal articles (2+2)

- **S. Mahfouz**, F. Mourad-Chehade, P. Honeine, J. Farah, and H. Snoussi. "Target tracking using machine learning and Kalman filter in wireless sensor networks". *IEEE Sensors Journal*, vol.14, no.10, pp.3715–3725, October 2014.
- **S. Mahfouz**, F. Mourad-Chehade, P. Honeine, J. Farah, and H. Snoussi. "Kernel-based machine learning using radio-fingerprints for localization in WSNs". *IEEE Transactions on Aerospace and Electronic Systems*, vol.51, no.2, pp.1324–1336, April 2015.
- **S. Mahfouz**, F. Mourad-Chehade, P. Honeine, J. Farah, and H. Snoussi. "Non-parametric and semi-parametric RSSI/distance modeling for target tracking in wireless sensor networks". *IEEE Sensors Journal*, second revision, November 2015.
- **S. Mahfouz**, F. Mourad-Chehade, P. Honeine, J. Farah, and H. Snoussi. "Gas sources parameters Estimation using machine learning in WSNs". Article submitted to *IEEE Sensors Journal* in October 2015.

Peer-reviewed international conference articles (4)

- **S. Mahfouz**, F. Mourad-Chehade, P. Honeine, J. Farah, and H. Snoussi. "Kernel-based localization using fingerprinting in wireless sensor networks". *14th IEEE International Workshop on Signal Processing Advances for Wireless Communications (SPAWC)*. Darmstadt, Allemagne, 16–19 June 2013.
- **S. Mahfouz**, F. Mourad-Chehade, P. Honeine, J. Farah, and H. Snoussi. "Decentralized Localization Using Fingerprinting and Kernel Methods in Wireless Sensor Networks". *21st European Signal Processing Conference (EUSIPCO)*. Marrakech, Maroc, 9–13 September 2013.
- **S. Mahfouz**, F. Mourad-Chehade, P. Honeine, J. Farah, and H. Snoussi. "Ridge regression and Kalman filtering for target tracking in wireless sensor networks". *8th IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM)*. A Coruna, Spain, 22–25 June 2014.

- **S. Mahfouz**, P. Honeine, F. Mourad-Chehade, J. Farah, and H. Snoussi. "Combining a physical model with a nonlinear fluctuation for signal propagation modeling in WSNs". *11th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*. Doha, Qatar, 10–13 November 2014.

Peer-reviewed national conference articles (2)

- **S. Mahfouz**, F. Mourad-Chehade, P. Honeine, J. Farah, and H. Snoussi. "Localisation par fingerprinting et méthodes à noyaux dans les réseaux de capteurs sans fil". *Actes de la 14-ème conférence de la Société française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF)*. France, 13–15 February 2013.
- **S. Mahfouz**, F. Mourad-Chehade, P. Honeine, J. Farah, and H. Snoussi. "Modèle semi-paramétrique RSSI/distance pour le suivi d'une cible dans les réseaux de capteurs sans fil". *Actes du 25-ème colloque du Groupe de Recherche et d'Etudes du Traitement du Signal et des Images (GRETSI)*. France, 8–11 September 2015.

Chapter 2

Machine Learning and Kernel Methods

Contents

In this chapter, our objective is to give a brief introduction of machine learning and kernel methods, that are the basic tools used in this thesis. As we will show, machine learning provides a framework for studying the existing relations between samples from a given training set, i.e., collected data, and determines decision functions in order to make predictions for new samples, thus generalizing the approach. We first give an overview of the statistical learning theory. We then introduce kernel methods, their main properties and their advantages; we also give some examples of the most used kernels. Finally, the well-known representer theorem is detailed, with an illustration using the kernel ridge regression.

2.1 An Overview of Machine Learning Theory

Lately, machine learning techniques have become very popular in the pattern recognition and data mining fields for discovering non-linear relations in data [Vapnik, 1995, 1998; Hofmann et al., 2008]. The goal of a machine learning algorithm is to synthesize functional relationships based on the available data. Thus, a process of induction is used to build up a model of the system, from which it is hoped to deduce responses of the system for newly observed data [Gunn, 1998]. Depending on the structure of the dataset, learning problems can be classified as being supervised, unsupervised, or semi-supervised.

2.1.1 Supervised learning

The supervised learning focuses on modeling a system that is described by its input sample $\boldsymbol{x} \in \mathcal{X}$ and corresponding output (response) $y \in \mathcal{Y}$. The space $\mathcal{X}$ is referred to as the *input space*, and $\mathcal{Y}$ as the *output space*. The objective is then to find the decision function ψ which makes a decision about the system, in the best possible way, based on the observable input state $\boldsymbol{x} \in \mathcal{X}$; in other words, this function predicts the responses of new samples. For instance, in binary classification problems, two classes of data are under competition. The decision function is designed such as to classify new samples based on the training samples. The function's response belongs to the set of labels $\mathcal{Y} = \{-1, +1\}$, and a new sample is either associated with the first class or with the second one. In order to define the function ψ, a common approach is to consider an optimization problem with risk minimization. The expected risk function is given as follows:

$$R(\psi) = \int_{\mathcal{X}\times\mathcal{Y}} \mathcal{L}(\psi(\boldsymbol{x}), y) P(\boldsymbol{x}, y) d\boldsymbol{x} dy,$$

where $P(\boldsymbol{x}, y)$ is the probability distribution of the data pairs $(\boldsymbol{x}, y) \in \mathcal{X} \times \mathcal{Y}$, and $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ represents the loss function that measures the error between the desired output y and the estimated value $\psi(\boldsymbol{x})$ provided by the learning machine.

However, since the distribution $P(\boldsymbol{x}, y)$ is usually unknown, the decision function is derived from a training set that consists of a finite number N of independent and identically distributed observations, namely N input-output pairs $(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i \in \{1, ..., N\}$. We can now replace the expected risk with a good approximation that can be evaluated without the unknown distribution. Then, the expected risk $R(\psi)$ is replaced by the empirical risk $R_{emp}(\psi)$ as follows:

$$R_{emp}(\psi) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\psi(\boldsymbol{x}_i), y_i). \tag{2.1}$$

Now let $\mathcal{F}$ denote the space of all admissible functions ψ for the considered problem; this space is also referred to as the *hypothesis space*. There exists an infinity of functions that can nullify the empirical risk $R_{emp}(\psi)$. However, the minimization of this risk does not always imply minimization of the expected risk $R(\psi)$. Indeed, the hypothesis space is often very large, leading to an arbitrary small empirical risk associated to a high expected risk. This effect is called over-fitting, and the minimization of the gap between the empirical and the expected risk is known as the problem of generalization. This means that, aside from the minimization of the empirical risk, the capacity of the hypothesis space must be controlled. To overcome this problem, one can use the regularization approach initially proposed in [Tikhonov and Arsenin, 1977]. In this way, the search for the optimal function ψ^* is restricted to a space of regular functions. Of particular interest are reproducing kernel Hilbert spaces; details are provided in the following section. Then, the function ψ^* is obtained by minimizing the following regularized risk functional:

$$R_{reg}(\psi) = R_{emp}(\psi) + \eta\Omega(\psi), \tag{2.2}$$

where $\eta > 0$ is called the regularization constant, and $\Omega(\psi) : \mathcal{F} \to \mathbb{R}$ is the regularization term. The first term guarantees that the empirical risk is minimized, while the second term penalizes non-smooth solutions. Finally, the constant η controls the tradeoff between the regularization and the empirical risk.

2.1.2 Unsupervised learning

Differently from supervised learning problems where both inputs and outputs are given, the only available observations for unsupervised learning problems are the samples of

the input space $\mathcal{X}$. Therefore, the objective of unsupervised approaches is to find the hidden relations between the input samples without the external information provided by labels as in the supervised learning. In this case, the expected risk function is given as follows:

$$R(\psi) = \int_{\mathcal{X}} \mathcal{L}(\psi(\boldsymbol{x}))P(\boldsymbol{x})d\boldsymbol{x},$$

where $P(\boldsymbol{x})$ is the probability distribution for any $\boldsymbol{x} \in \mathcal{X}$, and $\mathcal{L}$ is a given loss function. The available training samples are then $\boldsymbol{x}_i$, $i \in \{1, ..., N\}$. In a similar manner to the supervised learning, the expected risk $R(\psi)$ is approximated by the empirical risk $R_{emp}(\psi)$ as follows:

$$R_{emp}(\psi) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(\psi(\boldsymbol{x}_i)).$$

Here as well, one may encounter the over-fitting problem. Using also the regularization approach of [Tikhonov and Arsenin, 1977], we obtain a regularized risk functional $R_{reg}(\psi)$ similar to the one given in (2.2).

2.1.3 Semi-supervised learning

Semi-supervised learning problems fall in between supervised and unsupervised learning problems, and are characterized by the presence of both labeled and unlabeled samples in the training set. Typically, in such problems, we have a small amount of labeled samples with a large amount of unlabeled ones. This is usually due to the fact that obtaining labeled samples is costly, whereas obtaining unlabeled ones is not. The task may then simply be to predict the labels of the unlabeled samples, or to make use of the unlabeled data to improve the ability of the learned function to predict the labels of new samples.

2.2 Kernel Methods

Kernel methods rely on mapping the samples from the input space into a more suitable feature space. They then apply algorithms on the mapped samples based on linear algebra, geometry and statistics in order to discover non-linear patterns and hidden relations existing within the input samples [Shawe-Taylor and Cristianini, 2004; Vert et al., 2004]. It turns out that the applied algorithms can be expressed only as pairwise inner products between the mapped samples. Most importantly, these inner products can be replaced by kernel functions, applied directly to the input data, meaning that no explicit knowledge of the mapping function is needed. In the following, we provide the

definition of kernels and positive definiteness. Next, we give some examples of kernels and explain the rules for constructing new ones.

2.2.1 Positive definiteness

A kernel is a symmetric similarity function defined on the same domain, namely $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that:

$$\kappa(\boldsymbol{x}_1, \boldsymbol{x}_2) = \kappa(\boldsymbol{x}_2, \boldsymbol{x}_1).$$

It is called positive definite if, and only if, for any finite integer N, the following holds:

$$\sum_{i=1}^{N}\sum_{j=1}^{N} c_i c_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0, \quad \forall(\boldsymbol{x}_i, c_i) \in \mathcal{X} \times \mathbb{R}, i \in \{1, \ldots, N\}.$$

Consider now a training set of N entries $\boldsymbol{x}_i \in \mathcal{X}$, $i \in \{1, ..., N\}$. The kernel matrix $\boldsymbol{K}$, also known as the Gram matrix, is the matrix of size $N \times N$ having $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for (i, j)-th entry, $i, j \in \{1, ..., N\}$. This matrix is symmetric, since by definition, we have $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for any two samples $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$. Furthermore, it is positive definite since it satisfies the condition

$$\boldsymbol{c}^\top \boldsymbol{K} \boldsymbol{c} \geq 0, \quad \forall \boldsymbol{c} \in \mathbb{R}^N.$$

2.2.2 Kernels as mapping functions

In this subsection, in order to explain kernels as mapping functions, we start by defining kernels as inner products as in [Vert et al., 2004]. To this end, consider that the input samples $\boldsymbol{x}_i \in \mathcal{X}$, $i \in \{1, \ldots, N\}$, are real vectors, i.e., $\mathcal{X} = \mathbb{R}^p$, $p \in \mathbb{N}_+$. Then, any two vectors $\boldsymbol{x}_i, \boldsymbol{x}_j$ can be compared using their inner product $\boldsymbol{x}_i^\top \boldsymbol{x}_j$. One can prove easily that this inner product between the vectors is a kernel. Indeed, it is symmetric since $\boldsymbol{x}_i^\top \boldsymbol{x}_j = \boldsymbol{x}_j^\top \boldsymbol{x}_i$, and it is positive definite, since we have the following for any N:

$$\sum_{i=1}^{N}\sum_{j=1}^{N} c_i c_j \boldsymbol{x}_i^\top \boldsymbol{x}_j = \left\| \sum_{i=1}^{N} c_i \boldsymbol{x}_i \right\|^2 \geq 0, \quad \forall(\boldsymbol{x}_i, c_i) \in \mathbb{R}^p \times \mathbb{R}, i \in \{1, \ldots, N\}. \tag{2.3}$$

This inner product is then called the *linear kernel*. For more general cases where the input space is not necessarily a vector space, we resort to representing each input $\boldsymbol{x} \in \mathcal{X}$ as a vector $\phi(\boldsymbol{x})$, where ϕ is a mapping function, and then defining a kernel for any $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$ by:

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j), \tag{2.4}$$

Following the same line of computations as in (2.3), one can easily check that κ is a valid (positive definite) kernel on the space $\mathcal{X}$, which is not necessarily a vector space. More generally, following the Moore-Aronszajn theorem [Aronszajn, 1950], for any positive definite kernel κ on some space $\mathcal{X}$, there exists a Hilbert space $\mathcal{H}$ and a mapping

$$\phi\colon \mathcal{X} \longrightarrow \mathcal{H},$$

such that

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle_{\mathcal{H}}, \quad \text{for any} \quad \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}, i, j \in \{1, ..., N\},$$

where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ represents the inner product in the Hilbert space. As one can easily see, the advantage here is that the inner product in $\mathcal{H}$ is evaluated directly from the input data in $\mathcal{X}$ using the kernel function κ, thus without any explicit knowledge of the mapping function ϕ.

Now as we already mentioned, kernel functions map the data from the input space $\mathcal{X}$ into a feature space $\mathcal{H}$. By applying conventional linear models in the former, one can determine non-linear relations and hidden patterns existing among the training samples in the input space. Figure 2.1 illustrates an example of feature mapping, where a non-linear regression in the input space is converted to a linear one in the feature space.

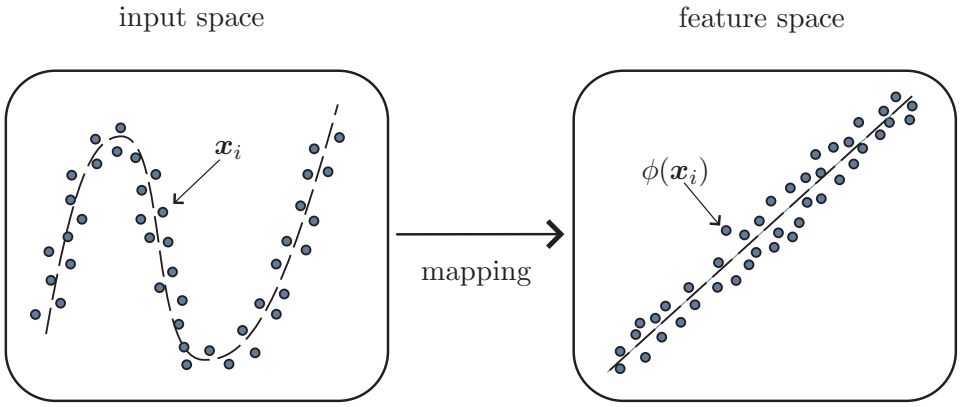


Figure 2.1: Mapping from the input space into the feature space

2.2.3 Reproducing kernel Hilbert space

The feature space $\mathcal{H}$, constructed from a set of training samples $\boldsymbol{x}_i \in \mathcal{X}$, $i \in \{1, \ldots, N\}$, is defined by the set of functions (up to a completion), each one being a linear combination of all the samples in that space, such that

$$\mathcal{H} = \left\{ f : f(\cdot) = \sum_{i=1}^{N} \alpha_i \kappa(\boldsymbol{x}_i, \cdot), \quad \boldsymbol{x}_i \in \mathcal{X}, \alpha_i \in \mathbb{R} \right\}. \tag{2.5}$$

In the following, we define the inner product associated with $\mathcal{H}$, which will lead to an interesting property of the kernel. Let f, g be two functions of the feature space $\mathcal{H}$ given by:

$$f(\cdot) = \sum_{i=1}^{N} \alpha_i \kappa(\boldsymbol{x}_i, \cdot) \quad \text{and} \quad g(\cdot) = \sum_{j=1}^{N} \beta_j \kappa(\boldsymbol{x}_j, \cdot). \tag{2.6}$$

The inner product on $\mathcal{H}$ between any two functions f and g is defined as follows:

$$\langle f, g \rangle_{\mathcal{H}} = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \beta_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{i=1}^{N} \alpha_i g(\boldsymbol{x}_i) = \sum_{j=1}^{N} \beta_j f(\boldsymbol{x}_j), \tag{2.7}$$

where $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$ and $\alpha_i, \beta_j \in \mathbb{R}$. As for the second and third equalities, they come from the definition of f and g. By taking the special case where $g = \kappa(\boldsymbol{x}, \cdot)$ for any $\boldsymbol{x} \in \mathcal{X}$, the inner product between f and g yields the following property:

$$\begin{aligned} \langle f, \kappa(\boldsymbol{x}, \cdot) \rangle_{\mathcal{H}} &= \sum_{i=1}^{N} \alpha_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}) \\ &= f(\boldsymbol{x}). \end{aligned}$$

This property is known as the *reproducing property* of the kernel. Now it is easy to check that κ satisfies the positive definiteness property, as follows:

$$\begin{aligned} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \langle \kappa(\boldsymbol{x}_i, \cdot), \kappa(\boldsymbol{x}_j, \cdot) \rangle_{\mathcal{H}} \\ &= \Big\langle \sum_{i=1}^{N} \alpha_i \kappa(\boldsymbol{x}_i, \cdot), \sum_{j=1}^{N} \alpha_j \kappa(\boldsymbol{x}_j, \cdot) \Big\rangle_{\mathcal{H}} \\ &= \Big\| \sum_{i=1}^{N} \alpha_i \kappa(\boldsymbol{x}_i, \cdot) \Big\|_{\mathcal{H}}^2 \\ &\geq 0, \end{aligned}$$

where $\| \cdot \|_{\mathcal{H}}$ represents the corresponding distance in the feature space $\mathcal{H}$. Given a function κ that satisfies the positive definiteness property, the corresponding space $\mathcal{H}$ is

referred to as its *reproducing kernel Hilbert space (RKHS)*. The advantage of using such a kernel is that we do not need any explicit knowledge of the mapping function ϕ. The key idea is to use linear learning algorithms, expressed only in terms of inner products of the samples, to learn a non-linear function or decision rule in the input space, by replacing the inner product by a positive definite kernel; this is known as the *kernel trick*.

2.2.4 Kernel construction

Due to the Moore-Aronszajn theorem [Aronszajn, 1950], positive definite kernels are reproducing kernels, and vice versa. In the following, we simply denote them by kernels. In this subsection, we state some operations that can be used to construct new kernels from existing ones. These operations preserve the positive definiteness of kernels. For more information on combination rules and properties of kernels, refer to [Shawe-Taylor and Cristianini, 2004].

Let $\kappa_1, \kappa_2 \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be two kernels satisfying the positive definiteness property. Then, the function $\kappa \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a reproducing kernel if, $\forall \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$, it can be written as:

- *a linear combination* of κ_1 and κ_2: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \beta_1 \kappa_1(\boldsymbol{x}_i, \boldsymbol{x}_j) + \beta_2 \kappa_2(\boldsymbol{x}_i, \boldsymbol{x}_j), \forall \beta_1, \beta_2 \in \mathbb{R}_+$,
- *a shifting* of κ_1 or κ_2: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \kappa_1(\boldsymbol{x}_i, \boldsymbol{x}_j) + \ell$, for $\ell \in \mathbb{R}_+$,
- *a product* of κ_1 and κ_2: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \kappa_1(\boldsymbol{x}_i, \boldsymbol{x}_j) \kappa_2(\boldsymbol{x}_i, \boldsymbol{x}_j)$,
- *a power* of κ_1 or κ_2: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \kappa_1(\boldsymbol{x}_i, \boldsymbol{x}_j)^p$, for $p \in \mathbb{N}_+$,
- *an exponential* of κ_1 or κ_2: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(\frac{\kappa_1(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\sigma^2}\right)$, for $\sigma \in \mathbb{R}^*$,
- *a normalization* of κ_1 or κ_2: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\kappa_1(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\sqrt{\kappa_1(\boldsymbol{x}_i, \boldsymbol{x}_i) \kappa_1(\boldsymbol{x}_j, \boldsymbol{x}_j)}}$.

The conservation of the positive definiteness property can be easily shown for all the above properties.

2.2.5 Examples of kernels

As already explained, kernels are often presented as measures of similarity, in the sense that, given any two samples $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$, $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is "large" when $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are "similar" [Vert et al., 2004]. Consequently, kernels are designed in a way to ensure a relevant

measure of similarity in a given context. Moreover, their evaluation should require significantly less computation than would be needed in an explicit evaluation of the corresponding feature mapping ϕ [Shawe-Taylor and Cristianini, 2004]. There are mainly two types of kernels: projective kernels and radial basis functions (RBF) kernels. In the case of the projective kernels, the notion of inner product is used to measure similarity. The RBF kernels measure dissimilarities between samples using distances. In the following, we give some examples of the most commonly used projective and RBF kernels.

2.2.5.1 Projective kernels

The polynomial kernel is the most popular projective kernel. In its general form, the polynomial kernel computes the inner product of all monomials up to degree p as follows:

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(\ell + \boldsymbol{x}_i^\top \boldsymbol{x}_j\right)^p,$$

where $p \in \mathbb{N}_+$, and $\ell \geq 0$. In particular, it is called the *inhomogeneous polynomial kernel* when $\ell > 0$, and the *homogeneous polynomial kernel* when $\ell = 0$. Also notice that when $\ell = 0$ and $p = 1$, we obtain the aforementioned *linear kernel*, that is the ordinary inner product, given by the following:

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^\top \boldsymbol{x}_j.$$

2.2.5.2 RBF kernels

A well known example of an RBF kernel is the *Gaussian radial basis function kernel*, or simply the *Gaussian kernel*, defined as follows:

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right),$$

where $\sigma > 0$ determines its bandwidth. One can see from the kernel's definition that it is a decreasing function of the distance between samples, and therefore has a relevant interpretation as a measure of similarity. Another example of RBF kernel is the *exponential kernel*, also called the *Laplacian kernel*. It is closely related to the Gaussian kernel, with only the square of the norm left out as follows:

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{1}{\sigma}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|\right).$$

2.3 The Representer Theorem

As explained in the first section of this chapter, the objective of machine learning theory is to find a function that characterizes the relationship between training data in the best possible way. We showed that the optimal function is obtained by minimizing a regularized risk functional. This section describes the representer theorem, which is behind the success of the kernel methods. This theorem was introduced by Kimeldorf and Wahba [1971], and a generalized version of it was proposed in [Schölkopf et al., 2001a]. According to this theorem, the solutions of a large class of kernel machines optimization problems can be expressed as kernel expansions over the training samples in the feature space. In the following, we first give an overview of the non-parametric representer theorem; then, we briefly describe the semi-parametric representer theorem.

2.3.1 The non-parametric representer theorem

Consider an input space $\mathcal{X}$, an output space $\mathcal{Y}$, a kernel κ with its reproducing kernel Hilbert space $\mathcal{H}$, and the following training pairs $(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i \in \{1, ..., N\}$. The objective is to find the optimal function ψ^* that minimizes the regularized risk functional of the form given in (2.2). According to the representer theorem, any function $\psi^* \in \mathcal{H}$ minimizing the following regularized risk functional

$$\frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\psi(\boldsymbol{x}_i), y_i) + \lambda \Omega(\|\psi\|_{\mathcal{H}}^2) \tag{2.8}$$

admits a representation of the following form:

$$\psi^*(\cdot) = \sum_{i=1}^{N} \alpha_i \kappa(\boldsymbol{x}_i, \cdot), \quad \forall \boldsymbol{x}_i \in \mathcal{X}. \tag{2.9}$$

In this way, the minimizer $\psi^*(\cdot)$ is expressed as a finite linear combination of the kernels "centered" at the training samples. Note that the monotonicity of Ω is necessary to ensure that the theorem holds. In addition, in order to ensure that the regularized risk functional does not have multiple local minima, the convexity of Ω and of the cost function $\mathcal{L}$ is required.

In order to prove this theorem, we decompose any $\psi(\cdot)$ of $\mathcal{H}$ into a part that lies in the span of the kernels centered at the training samples and a part which is orthogonal to

it as follows:

$$\begin{aligned}\psi(\cdot) &= \psi^{\|}(\cdot) + \psi^{\perp}(\cdot) \\ &= \sum_{i=1}^{N} \alpha_i \, \kappa(\boldsymbol{x}_i, \cdot) + \psi^{\perp}(\cdot) \end{aligned} \tag{2.10}$$

Since $\psi(\cdot)^{\perp}$ is orthogonal to any $\kappa(\boldsymbol{x}_i, \cdot)$ of the training samples, one has $\langle \psi(\cdot)^{\perp}, \kappa(\boldsymbol{x}_i, \cdot) \rangle_{\mathcal{H}} = 0$, $\forall i \in \{1, \ldots, N\}$. Using the latter and (2.10), the evaluation of $\psi(\cdot)$ at an arbitrary training sample $\boldsymbol{x}_j$, where $j \in \{1, \ldots, N\}$, yields

$$\begin{aligned}\psi(\boldsymbol{x}_j) &= \langle \psi(\cdot), \kappa(\boldsymbol{x}_j, \cdot) \rangle_{\mathcal{H}} \\ &= \sum_{i=1}^{N} \alpha_i \, \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) + \langle \psi^{\perp}(\cdot), \kappa(\boldsymbol{x}_j, \cdot) \rangle_{\mathcal{H}} \\ &= \sum_{i=1}^{N} \alpha_i \, \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)\end{aligned}$$

Consequently, the first term of the regularized risk functional (2.8) is independent of $\psi^{\perp}(\cdot)$. As for the second term, since $\psi^{\perp}(\cdot)$ is orthogonal to $\sum_{i=1}^{N} \alpha_i \, \kappa(\boldsymbol{x}_i, \cdot)$, and Ω is strictly monotonic, we get from the Pythagorean theorem:

$$\begin{aligned}\Omega(\|\psi\|_{\mathcal{H}}^2) &= \Omega\Big(\| \sum_{i=1}^{N} \alpha_i \, \kappa(\boldsymbol{x}_i, \cdot) + \psi^{\perp}(\cdot) \|_{\mathcal{H}}^2\Big) \\ &\geq \Omega\Big(\| \sum_{i=1}^{N} \alpha_i \kappa(\boldsymbol{x}_i, \cdot) \|_{\mathcal{H}}^2\Big).\end{aligned}$$

Setting $\psi^{\perp}(\cdot)$ to zero does not affect the fitness term of the regularized risk (2.8), but it strictly reduces the second term; hence, to minimize the expression in (2.8), we must have $\psi^{\perp}(\cdot) = 0$, and the optimal solution has the form given in (2.9). The representer theorem is then proven. It is interesting to note that Argyriou et al. [2008] showed that the representer theorem could also be used in the case of multi-task learning, yielding to vector-output algorithms, such as the vector-output regularized least squares [Honeine et al., 2013]. See succeeding chapter for more details.

2.3.2 The semi-parametric representer theorem

Now in addition to the assumptions considered in the case of the non-parametric representer theorem, we suppose that we are given a set of M real-valued functions $\varphi_j(\cdot)$ of $\mathcal{H}$, $j \in \{1, \ldots, M\}$, with the property that the $N \times M$ matrix, whose (i, j)-th entry is given by $\varphi_j(\boldsymbol{x}_i)$, $i \in \{1, \ldots, N\}$, has rank M, with $N \geq M$. Then any $\tilde{\psi} = \psi + h$, with

$\psi \in \mathcal{H}$ and $h \in \text{span}\{\varphi_j\}$, minimizing the following regularized risk

$$\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(\tilde{\psi}(\boldsymbol{x}_i), y_i) + \lambda\Omega(\|\psi\|^2_{\mathcal{H}})$$

admits a representation of the form

$$\tilde{\psi}(\cdot) = \sum_{i=1}^{N}\alpha_i\kappa(\boldsymbol{x}_i, \cdot) + \sum_{j=1}^{M}\beta_j\varphi_j(\cdot),$$

with $\beta_j \in \mathbb{R}$, $j \in \{1, \ldots, M\}$. By decomposing $\psi(\cdot)$ into two parts here as well, and by following the same line of reasoning as in the previous paragraph, one can prove the semi-parametric representer theorem.

2.4 Example: the kernel ridge regression

In this section, we give an example of kernel methods with the kernel ridge regression, a well-known kernel-based algorithm used for non-linear regression [Saunders et al., 1998]. This algorithm puts to work the representer theorem and several definitions given in this chapter, and it thus provides a simple and basic illustration of kernel methods.

Consider the following training set $(\boldsymbol{x}_i, y_i)$, $i \in \{1, \ldots, N\}$, where N is the size of the training set. We aim at finding the function $\psi(\cdot)$ that takes as input $\boldsymbol{x}_i$ and yields as output y_i. To this end, we consider the minimization of the mean quadratic error between the estimated outputs $\psi(\boldsymbol{x}_i)$ and the desired outputs y_i. Consequently, the loss function $\mathcal{L}$ of the regularized risk given in (2.2) is taken using the mean squared loss and the regularization term $\Omega(\|\psi\|^2_{\mathcal{H}})$ is taken equal to $\|\psi\|^2_{\mathcal{H}}$. Now that the regularized risk is defined, we can write the following problem:

$$\psi = \arg\min_{\underline{\psi}\in\mathcal{H}} \frac{1}{N}\sum_{i=1}^{N}(y_i - \underline{\psi}(\boldsymbol{x}_i))^2 + \eta\|\underline{\psi}\|^2_{\mathcal{H}}, \tag{2.11}$$

where η is a regularization parameter. According to the aforementioned representer theorem, the optimal function $\psi(\cdot)$ can be written as a finite linear combination of kernels evaluated on the input samples as follows:

$$\psi(\cdot) = \sum_{i=1}^{N}\alpha_i\,\kappa(\boldsymbol{x}_i, \cdot). \tag{2.12}$$

By incorporating (2.12) into (2.11), we get the dual optimization problem in terms of $\boldsymbol{\alpha} = (\alpha_1 \dots \alpha_N)^\top$, such that:

$$\boldsymbol{\alpha} = \arg\min_{\underline{\boldsymbol{\alpha}}} \|\boldsymbol{y} - \boldsymbol{K}\underline{\boldsymbol{\alpha}}\|^2 + \eta\, \underline{\boldsymbol{\alpha}}^\top \boldsymbol{K}\underline{\boldsymbol{\alpha}},$$

where $\boldsymbol{K}$ is the $N \times N$ Gram matrix whose (i,j)-th entry is $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$, for $i, j \in \{1, ..., N\}$, and $\boldsymbol{y} = (y_1 \dots y_N)^\top$. The solution of this problem is given by solving the following linear system having N unknowns and N equations:

$$(\boldsymbol{K} + \eta N \mathbf{I})\, \boldsymbol{\alpha} = \boldsymbol{y},$$

where $\mathbf{I}$ is the $N \times N$ identity matrix. Notice that for an appropriate value of the regularization parameter η, the matrix between parenthesis is always non-singular.

2.5 Conclusion

In this chapter, we provided a brief review of the statistical learning theory, and we highlighted some of the fondamental properties of kernels. We also described the concept of reproducing kernel Hilbert space and one of the most important theorems in machine learning: the representer theorem. Kernel methods have been greatly used in the context of function approximation and model fitting. They proved to be successful in solving regression problems, and several machine learning algorithms have been proposed to serve this purpose, such as the kernel ridge regression [Saunders et al., 1998], the support vector regression [Vapnik, 1995] and the vector-output regularized least squares [Honeine et al., 2013]. These algorithms will be detailed in the succeeding chapter. Kernel methods have also been used to solve classification and detection problems. In particular, kernel methods were used to solve one-class classification problems using the one-class support vector machines [Schölkopf et al., 2001b] or the support vector data description [Tax and Duin, 2004]. More details are provided in Chapter 6.

Chapter 3

Kernel-Based Localization

Contents

Localization is an important issue in wireless sensor networks for a very large number of applications. In this chapter, we propose a new localization approach based on radio-location fingerprinting and kernel methods. We then develop a clusterized version of this approach, where several local fusion centers are considered instead of only one global fusion center. Finally, we examine the performance of the approach, in terms of accuracy and time complexity. Moreover, we provide a thorough comparison of the proposed approaches to well-known state-of-the-art localization techniques.

3.1 Introduction

In most wireless sensor networks applications, the data collected by the sensors are only meaningful if they are coupled with the correct locations of the corresponding sensors. Therefore, sensor localization has become a fundamental issue, especially because sensors lack for a fixed infrastructure and are often able to move in an uncontrollable manner. As we already mentioned in Chapter 1, a direct way to locate sensors is the use of Global Positioning System (GPS) devices. However, this solution is not practical because of the high energy consumption of GPS receivers and their limited performance in indoor environments and other poor sky visibility. An alternative solution is to develop localization algorithms that estimate the unknown sensor locations with respect to others having fixed known locations. Two types of sensors are thus defined: those with known positions called *anchors*, and the others having unknown locations called *nodes*. Anchor positions can be obtained, for instance, by setting them at fixed locations with known coordinates, whereas nodes are localized by processing the information exchanged with anchors.

Many localization algorithms using anchors have been proposed. They are mainly based on estimating the distances between the anchors and the nodes. Such methods are either time-based using the time of arrival (ToA) or the time difference of arrival (TDoA) techniques [Okello et al., 2011], angle-based using the angle of arrival (AoA) technique [Rong and Sichitiu, 2006], or power-based by employing the received signal strength indicator (RSSI) [Gholami et al., 2013]. Distance estimates are then combined using, for instance, triangulation [Esteves et al., 2003] or trilateration [Manolakis, 1996] to find nodes positions. Compared to ToA, TDoA, and AoA, RSSI-based methods are being widely used due to their low-power consumption and cost competitiveness, since no extra devices are needed to be integrated within each sensor. However, methods that estimate exact distances using RSSI are not always efficient. This is due to the fact that the signals' powers can be altered by the presence of additive noise, multipath fading, shadowing, and other interferences. Alternative RSSI-based methods for sensor

localization employ connectivity measurements [Y. Shang and Fromherz, 2003]. Such methods do not estimate the distances separating a considered node from anchors, since distance estimates may be inaccurate. They instead compare the RSSIs measured by the considered node to a fixed power threshold in order to set some distance constraints and detect the neighboring anchors of the node. These methods are based on the assumption that all anchors have ideal circular transmission ranges. The node's position is then given by the intersection of the communication disks of its neighboring anchors using, for example, the particle filter [Ozdemir et al., 2009], interval analysis [Mourad et al., 2009, 2011], polar-interval analysis [Mourad et al., 2013], the variational filter [Teng et al., 2010], etc. For instance, Mourad et al. [2009] use intervals to perform an outer approximation of the solution areas leading to boxes guaranteed to include the actual locations of the nodes.

On the other hand, one can also use the fingerprinting technique for RSSI-based localization [Lin and Lin, 2005]. Localization techniques based on fingerprinting are more reliable than connectivity-based ones because they do not make any assumption on the communications in the network. However, they need a pre-configuration phase where a collection of fingerprints is performed to model the network, leading to a database of fingerprints, i.e., a radio map. In the localization phase, RSSI measures collected by any given node are combined with information from the fingerprints database in order to locate the considered node. The advantage of fingerprinting-based techniques is that gathering information from the network allows us to take into account many characteristics of the environment such as multipath propagation, wall attenuation, etc. A well-known algorithm based on location fingerprinting is the weighted K-nearest neighbor (WKNN) algorithm [Koyuncu and Yang, 2011]. In this algorithm, the node position is given by a weighted combination of the K nearest neighboring positions in the database; the nearness indicator for this method is based on the Euclidean distance between the RSSIs collected by the node and the ones in the database. Motivated by the complexity of the RSSI patterns, Kushki et al. [2007] propose a kernelized combination algorithm that compares the RSSI values collected by the nodes with the samples in the database. Moreover, they provide a new method for choosing the training samples to be used in the weighting process, instead of using the K nearest neighboring positions as with the traditional WKNN. Another fingerprinting-based localization method is proposed in [Wu et al., 2007], where the authors consider location estimation as a machine learning problem and use the support vector regression for localization. However, instead of using RSSIs of the signals exchanged in a WSN to construct a database of fingerprints, the authors use signals from the Global System for Mobile communication (GSM) network. It should be noted that such methods can only be used in areas where GSM coverage is available, and that they necessitate a GSM receiver to be integrated within each sensor,

thus limiting the power lifetime of the nodes, as well as the possible range of applications of the method.

In this chapter, we propose an original localization approach based on radio-location fingerprinting and kernel methods. The proposed approach is an anchor-based method carried out first in a centralized scheme. It consists of two phases: a training phase and a localization phase. After the fingerprints collection has been created, the training phase consists in defining a model that associates to the measured RSSIs the exact reference positions where they are collected. In the localization phase, the RSSIs measured by a node are used with the computed model to estimate its position. The model is estimated using kernel-based machine learning algorithms, such as the kernel ridge regression (KRR) [Saunders et al., 1998], the support vector regression (SVR) [Vapnik, 1995] and the vector-output regularized least squares (vo-RLS) [Honeine et al., 2013]. This approach is then extended to a clusterized scheme with several local fusion centers (i.e., cluster heads) instead of only one. As it will be shown later on within the study, the advantages of clustering the location estimation approach are at least two-fold: on one hand, it allows distributing the information processing among several cluster heads; on the other hand, it reduces the probability of network failure by partitioning the flow of information among several regions of the network (instead of being directed towards a single fusion center). Finally, we examine the performance of the proposed framework for different scenarios, and provide a thorough comparison between the different aforementioned learning techniques in terms of accuracy and complexity. Moreover, the proposed approach is compared to several well-known state-of-the-art techniques.

3.2 Problem Statement

The first localization approach proposed in this chapter is centralized, thus the network has the topology illustrated in Figure 1.2(a) on page 6. In such a topology, all collected data are transmitted to the central fusion station, where all processing and computations are conducted. Therefore, sensors only send and receive measurements, and do not perform any computation. More details about computation management will be given in the sequel. The proposed algorithm is based on fingerprinting, and hence it consists of two phases: a training phase and a localization phase. In the first subsection, a description of the network's configuration is given. Then, detailed descriptions of both training and localization phases are provided.

3.2.1 Network configuration

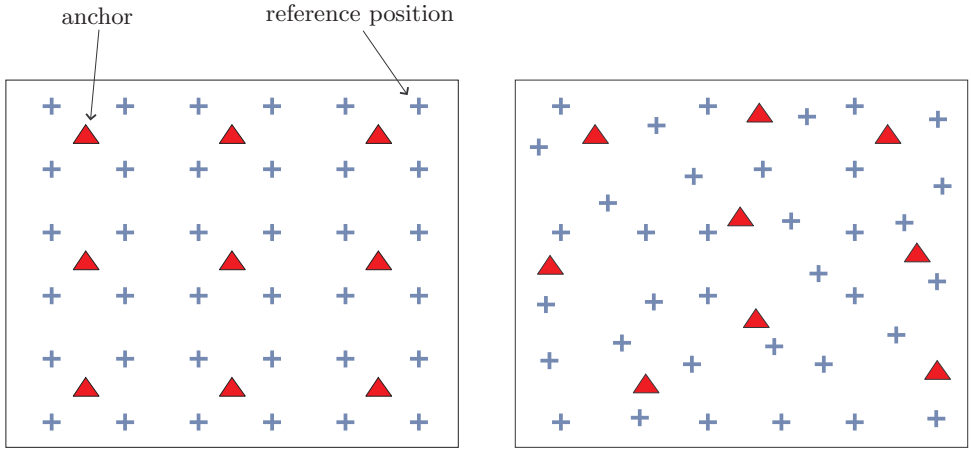
We consider an environment of δ dimensions, with $\delta = 2$ for a two-dimensional environment or $\delta = 3$ for a three-dimensional one. Let $\boldsymbol{a}_i,\ i \in \{1,\dots,N_a\}$, denote the anchors having known fixed locations in the network. On the other hand, mobile nodes are moving with unknown positions, denoted by $\boldsymbol{x}_j,\ j \in \{1,...,N_x\}$, and hence they need to be regularly localized. Without loss of generality, and since nodes are localized independently from each other, using only anchor information, we will withdraw the index j in the sequel. Henceforth, only one mobile node with the unknown position $\boldsymbol{x}$ is considered, knowing that the proposed method can be similarly reproduced for all nodes to be localized. To avoid confusion, it is worth noting that all coordinates are δ-dimension row vectors. We list in Table 3.1 all the variables that will be used in this chapter, along with their respective sizes. Note that N_p and Z will be defined in the following, and ℓ, d, and z are indices taking values respectively in $\{1,\dots,N_p\}$, $\{1,\dots,\delta\}$ and $\{1,\dots,Z\}$.

Table 3.1: List of the used variables in Chapter 3, along with their respective sizes

Notation	Variable	Size
δ	environment dimension	1
N_a	number of anchors	1
N_p	number of reference positions	1
Z	number of clusters	1
$N^{(z)}$	number of reference positions in a cluster z	1
ξ_z	RSSI measure	1
$\boldsymbol{a}_i$	anchor i position	$1 \times\delta$
$\boldsymbol{x}$	node position	$1 \times\delta$
$\boldsymbol{p}_\ell$, $\boldsymbol{P}_{\ell,*}$, $\boldsymbol{p}_\ell^{(z)}$	reference position	$1 \times\delta$
$\boldsymbol{\psi}(\cdot)$, $\boldsymbol{\psi}^{(z)}(\cdot)$	estimated position model	$1 \times\delta$
$\boldsymbol{\alpha}_{\ell,*}$	learning coefficients	$1 \times\delta$
$\boldsymbol{b}$	bias coefficient	$1 \times\delta$
$\boldsymbol{\rho}$, $\boldsymbol{\rho}_\ell$, $\boldsymbol{\rho}_\ell^{(z)}$	vector of RSSI measures	$N_a\times 1$
$\boldsymbol{P}$	matrix of reference positions	$N_p \times \delta$
$\boldsymbol{\alpha}$	matrix of learning coefficients	$N_p \times \delta$
$\boldsymbol{P}_{*,d}$	vector of reference coordinates	$N_p\times 1$
$\boldsymbol{\beta}$	learning coefficients	$N_p\times 1$
$\boldsymbol{v}$	learning vector	$N_p\times 1$
$\boldsymbol{K}$	Gram matrix	$N_p \times N_p$
$\boldsymbol{G}$	learning matrix	$N_p \times N_p$

3.2.2 The training phase

The objective of this phase is to collect information from the network and construct a database of measurements that will be used in the localization phase. To this end, N_p reference positions, denoted by $\boldsymbol{p}_\ell$, $\ell \in \{1, \ldots, N_p\}$, are generated over a uniform grid or randomly in the studied region, as illustrated in Figure 3.1. Then, anchors broadcast signals in the network at a fixed initial power. Meanwhile, a sensor is temporarily placed at each reference position $\boldsymbol{p}_\ell$ to detect the anchor signals and to measure their RSSIs. All anchor signals are assumed to be received at all reference positions. Let $\boldsymbol{\rho}_\ell = (\rho_{\boldsymbol{a}_1,\boldsymbol{p}_\ell} \cdots \rho_{\boldsymbol{a}_{N_a},\boldsymbol{p}_\ell})^\top$ be the column vector of RSSIs sent by all N_a anchors and received at the position $\boldsymbol{p}_\ell$. This way, a database of N_p pairs $(\boldsymbol{\rho}_\ell, \boldsymbol{p}_\ell)$ is obtained, where $\ell \in \{1, \ldots, N_p\}$.



(a)-uniform grid (b)-random distribution

Figure 3.1: Illustration of the fingerprinting configuration

Based on the information from the database, our objective is to define a function

$$\boldsymbol{\psi}(\cdot)\colon \mathbb{R}^{N_a} \to \mathbb{R}^{\delta},$$

that associates to each RSSI vector $\boldsymbol{\rho}_\ell$ the estimated position $\hat{\boldsymbol{p}}_\ell$, as shown in Figure 3.2. Kernel methods provide an elegant framework to find the model $\boldsymbol{\psi}(\cdot)$, as it will be shown in Section 3.3.

3.2.3 The localization phase

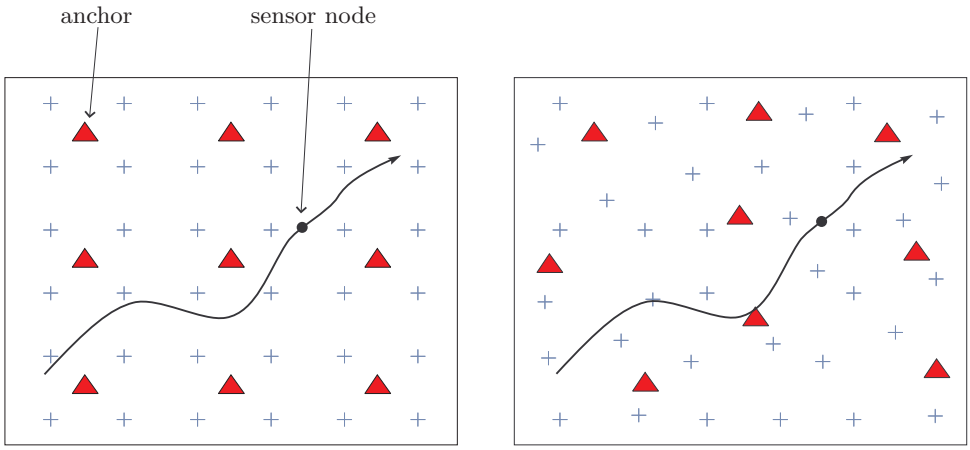
In the localization phase, the defined model $\boldsymbol{\psi}(\cdot)$ is used to estimate the node's position. Indeed, consider the case of Figure 3.3, where a node is moving in the network and needs

Figure 3.2: Function $\boldsymbol{\psi}$

to be localized. This node receives signals from the N_a anchors in the environment, at a given time, measures their RSSI values, and stores them into a vector $\boldsymbol{\rho}$. Its estimated coordinates are then given by:

$$\hat{\boldsymbol{x}} = \boldsymbol{\psi}(\boldsymbol{\rho}).$$

Note that the database construction and the computation of the model $\boldsymbol{\psi}(\cdot)$ are performed only once at the network's fusion center, in the training phase. Then, depending on the type of application and the type of sensors used, the localization process can be conducted at the sensor node itself or at the fusion center. In fact, if the node to be localized has enough computational capabilities and memory for data storage, the fusion center can communicate the model $\boldsymbol{\psi}(\cdot)$ to the moving node, that performs all subsequent computations in the localization phase by itself. Otherwise, if the node does not meet the necessary requirements for such processing, it can send the collected RSSIs to the fusion center, where the localization will then take place.



(a)-uniform grid (b)-random distribution

Figure 3.3: Illustration of the localization phase

3.3 Solving the Localization Problem

In this section, the objective is to find the model $\boldsymbol{\psi}(\cdot)$ that associates to each entry $\boldsymbol{\rho}_\ell$ the corresponding output $\boldsymbol{p}_\ell$, by taking advantage of the kernel-based machine learning. Consider a training set $\{(\boldsymbol{\rho}_\ell, p_{\ell,d})\}_{\ell=1}^{N_p}$, where $d \in \{1, \ldots, \delta\}$, and $p_{\ell,d}$ is the d-th element of $\boldsymbol{p}_\ell = (p_{\ell,1} \ldots p_{\ell,\delta})$. Let $\boldsymbol{\psi}(\cdot) = (\psi_1(\cdot) \ldots \psi_\delta(\cdot))$, where $\psi_d(\cdot)\colon \mathbb{R}^{N_a} \to \mathbb{R}$ estimates the d-th coordinate in $\boldsymbol{p}_\ell = (p_{\ell,1} \ldots p_{\ell,\delta})$, for an input $\boldsymbol{\rho}_\ell$. The functions $\psi_d(\cdot)$ to be determined are now univariate. To illustrate this, see Figure 3.4.

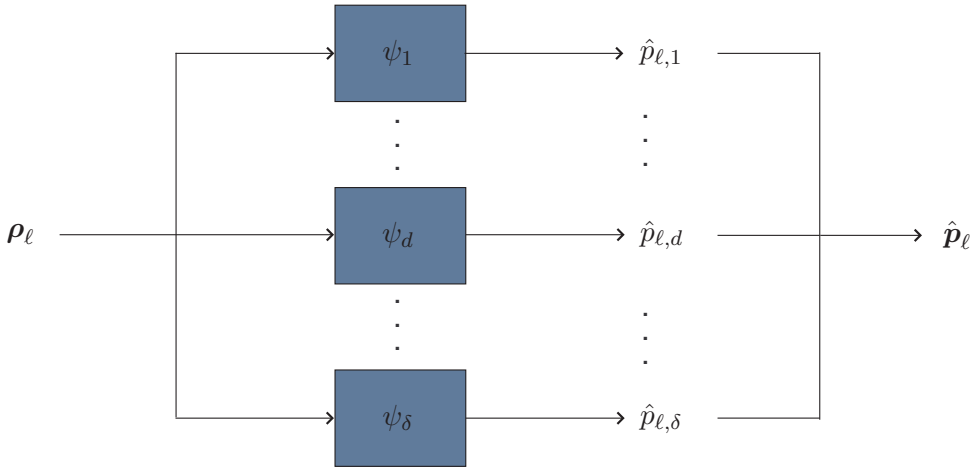


Figure 3.4: Set of functions ψ_d

Consider a reproducing kernel (i.e., positive definite) $\kappa : \mathbb{R}^{N_a} \times \mathbb{R}^{N_a} \to \mathbb{R}$, and denote by $\mathcal{H}$ its reproducing kernel Hilbert space (RKHS) with the induced inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ and norm $\| \cdot \|_{\mathcal{H}}$. From the reproducing property [Aronszajn, 1950], every $\psi_d(\cdot)$ of $\mathcal{H}$ can be evaluated at any $\boldsymbol{\rho}_\ell \in \mathbb{R}^{N_a}$ by

$$\psi_d(\boldsymbol{\rho}_\ell) = \langle \psi_d(\cdot), \kappa(\cdot, \boldsymbol{\rho}_\ell) \rangle_{\mathcal{H}}.$$

The function $\psi_d(\cdot)$ is obtained by minimizing the error between the model's outputs $\psi_d(\boldsymbol{\rho}_\ell)$ and the desired outputs $p_{\ell,d}$, namely by minimizing a regularized empirical risk as follows:

$$\psi_d = \arg\min_{\underline{\psi}_d \in \mathcal{H}} \mathcal{L}\big((p_{1,d}, \underline{\psi}_d(\boldsymbol{\rho}_1)), \ldots, (p_{N_p,d}, \underline{\psi}_d(\boldsymbol{\rho}_{N_p}))\big) + \eta\, \Omega(\|\underline{\psi}_d\|_{\mathcal{H}}^2), \tag{3.1}$$

where $\mathcal{L}$ is an arbitrary cost function, such as the mean squared error, and Ω a strictly monotonically increasing real-valued function on $[0, \infty[$. Here, the second term is a regularization term, with η a positive tunable parameter that controls the tradeoff between the fitness error and the complexity of the solution.

All machine learning algorithms share the same fundamental foundation: the representer theorem [Kimeldorf and Wahba, 1971; Schölkopf et al., 2001a], see Section 2.3 on page 30. It is a key property that underlines the success of the kernel methods, by allowing one to conduct all optimizations in a space whose dimension does not exceed the size of the training set. According to this theorem, it turns out that the problem can be reduced to a much more computation-friendly one. Hence, the minimizer $\psi_d(\cdot)$ of the regularized empirical risk (3.1) is expressed as a finite linear combination of the kernels centered at the training samples $\boldsymbol{\rho}_\ell$. The weights in the linear combination are estimated according to the used cost function $\mathcal{L}$ and regularization term Ω of the general problem (3.1). In fact, the fitness term $\mathcal{L}$ and the regularization term Ω can have different forms, thus leading to different optimization problems and different solutions. In the following first two subsections, we consider two distinct cases of the kernel ridge regression in which the cost function is the mean squared loss. The support vector regression is discussed in the third subsection, where the cost function is taken using the hinge loss. Finally, a multi-tasking approach is considered in the fourth subsection by means of the vector-output regularized least squares algorithm.

For the sake of clarity, we introduce the notations used in the following. Let $\boldsymbol{P} = (\boldsymbol{p}_1^\top \dots \boldsymbol{p}_{N_p}^\top)^\top$. The matrix $\boldsymbol{P}$ is then of size $N_p \times \delta$ having $p_{\ell,d}$ for the (ℓ, d)-th entry, and $\boldsymbol{p}_\ell$ for the ℓ-th row. In the following, we denote the ℓ-th row of $\boldsymbol{P}$ by $\boldsymbol{P}_{\ell,*}$ and the d-th column of $\boldsymbol{P}$ by $\boldsymbol{P}_{*,d}$. The vector $\boldsymbol{P}_{*,d}$ operates now on all N_p points for a fixed coordinate d.

3.3.1 Using the kernel ridge regression

In this subsection, we start by defining the solution using the original form of the optimization problem given in [Saunders et al., 1998], that is the kernel ridge regression. In its original form, the data driven term of (3.1) is taken using the mean squared loss, namely

$$\mathcal{L}\big((p_{1,d}, \psi_d(\boldsymbol{\rho}_1)), \dots, (p_{N_p,d}, \psi_d(\boldsymbol{\rho}_{N_p}))\big) = \frac{1}{N_p} \sum_{\ell=1}^{N_p} (p_{\ell,d} - \psi_d(\boldsymbol{\rho}_\ell))^2.$$

This way, the function $\psi_d(\cdot)$ minimizes the mean quadratic error between the model's outputs $\psi_d(\boldsymbol{\rho}_\ell)$ and the desired outputs $p_{\ell,d}$. As for the regularization term $\Omega(\|\psi_d\|_{\mathcal{H}}^2)$, it is taken in its simplest form, as $\|\psi_d\|_{\mathcal{H}}^2$. Now that the optimization problem is defined, one can write the following:

$$\psi_d = \arg\min_{\underline{\psi}_d \in \mathcal{H}} \frac{1}{N_p} \sum_{\ell=1}^{N_p} (p_{\ell,d} - \underline{\psi}_d(\boldsymbol{\rho}_\ell))^2 + \eta \|\underline{\psi}_d\|_{\mathcal{H}}^2. \tag{3.2}$$

According to the representer theorem, the optimal function $\psi_d(\cdot)$ is of the following form:

$$\psi_d(\cdot) = \sum_{\ell=1}^{N_p} \alpha_{\ell,d}\, \kappa(\boldsymbol{\rho}_\ell, \cdot). \tag{3.3}$$

The problem is now described in a simpler form allowing the weighting coefficients $\alpha_{\ell,d}$ to be determined. Then, let $\boldsymbol{\alpha}$ denote the matrix whose (ℓ, d)-th entry is $\alpha_{\ell,d}$. The d-th column of $\boldsymbol{\alpha}$ is denoted by $\boldsymbol{\alpha}_{*,d}$ and its ℓ-th row by $\boldsymbol{\alpha}_{\ell,*}$.

By substituting (3.3) in (3.2), we get the following dual optimization problem in terms of $\boldsymbol{\alpha}_{*,d}$:

$$\boldsymbol{\alpha}_{*,d} = \arg\min_{\underline{\boldsymbol{\alpha}}_{*,d}} \left(\boldsymbol{P}_{*,d} - \boldsymbol{K}\underline{\boldsymbol{\alpha}}_{*,d}\right)^\top \left(\boldsymbol{P}_{*,d} - \boldsymbol{K}\underline{\boldsymbol{\alpha}}_{*,d}\right) + \eta N_p\, \underline{\boldsymbol{\alpha}}_{*,d}^\top \boldsymbol{K}\underline{\boldsymbol{\alpha}}_{*,d},$$

where $\boldsymbol{K}$ is the $N_p \times N_p$ Gram matrix whose (i,j)-th entry is $\kappa(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j)$, for $i, j \in \{1, ..., N_p\}$. This is a classical quadratic regression problem, whose solution is given by taking the derivative of the above objective function with respect to $\underline{\boldsymbol{\alpha}}_{*,d}$ and setting it to zero:

$$-\boldsymbol{K}\boldsymbol{P}_{*,d} + \boldsymbol{K}^2\boldsymbol{\alpha}_{*,d} + \eta N_p \boldsymbol{K}\boldsymbol{\alpha}_{*,d} = \mathbf{0}^\top,$$

where $\mathbf{0}$ is a row vector of zeros of appropriate size. One can easily find the following form of the solution:

$$\boldsymbol{\alpha}_{*,d} = (\boldsymbol{K} + \eta N_p \mathbf{I})^{-1} \boldsymbol{P}_{*,d}, \tag{3.4}$$

where $\mathbf{I}$ is the $N_p \times N_p$ identity matrix.

Equation (3.4) shows that the same matrix $(\boldsymbol{K} + \eta N_p \mathbf{I})$ needs to be inverted in order to estimate each coordinate. Nevertheless, it is reasonable to collect all δ estimations (δ being the space's dimension) in a single matrix inversion problem, to reduce the computational complexity, by writing:

$$\boldsymbol{\alpha} = (\boldsymbol{K} + \eta N_p \mathbf{I})^{-1} \boldsymbol{P}. \tag{3.5}$$

Then, using model (3.3) and the definition of the vector of functions $\boldsymbol{\psi}(\cdot)$, we define a model that allows us to estimate all δ coordinates at once, as follows:

$$\boldsymbol{\psi}(\cdot) = \sum_{\ell=1}^{N_p} \boldsymbol{\alpha}_{\ell,*}\, \kappa(\boldsymbol{\rho}_\ell, \cdot). \tag{3.6}$$

Now that we have considered a bias-free model, we may also consider an offset in the model as in [Suykens and Vandewalle, 1999], with

$$\boldsymbol{\psi}(\cdot) = \sum_{\ell=1}^{N_p} \boldsymbol{\alpha}_{\ell,*}\, \kappa(\boldsymbol{\rho}_\ell, \cdot) + \boldsymbol{b}, \tag{3.7}$$

where $\boldsymbol{b}$ is the bias parameter of size $1 \times \delta$. Note that this offset can be added in cases where shifting the function by a constant is not penalized [**?**]. Here, shifting $\boldsymbol{\psi}$ by a constant $\boldsymbol{b}$ does not affect the regression problem since the relationship between the positions and the RSSIs remains the same. Therefore, if we need to shift the environment by a constant $\boldsymbol{b}$, we only have to add $\boldsymbol{b}$ to the output given by $\boldsymbol{\psi}$, without having to go through the learning process all over again. If the constant $\boldsymbol{b}$ is unknown, we can estimate it, along with $\boldsymbol{\alpha}$, using the following equations:

$$\begin{pmatrix} 0 & \mathbf{1} \\ \mathbf{1}^\top & \boldsymbol{K} + \eta N_p \mathbf{I} \end{pmatrix} \times \begin{pmatrix} \boldsymbol{b} \\ \boldsymbol{\alpha} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \boldsymbol{P} \end{pmatrix},$$

where $\mathbf{1}$ is a row vector of ones of appropriate size. This is a linear system that can be easily solved in order to compute $\boldsymbol{\alpha}$ and $\boldsymbol{b}$.

3.3.2 Using the kernel ridge regression by regularizing the $\boldsymbol{\alpha}_{*,d}$

In this subsection, the function $\psi_d(\cdot) \in \mathcal{H}$ is taken as in (3.3). However, the regularization term is taken equal to $\|\underline{\boldsymbol{\alpha}}_{*,d}\|^2$ instead of $\|\underline{\psi}_d\|^2_{\mathcal{H}}$. This way, we get the following regularized risk:

$$\min_{\underline{\psi}_d \in \mathcal{H}, \underline{\boldsymbol{\alpha}}_{*,d}} \frac{1}{N_p} \sum_{\ell=1}^{N_p} \Big((p_{\ell,d} - \underline{\psi}_d(\boldsymbol{\rho}_\ell) \Big)^2 + \eta \|\underline{\boldsymbol{\alpha}}_{*,d}\|^2. \tag{3.8}$$

The relationship between the kernel ridge regression using the optimization problem in (3.2) and the minimization of (3.8) will be shown in the following remark. The substitution of (3.3) into (3.8) allows us to write the problem in terms of $\underline{\boldsymbol{\alpha}}_{*,d}$ as follows:

$$\min_{\underline{\boldsymbol{\alpha}}_{*,d}} \left(\boldsymbol{P}_{*,d} - \boldsymbol{K}\underline{\boldsymbol{\alpha}}_{*,d}\right)^\top \left(\boldsymbol{P}_{*,d} - \boldsymbol{K}\underline{\boldsymbol{\alpha}}_{*,d}\right) + \eta N_p \underline{\boldsymbol{\alpha}}_{*,d}^\top \underline{\boldsymbol{\alpha}}_{*,d}.$$

The solution is obtained by taking the derivative of the above objective function with respect to $\underline{\boldsymbol{\alpha}}_{*,d}$ and setting it to zero, namely:

$$-\boldsymbol{K}\boldsymbol{P}_{*,d} + \boldsymbol{K}^2 \boldsymbol{\alpha}_{*,d} + \eta N_p \boldsymbol{\alpha}_{*,d} = \mathbf{0}^\top.$$

This leads to the following form of $\boldsymbol{\alpha}_{*,d}$:

$$\boldsymbol{\alpha}_{*,d} = (\boldsymbol{K}^2 + \eta N_p \mathbf{I})^{-1} \boldsymbol{K} \boldsymbol{P}_{*,d}.$$

By collecting the δ estimations as in the previous subsection, we avoid inverting δ times the matrix $(\boldsymbol{K}^2 + \eta N_p \mathbf{I})$. We then get:

$$\begin{cases} \boldsymbol{\alpha} = (\boldsymbol{K}^2 + \eta N_p \mathbf{I})^{-1} \boldsymbol{K} \boldsymbol{P}, \\ \boldsymbol{\psi}(\cdot) = \sum_{\ell=1}^{N_p} \boldsymbol{\alpha}_{\ell,*} \, \kappa(\boldsymbol{\rho}_\ell, \cdot). \end{cases}$$

Here, we may also consider an offset in the model as in (3.7). We obtain a linear system, which can be easily solved to estimate the unknown variables, given by the following:

$$\begin{pmatrix} 0 & \mathbf{1} \\ \mathbf{1}^\top & \boldsymbol{K}^2 + \eta N_p \mathbf{I} \end{pmatrix} \times \begin{pmatrix} \boldsymbol{b} \\ \boldsymbol{\alpha} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \boldsymbol{K} \boldsymbol{P} \end{pmatrix}.$$

Remark 3.1. Note that the optimization problem in (3.2) and the minimization of (3.8) are connected. In fact, both problems have the form given in (3.1), the fitness term being identical; the difference then resides in the second term, i.e., the regularization. From (3.2), we have:

$$\begin{aligned} \Omega(\|\psi_d\|_{\mathcal{H}}^2) &= \|\psi_d\|_{\mathcal{H}}^2 \\ &= \boldsymbol{\alpha}_{*,d}^\top \boldsymbol{K} \boldsymbol{\alpha}_{*,d}. \end{aligned}$$

Using Rayleigh's inequalities, we write:

$$\lambda_{\min} \boldsymbol{\alpha}_{*,d}^\top \boldsymbol{\alpha}_{*,d} \leq \boldsymbol{\alpha}_{*,d}^\top \boldsymbol{K} \boldsymbol{\alpha}_{*,d} \leq \lambda_{\max} \boldsymbol{\alpha}_{*,d}^\top \boldsymbol{\alpha}_{*,d},$$

where $\lambda_{\min}$ and $\lambda_{\max}$ are respectively the smallest and the largest eigenvalues of $\boldsymbol{K}$. Therefore, we have:

$$\lambda_{\min} \|\boldsymbol{\alpha}_{*,d}\|^2 \leq \|\psi_d\|_{\mathcal{H}}^2 \leq \lambda_{\max} \|\boldsymbol{\alpha}_{*,d}\|^2.$$

By considering these inequalities, one can see that minimizing $\|\boldsymbol{\alpha}_{*,d}\|^2$ results in minimizing $\|\psi_d\|_{\mathcal{H}}^2$ and vice versa. In fact, on one hand, the minimization of (3.8) constrains the norm $\|\boldsymbol{\alpha}_{*,d}\|^2$, which leads to an upper bound on $\|\psi_d\|_{\mathcal{H}}^2$, since $\|\psi_d\|_{\mathcal{H}}^2 \leq \lambda_{\max} \|\boldsymbol{\alpha}_{*,d}\|^2$. On the other hand, in (3.2), the norm $\|\psi_d\|_{\mathcal{H}}^2$ is constrained, providing an upper bound on $\|\boldsymbol{\alpha}_{*,d}\|^2$ as well, since $\lambda_{\min} \|\boldsymbol{\alpha}_{*,d}\|^2 \leq \|\psi_d\|_{\mathcal{H}}^2$.

3.3.3 Using the support vector regression

In this subsection, the support vector regression (SVR) with the ϵ-insensitive loss function, introduced by Vapnik [1995], is used to find the model $\psi_d(\cdot)$. The goal is to find $\psi_d(\boldsymbol{\rho}_\ell)$ that has a maximum deviation of ϵ from the target $p_{\ell,d}$ for all the training data, and that is, at the same time, as flat as possible. In other words, errors are accepted as long as they are less than ϵ, such that $|p_{\ell,d} - \psi_d(\boldsymbol{\rho}_\ell)| \leq \epsilon$. Consequently, the data fidelity term is given by:

$$\mathcal{L}\big((p_{1,d}, \psi_d(\boldsymbol{\rho}_1)), \ldots, (p_{N_p,d}, \psi_d(\boldsymbol{\rho}_{N_p}))\big) = \frac{1}{2N_p} \sum_{\ell=1}^{N_p} \max(0, |p_{\ell,d} - \psi_d(\boldsymbol{\rho}_\ell)| - \epsilon).$$

As for the regularization term, it is set to $\Omega(\|\psi_d\|_{\mathcal{H}}^2) = \|\psi_d\|_{\mathcal{H}}^2$. The two quantities $\eta > 0$ and $\epsilon \geq 0$ are tunable parameters that determine the tradeoff between the regularization and the fit to the training set.

Following [Gunn, 1998; Smola and Schölkopf, 2004], the dual formulation of the optimization problem is given in terms of the Lagrange multipliers $\boldsymbol{\alpha}_{*,d}$ and $\widetilde{\boldsymbol{\alpha}}_{*,d}$ by the following:

$$\max_{\boldsymbol{\alpha}_{*,d}, \widetilde{\boldsymbol{\alpha}}_{*,d}} \sum_{\ell=1}^{N_p} \left(\widetilde{\alpha}_{\ell,d} \left(p_{\ell,d} - \epsilon\right) - \alpha_{\ell,d} \left(p_{\ell,d} + \epsilon\right)\right) - \frac{1}{2} \sum_{\ell=1}^{N_p} \sum_{j=1}^{N_p} (\widetilde{\alpha}_{\ell,d} - \alpha_{\ell,d})(\widetilde{\alpha}_{j,d} - \alpha_{j,d}) \kappa(\boldsymbol{\rho}_\ell, \boldsymbol{\rho}_j),$$

with the following constraints

$$0 \leq \alpha_{\ell,d}, \widetilde{\alpha}_{\ell,d} \leq \frac{1}{2\eta N_p}, \ell \in 1, \ldots, N_p \quad \text{and} \quad \sum_{\ell=1}^{N_p} (\alpha_{\ell,d} - \widetilde{\alpha}_{\ell,d}) = 0. \tag{3.9}$$

Solving the dual optimization problem with the constraints (3.9) determines the Lagrange multipliers, while also having the following form of the regression function:

$$\psi_d(\cdot) = \sum_{\ell=1}^{N_p} (\alpha_{\ell,d} - \widetilde{\alpha}_{\ell,d}) \, \kappa(\boldsymbol{\rho}_\ell, \cdot) + b.$$

One way to compute the offset b is given in [Smola and Schölkopf, 2004] as follows:

$$b = p_{\ell,d} - \sum_{\ell=1}^{N_p} (\alpha_{\ell,d} - \widetilde{\alpha}_{\ell,d}) \, \kappa(\boldsymbol{\rho}_\ell, \boldsymbol{\rho}_\ell) - \epsilon \quad \text{for} \quad \alpha_{\ell,d} \in (0, \frac{1}{2\eta N_p}),$$

$$b = p_{\ell,d} - \sum_{\ell=1}^{N_p} (\alpha_{\ell,d} - \widetilde{\alpha}_{\ell,d}) \, \kappa(\boldsymbol{\rho}_\ell, \boldsymbol{\rho}_\ell) + \epsilon \quad \text{for} \quad \widetilde{\alpha}_{\ell,d} \in (0, \frac{1}{2\eta N_p}).$$

Other techniques for computing the offset b are discussed in [Gunn, 1998; Keerthi et al., 1999; Smola and Schölkopf, 2004].

In matrix form, the dual problem with the constraints in (3.9) can be written as,

$$\max_{\boldsymbol{\alpha}_{*,d},\widetilde{\boldsymbol{\alpha}}_{*,d}} 2\begin{pmatrix} -\epsilon\mathbf{1}^\top + \boldsymbol{P}_{*,d} \\ -\epsilon\mathbf{1}^\top - \boldsymbol{P}_{*,d} \end{pmatrix}^\top \begin{pmatrix} \boldsymbol{\alpha}_{*,d} \\ \widetilde{\boldsymbol{\alpha}}_{*,d} \end{pmatrix} - \begin{pmatrix} \boldsymbol{\alpha}_{*,d} \\ \widetilde{\boldsymbol{\alpha}}_{*,d} \end{pmatrix}^\top \begin{pmatrix} \boldsymbol{K} & -\boldsymbol{K} \\ -\boldsymbol{K} & \boldsymbol{K} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha}_{*,d} \\ \widetilde{\boldsymbol{\alpha}}_{*,d} \end{pmatrix}$$

subject to

$$\begin{pmatrix} \mathbf{1} & -\mathbf{1} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha}_{*,d} \\ \widetilde{\boldsymbol{\alpha}}_{*,d} \end{pmatrix} = 0 \quad \text{and} \quad \boldsymbol{\alpha}_{*,d}, \widetilde{\boldsymbol{\alpha}}_{*,d} \in \left[0, \frac{1}{2\eta N_p}\right]^{N_p}.$$

We now have a quadratic programming problem, whose solution is found using an off-the-shelf optimization technique. For instance, one can use the Matlab function quadprog. Note that this problem has to be solved δ times in order to obtain δ models, one model per coordinate. This results in high complexity in terms of computations and time. In the following subsection, we describe the vector-output regularized least squares, which allows us to define a vector-output model capable of simultaneously estimating all δ coordinates.

3.3.4 Using the vector-output regularized least squares

In the previous algorithms, δ optimization problems are set separately to define δ univariate models $\psi_d(\cdot)$, one per coordinate. In this subsection, a single optimization problem is considered to estimate simultaneously all δ coordinates. To this end, a vector-output model $\boldsymbol{\psi}(\cdot)$ is determined by exploring multi-task learning. Recently, such a learning has become essential for solving a variety of practical problems that necessitate the estimation of vector-output functions. For instance, Micchelli and Pontil [2005] and Evgeniou et al. [2005] presented a framework to study the problem of learning vector-output functions, and the goal was to extend the single-task kernel methods, which have been successfully used in recent years, to multi-task learning. Argyriou et al. [2008] showed that the representer theorem could still be used in the case of multi-task learning. In this subsection, we take advantage of multi-task learning by using the vector-output regularized least squares (vo-RLS) algorithm [Honeine et al., 2013] to estimate all δ coordinates at once. Therefore, we now determine the function $\boldsymbol{\psi}(\cdot)$, whose output is a position vector of dimension δ, without having to separately determine the set of functions $\psi_d(\cdot)$.

In multi-task learning, $\boldsymbol{\psi}(\cdot)$ takes the following form:

$$\boldsymbol{\psi}(\cdot) = \sum_{\ell=1}^{N_p} \beta_\ell \, \boldsymbol{P}_{\ell,*} \, \kappa(\boldsymbol{\rho}_\ell, \cdot),$$

where β_ℓ, $\ell \in \{1, \ldots, N_p\}$, are parameters to be defined. In the following, let $\boldsymbol{\beta} = \left(\beta_1 \ldots \beta_{N_p}\right)^\top$ be the vector whose ℓ-th element is β_ℓ. We then consider the following optimization problem:

$$\min_{\underline{\boldsymbol{\psi}}, \underline{\boldsymbol{\beta}}} \frac{1}{N_p} \sum_{\ell=1}^{N_p} \|\boldsymbol{P}_{\ell,*} - \underline{\boldsymbol{\psi}}(\boldsymbol{\rho}_\ell)\|^2 + \eta \|\underline{\boldsymbol{\beta}}\|^2. \tag{3.10}$$

By substituting the expression of $\boldsymbol{\psi}(\cdot)$ in the above optimization problem, we get the following problem formulation in matrix form:

$$\boldsymbol{\beta} = \arg\min_{\underline{\boldsymbol{\beta}}} \operatorname{Tr}(\boldsymbol{P}\boldsymbol{P}^\top) - 2\boldsymbol{v}^\top \underline{\boldsymbol{\beta}} + \underline{\boldsymbol{\beta}}^\top \boldsymbol{G} \underline{\boldsymbol{\beta}} + \eta N_p \underline{\boldsymbol{\beta}}^\top \underline{\boldsymbol{\beta}}, \tag{3.11}$$

where $\operatorname{Tr}(\cdot)$ is the matrix trace operator, $\boldsymbol{G}$ is the $N_p \times N_p$ matrix whose (j,k)-th entry is given as follows:

$$\boldsymbol{P}_{j,*} \, \boldsymbol{P}_{k,*}^\top \sum_{i=1}^{N_p} \kappa(\boldsymbol{\rho}_j, \boldsymbol{\rho}_i) \, \kappa(\boldsymbol{\rho}_k, \boldsymbol{\rho}_i),$$

and $\boldsymbol{v}$ is the $N_p \times 1$ vector whose j-th entry is given by:

$$\sum_{k=1}^{N_p} \boldsymbol{P}_{j,*} \, \boldsymbol{P}_{k,*}^\top \, \kappa(\boldsymbol{\rho}_j, \boldsymbol{\rho}_k).$$

By taking the gradient of the objective function in (3.11) with respect to $\underline{\boldsymbol{\beta}}$, and setting it to zero, we obtain the following:

$$-\boldsymbol{v} + \boldsymbol{G}\boldsymbol{\beta} + \eta N_p \boldsymbol{\beta} = \boldsymbol{0}.$$

The final solution can then be written as follows:

$$\boldsymbol{\beta} = (\boldsymbol{G} + \eta N_p \mathbf{I})^{-1} \boldsymbol{v}.$$

Having estimated the vector $\boldsymbol{\beta}$, it is now possible to determine the needed model $\boldsymbol{\psi}(\cdot)$. It is worth noting that N_p unknown variables need to be computed in this algorithm, while $N_p \times \delta$ unknown variables need to be found for the previous algorithms. However, in the localization phase, using the vo-RLS model, we need the reference RSSI measures, as well as the reference positions, to localize a node using the model $\boldsymbol{\psi}$ defined in this subsection, whereas for the previous algorithms, only the reference RSSI measures are

needed.

3.4 Clusterized Version of the Method

Several topologies exist for wireless sensor networks. Each topology has its own limitations and its own advantages, as was explained in Chapter 1 Subsection 1.1.2. We proposed earlier a centralized localization approach, where all information are sent to one fusion center or to the sensor node, if the latter has the computational capacities of a fusion center. Information processing is then limited to only one entity, the fusion center or the sensor node. We introduce in this section an extension of the proposed centralized localization approach that allows us to localize sensor nodes in a clusterized framework, where several entities participate in the localization process. We first discuss the motivation behind this extension, then we describe the clusterized approach.

3.4.1 Motivation

As we already explained in Subsection 1.1.2 on page 4, using a centralized scheme provides high quality processing; however, it might reduce the lifetime and utility of the network because of the high transmission cost of all measurements to the fusion center. In clusterized algorithms, local information exchanged between neighboring nodes is processed locally and, hence, information processing is no more limited to one fusion center. Compared to the centralized strategy, the clusterized approach is more robust to failures, since several fusion centers, also called cluster heads, are engaged. It is also less energy consuming and thus more adapted to the WSN limitations. Moreover, such an approach is particularly useful for large-scale sensor networks.

3.4.2 Description of the approach

In order to obtain a clusterized version of the already described localization approach, we propose to partition the region of interest into Z distinct clusters depending on the characteristics of the environment. Each cluster has its own fusion center, that is its cluster head, capable of handling data, performing calculations, and exchanging information with the sensors. These smart devices have fixed known locations and can be placed anywhere in a cluster. Without loss of generality, we consider that clusters are rectangular, each having one cluster head located at its center, as shown in Figure 3.5.

The basic idea now is to apply the centralized localization algorithm locally at every cluster head. Consequently, for the training phase, only the reference positions in the

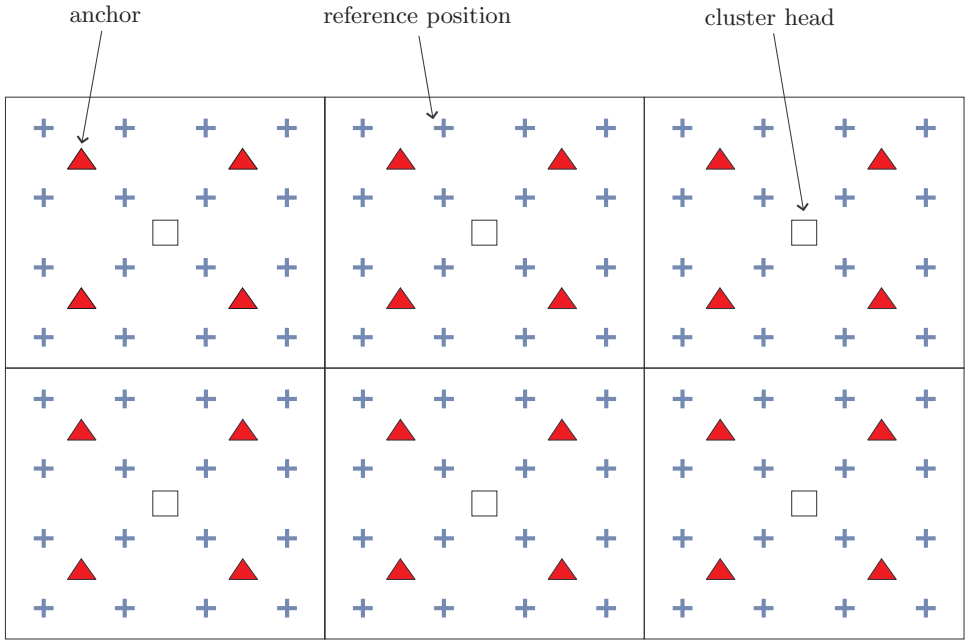


Figure 3.5: Illustration of the clusterized configuration

considered cluster will be taken into account. Let $\boldsymbol{p}_\ell^{(z)}$, $\ell \in 1, \ldots, N^{(z)}$, denote the reference positions in a considered cluster z, $z \in \{1, ..., Z\}$, and let $N^{(z)}$ denote their number. All anchors signals are assumed to be received at all reference positions regardless of their clusters. Let $\boldsymbol{\rho}_\ell^{(z)} = (\rho_{\boldsymbol{a}_1, \boldsymbol{p}_\ell^{(z)}}, \ldots, \rho_{\boldsymbol{a}_{Na}, \boldsymbol{p}_\ell^{(z)}})^\top$ be the RSSI vector of signals received from the N_a anchors at the position $\boldsymbol{p}_\ell^{(z)}$, with $\ell \in \{1, ..., N^{(z)}\}$. Therefore, for each cluster z, a local power map is obtained, composed of $N^{(z)}$ couples $\left(\boldsymbol{\rho}_\ell^{(z)}, \boldsymbol{p}_\ell^{(z)}\right)$. Once the local map is set, the algorithm aims at defining a function $\boldsymbol{\psi}^{(z)}(\cdot)$ that associates to each RSSI vector $\boldsymbol{\rho}_\ell^{(z)}$, for $\ell \in \{1, ..., N^{(z)}\}$, the corresponding position $\boldsymbol{p}_\ell^{(z)}$. The definition of the function $\boldsymbol{\psi}^{(z)}(\cdot)$ is done according to Section 3.3 of this chapter, using any of the aforementioned learning algorithms.

Then, in the localization phase, the node to be localized measures the RSSIs of the signals received from all anchors at a given time, and stores them in the vector $\boldsymbol{\rho}$. Then, the considered node sends its RSSI vector to all the cluster heads located within its communication range. Let $\mathcal{I}$ be the set of indices of the cluster heads detected by the node. Cluster heads referred in $\mathcal{I}$ apply their estimated functions $\boldsymbol{\psi}^{(z)}(\cdot)$ to the RSSI vector $\boldsymbol{\rho}$ to compute local estimates of the node's position as follows:

$$\hat{\boldsymbol{x}}^{(z)} = \boldsymbol{\psi}^{(z)}(\boldsymbol{\rho}), \ z \in \mathcal{I}.$$

Then, a global estimate is given by a combination of all local estimates using:

$$\hat{\boldsymbol{x}} = \sum_{z \in \mathcal{I}} w_z \, \hat{\boldsymbol{x}}^{(z)}.$$

The quantities w_z are weights estimated by using the distances separating the node from the cluster heads. Indeed, the closer the node to a given cluster head, the more reliable the cluster head's estimate and thus the greater the corresponding weight. Since distances to cluster heads are not available, we will use instead the powers of the signals emitted by the nodes while sending their RSSI vectors to the cluster heads. Let ξ_z be the RSSI of the message sent by the node to be localized to the cluster head z. Since signal powers decrease with the increase of their traveled distance, the closer the node to the cluster head z, the greater ξ_z. In order to give more importance to the estimates given by the closest cluster heads to the node, we propose the following expression for the weights w_z:

$$\frac{\exp(\xi_z)}{\sum_{v \in \mathcal{I}} \exp(\xi_v)}, z \in \mathcal{I}. \tag{3.12}$$

By using several fusion centers, the proposed method is more robust to failures compared to the centralized scheme where only one fusion center is considered. Indeed, a failure of the fusion center in the centralized method is fatal, whereas it is much less binding in this clusterized method, since many fusion centers exist. Also, the localization in this method is less energy consuming, since nodes only send information to cluster heads in their sensing range, whereas in the centralized scheme data must be routed to a single fusion center regardless of the distance to be traveled. Moreover, in the training phase, less computations are needed, since N_p reference positions are considered for the global model in the centralized approach, while $N^{(z)} < N_p$ reference positions are considered for the local model assigned to a given cluster z. In addition, the computations are less complex, especially for the matrix inversion process; indeed, the Gram matrix $\boldsymbol{K}$ is of size $N_p \times N_p$ for the centralized approach, whereas in the clusterized approach, the local Gram matrices are of size $N^{(z)} \times N^{(z)}$. As for the localization phase, we need less memory storage for the clusterized approach. In fact, in the localization phase, we only need the reference RSSI measures and the learning coefficients to estimate a node's position. In the centralized approach, N_p reference RSSI measures and $N_p \times \delta + 1$ learning coefficients are stored in the global fusion center, whereas in the clusterized approach, $N^{(z)} < N_p$ reference measures and $N^{(z)} \times \delta + 1$ learning coefficients are stored in the cluster head of a given cluster z.

3.5 Simulation and Experimental Results

In this section, we first highlight the performance of the centralized method applied for different learning algorithms. To this end, we expose the results obtained when using each of the learning algorithms described in Section 3.3, for different scenarios. In the first subsection, the proposed centralized method is tested on simulated data, and results are compared for different machine-learning techniques. In the second subsection, we test the performance of our method using real data gathered in a 10 m $\times$ 10 m real indoor environment [Zanca et al.]. In the third subsection, we compare the centralized localization approach to its clusterized version for both real and simulated data. In the final subsection, the results obtained with the centralized and clusterized methods are compared to the ones obtained when performing localization using connectivity information, as well as the weighted K-nearest neighbor (WKNN) algorithm and the method that uses kernelized calculation proposed in [Kushki et al., 2007].

In the following, we consider the Gaussian kernel (Subsection 2.2.5 on page 28) given by:

$$\kappa(\boldsymbol{\rho}_s, \boldsymbol{\rho}_{s'}) = \exp\left(\frac{-\|\boldsymbol{\rho}_s - \boldsymbol{\rho}_{s'}\|^2}{2\sigma^2}\right), \tag{3.13}$$

where σ is the bandwidth of the Gaussian kernel. This quantity, together with the regularization parameter η, controls the degree of smoothness, noise tolerance, and generalization of the solution. It is important to mention that the choice of the regularization parameter η and the kernel parameters are performed using the cross-validation technique. This approach is a statistical method that consists in dividing data into two segments: one for training the model and the other one for validating it [Stone, 1974]. The k-fold cross-validation, which is the basic form of cross-validation, is used here; it consists in partitioning the data into k roughly equally sized folds. Subsequently, k iterations of training and validation are performed such that, within each iteration, $k-1$ folds are used for learning and the remaining one for validation. In each iteration, the error on the validation set is computed for different values of the tuning parameters. Then, the values of the parameters that give a minimum average error for all iterations are retained. In the following, the cross-validation technique finds the optimal values of the parameters η and σ with a grid search over $\eta N_p = 2^s$ with $s \in \{-20, -19, \cdots, -1\}$ and $\sigma = 2^{s'}$ with $s' \in \{1, 2, \cdots, 10\}$.

3.5.1 Evaluation of the centralized method on simulated data

In this subsection, we evaluate the performance of the centralized method, in terms of accuracy, using simulated data. To this end, we consider a 100 m $\times$ 100 m area, with

16 static anchors and 100 reference positions on a uniformly grid over the area. The RSSI values are obtained using the well-known path-loss propagation model [Medeisis and Kajackas, 2000] given by:

$$\rho_{\boldsymbol{a}_i,\boldsymbol{p}_\ell} = \rho_0 - 10\, n_P \, \log_{10} \|\boldsymbol{a}_i - \boldsymbol{p}_\ell\| + \varepsilon_{i,\ell}, \tag{3.14}$$

where $\rho_{\boldsymbol{a}_i,\boldsymbol{p}_\ell}$ (in dBm) is the power received from the anchor at position $\boldsymbol{a}_i$ by the node at position $\boldsymbol{p}_\ell$, that is the i-th entry of the vector $\boldsymbol{\rho}_\ell$, ρ_0 is the power at a distance of 1 m set to 1 dBm, n_P is the path-loss exponent set to 4 (as often given in the literature), $\|\boldsymbol{a}_i - \boldsymbol{p}_\ell\|$ is the Euclidian distance between the position $\boldsymbol{p}_\ell$ of the considered node and the anchor position $\boldsymbol{a}_i$. Finally, $\varepsilon_{i,\ell}$ is the noise affecting the RSSI measures with $\sigma_{\boldsymbol{\rho}}$ its standard deviation.

Then, we generate the trajectory given in Figure 3.6. Using the defined RSSI model, we determine the RSSI measures of the signals exchanged between the moving node and the anchors, i.e., $\boldsymbol{\rho}$. The trajectory is estimated using different machine-learning algorithms (in the case of bias-free models), for noiseless RSSI measures. The estimation errors (in meters), measured by the root mean squared distance between the exact positions and the estimated ones, are shown in Table 3.2. We notice that the kernel ridge regression and the vo-RLS yield close results, and that the best result is obtained when using the kernel ridge regression from Section 3.3.1, with an estimation error of 0.17 m. Then, to evaluate the robustness of our method against noise, we add a zero-mean Gaussian white noise of standard deviation $\sigma_{\boldsymbol{\rho}}$ =1 dBm to the RSSI values. The obtained estimation errors are averaged over 50 Monte-Carlo simulations for the different learning algorithms, and are shown in Table 3.3. We notice that the best estimations are also obtained when using the kernel ridge regression and the vo-RLS. In the following, we evaluate the time complexity and the memory consumption of the centralized method when using each of the aforementioned learning algorithms. We then study the influence of the number of anchors N_a and the number of reference positions N_p on the performance of the method, i.e., on the estimation error.

3.5.1.1 Evaluation of the time complexity

In the aim of evaluating the time complexity of our method when using each of the aforementioned learning algorithms, we measure the elapsed time for the training phase for each one of them. Simulations are run on version 7.10.0.499 of Matlab on a Dell laptop with Windows 7 and Intel Core i7 CPU. The measured time intervals are stored in Table 3.4. One can see that the training phase for the kernel ridge regression is the fastest; indeed, its elapsed time is around 8 milliseconds, while 15 seconds are needed for

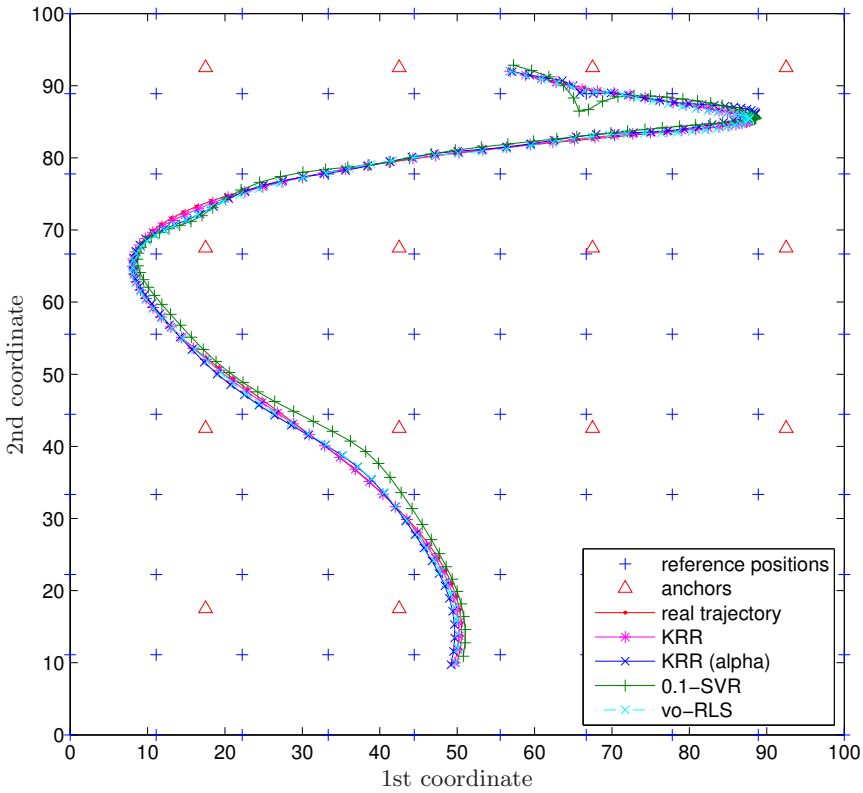


Figure 3.6: Estimation of the trajectory in the absence of noise and with a grid distribution

the training phase of the ϵ-SVR. Note that the high training time needed for the ϵ-SVR can be explained by the fact that the function quadprog is used to solve the problem, as we already explained in Subsection 3.3.3. It is also worth noting that finding the optimal values for the tuning parameters η and σ is the most complex part in terms of computations and time, since one has to repeat the training phase a number of times depending on the number of folds considered and on the range of the variables. The measured time intervals for finding the optimal values of η and σ, using a 10-fold cross-validation in our case, are stored in Table 3.5. It is easy to see that the ϵ-SVR has the worst performance when it comes to studying time complexity and also estimation error. Finally, Table 3.6 shows the elapsed time needed to estimate a trajectory point in the localization phase. Here, one can see that all learning algorithms yield fast estimation times.

Table 3.2: Estimation errors (in meters) for the different techniques, using noiseless simulated data

Learning algorithm	Mean error
KRR	**0.17**
KRR with bias	**0.17**
KRR (α)	0.59
KRR (α) with bias	0.70
0-SVR	2.22
0.1-SVR	1.10
0.25-SVR	1.72
0.5-SVR	3.13
1-SVR	1.37
2-SVR	2.23
vo-RLS	**0.45**

Table 3.3: Estimation errors for the different techniques, using noisy simulated data

Learning algorithm	Mean error
KRR	**1.88**
KRR with bias	**1.87**
KRR (α)	**1.92**
KRR (α) with bias	**1.95**
0-SVR	2.46
0.1-SVR	2.69
0.25-SVR	2.67
0.5-SVR	2.35
1-SVR	2.42
2-SVR	2.36
vo-RLS	**2.09**

Table 3.4: Elapsed training time for the different techniques

Learning algorithm	Elapsed time
KRR (all models)	8 ms
ϵ-SVR	15 000 ms
vo-RLS	25 ms

3.5.1.2 Evaluation of the memory consumption

Now, to assess the memory consumption of the proposed methods, we consider the number of necessary variables in the localization phase, i.e., the number of variables that are stored in the fusion center to be used for the localization process. Table 3.7 shows the number of variables for all of the considered training algorithms. One can see that the bias-free kernel ridge regression necessitates the smallest number of variables in

Table 3.5: Elapsed cross-validation time for the different techniques

Learning algorithm	Elapsed time
KRR (all models)	16 s
ϵ-SVR	more than five hours
vo-RLS	50 s

Table 3.6: Elapsed position estimation time for the different techniques

Learning algorithm	Elapsed time
KRR (all models)	0.3 ms
ϵ-SVR	1.5 ms
vo-RLS	1.7 ms

the localization phase, while the highest number of variables needed is for the vo-RLS, since N_p is significantly higher than δ.

Table 3.7: Number of stored variables for the different techniques

Learning algorithm	Number of variables
KRR	$N_p(N_a + \delta) + 1$
KRR with bias	$N_p(N_a + \delta) + \delta + 1$
ϵ-SVR	$N_p(N_a + \delta) + \delta + 1$
vo-RLS	$N_p(N_a + \delta + 1) + 1$

3.5.1.3 Infuence of N_a and N_p

We now study the effect of the number of anchors and the number of reference positions on the performance of the method. We consider noiseless data and the same scenario as the one used in Figure 3.6. The number of reference positions N_p is fixed and set to 100, and we vary the number of anchors, such that $N_a = 1^2, 2^2, \ldots, 20^2$. In this paragraph, we consider a bias-free model obtained using the kernel ridge regression learning algorithm of Subsection 3.3.1, since this model yields the best results with noiseless data as shown before. Figure 3.7 shows the evolution of the estimation error in terms of the number of anchors. For the results in Figure 3.8, we consider the same settings, and fix the number of anchors to 16, but vary the number of reference positions, $N_p = 5^2, 6^2, \ldots, 25^2$. By comparing the obtained results, one can notice that the changes in the number of anchors do not affect the estimation error as much as the changes in the number of reference positions. Indeed, increasing the number of reference positions allows a better coverage and a better knowledge of the environment, which explains the improvement in the results. For instance, for $N_p = 25^2 = 625$ and $N_a = 16$, we get a low estimation error of 0.0045 m, but with a significant increase in the algorithm's

complexity, compared to the case where $N_p = 25$ for example. Therefore, depending on the practical system constraints, a tradeoff should be found between the algorithm's accuracy and the computational load.

Also, notice the irregularities in the curve representing the estimation error as a function of the number of anchors. In fact, the high positioning error is actually related to the proximity of the trajectory to the anchors. Indeed, with very small distances between the node and the anchors, the RSSIs get higher and also highly vary even with small distance variations. This is due to the logarithmic relationship between the distance and the RSSI. Now with some anchors configurations, if the node comes very close to an anchor, the error on its estimated position could get high and thus it affects the whole estimation error, which was the case with 225 and 250 anchors, where the estimated trajectory deviates from the original one at really close positions to anchors. A possible solution to this problem could be a dynamic selection of the anchors to be included in each position estimation, using, for instance, a predefined threshold on the power level of the signals received by the node from the anchors (to replace the unavailable distance information). However, such solution would only be appropriate when the estimation is performed at a fusion center with sufficient resources, and not at the sensor node, because of the significant increase in the computational load.

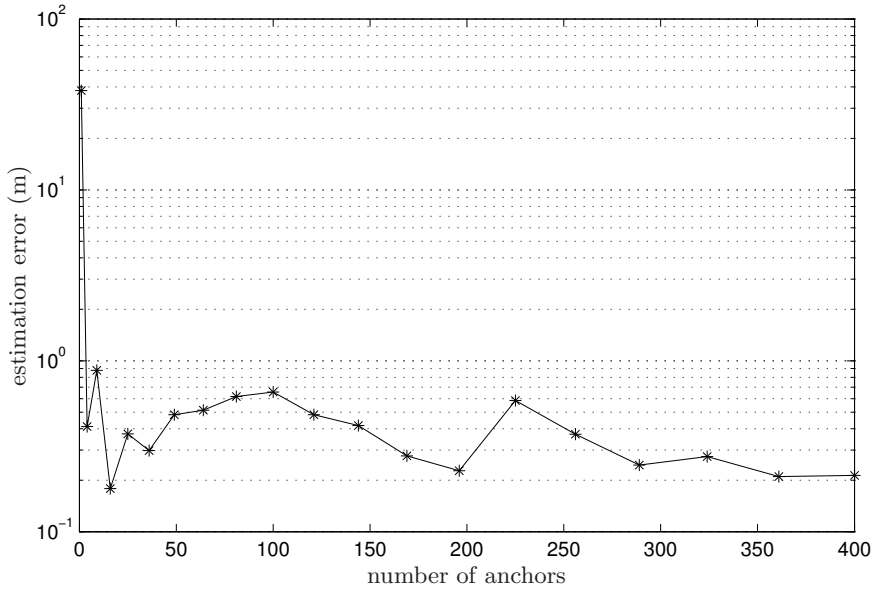


Figure 3.7: Estimation error as a function of the number of anchors

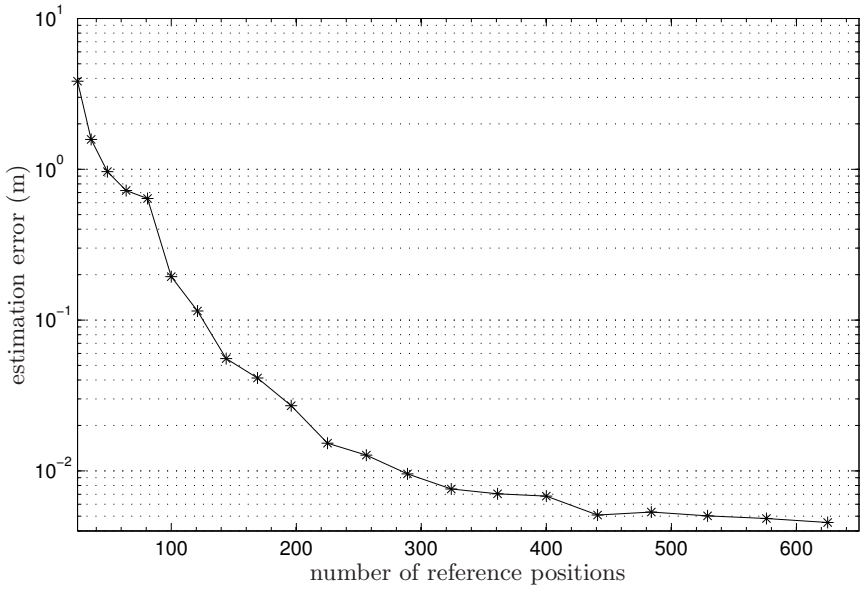


Figure 3.8: Estimation error as a function of the number of reference positions

3.5.1.4 Influence of choosing a random distribution

We now consider a random uniform distribution of the 16 anchors and the 100 reference positions, instead of a uniform grid. We repeat the experiment 50 times for the kernel ridge regression algorithms and only once for the 0.1-SVR. The 0.1-SVR is chosen for comparison because it yields one of the lowest estimation errors in the noiseless setup, as shown in Table 3.2. However, the experiment is done only one time because of the huge time complexity of this algorithm. The mean estimation errors (in meters) are shown in Table 3.8 in the case of noiseless data, and in Table 3.9 for noisy data, with an additive zero-mean Gaussian white noise with standard deviation equal to 1 dBm. In both tables, σ_{MSE} is the standard deviation of the mean estimation error. Figure 3.9 shows the generated trajectory and the estimated one using noiseless RSSIs and a random distribution. Compared to the results obtained when the anchors and reference positions are over uniform grids, one can see that the estimation error increases with the use of random distributions. This can be explained by the fact that a uniform grid allows a better coverage of the region of interest, while a random distribution does not always guarantee a good coverage of the region. Nevertheless, the results are still satisfactory, and random distributions can still be used for accurate localization when uniform grids are not applicable.

Table 3.8: Estimation errors for the different techniques with a random distribution and for noiseless simulated data

Learning algorithm	Mean error	σ_{MSE}
KRR	0.33	0.23
KRR with bias	0.45	0.40
KRR (α)	1.13	0.61
KRR (α) with bias	1.14	0.52
0.1-SVR	2.62	-
vo-RLS	1.25	0.27

Table 3.9: Estimation errors for the different techniques with a random distribution and for noisy simulated data

Learning algorithm	Mean error	σ_{MSE}
KRR	2.71	1.09
KRR with bias	2.55	0.82
KRR (α)	2.87	1.63
KRR (α) with bias	2.55	0.57
0.1-SVR	2.83	-
vo-RLS	3.46	0.82

3.5.2 Evaluation of the centralized method on real data

In this subsection, we study the performance of the centralized localization approach in the case of real collected data, for the different learning algorithms. The set of collected measurements used in this study are available from [Zanca et al.]. The measurements are performed in a room of approximately 10 m $\times$ 10 m, where 48 EyesIFX nodes are deployed over a uniform grid. Furniture and people in the room cause multi-path interferences affecting the collected RSSI values. Five nodes are chosen to be anchors, leaving us with 43 nodes to use as reference positions, that is 43 RSSI/position pairs $(\ddot{\boldsymbol{\rho}}_i, \ddot{\boldsymbol{p}}_i)$, $i \in \{1, 2, \ldots, 43\}$. To get better results in the training process, we generate additional reference positions in order to get a total of 100 reference positions. This is done using a weighting function that relies on the Euclidian distance between the existing points and the new ones, such that $\boldsymbol{\rho}_\ell = \sum_i^{43} w_i \ddot{\boldsymbol{\rho}}_i$, with $w_i = \frac{\exp(-\|\boldsymbol{p}_\ell - \ddot{\boldsymbol{p}}_i\|)}{\sum_{j=1}^{100} \exp(-\|\boldsymbol{p}_\ell - \ddot{\boldsymbol{p}}_j\|)}$. We generate the trajectory in the same manner, and apply the proposed method to estimate the node's position as shown in Figure 3.10. Table 3.10 shows the estimation errors, in meters, for the different learning algorithms. The lowest estimation error is again obtained with the kernel ridge regression of Subsection 3.3.1. The other results are also satisfactory. However, we should recall that the ϵ-SVR does not allow a rapid localization because of its high complexity in terms of computation time.

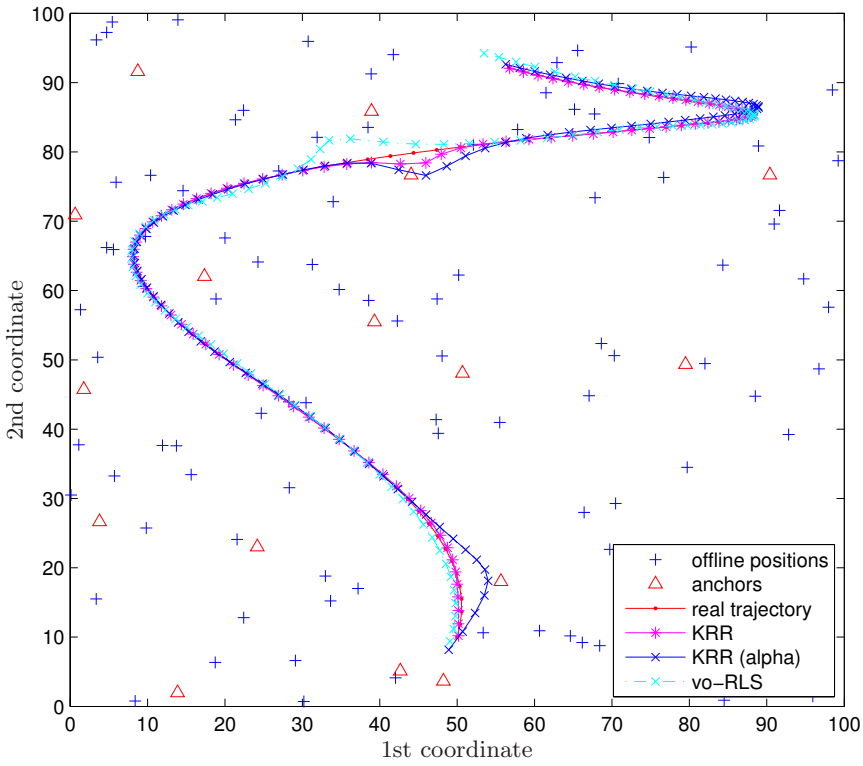


Figure 3.9: Estimation of the trajectory in the absence of noise and with a random distribution of the anchors and reference positions

Table 3.10: Estimation errors for the different techniques, using real data

Learning algorithm	Mean error
KRR	0.34
KRR with bias	0.35
KRR (α)	0.42
KRR (α) with bias	0.56
0.1-SVR	0.41
0.25-SVR	0.53
0.5-SVR	0.62
vo-RLS	0.41

3.5.3 Centralized vs clusterized estimation

In this subsection, we consider the clusterized localization approach introduced in Section 3.4, and evaluate its performance in the case of simulated data and then for real data. We then compare the results to the ones obtained with the centralized approach when using the kernel ridge regression learning algorithm.

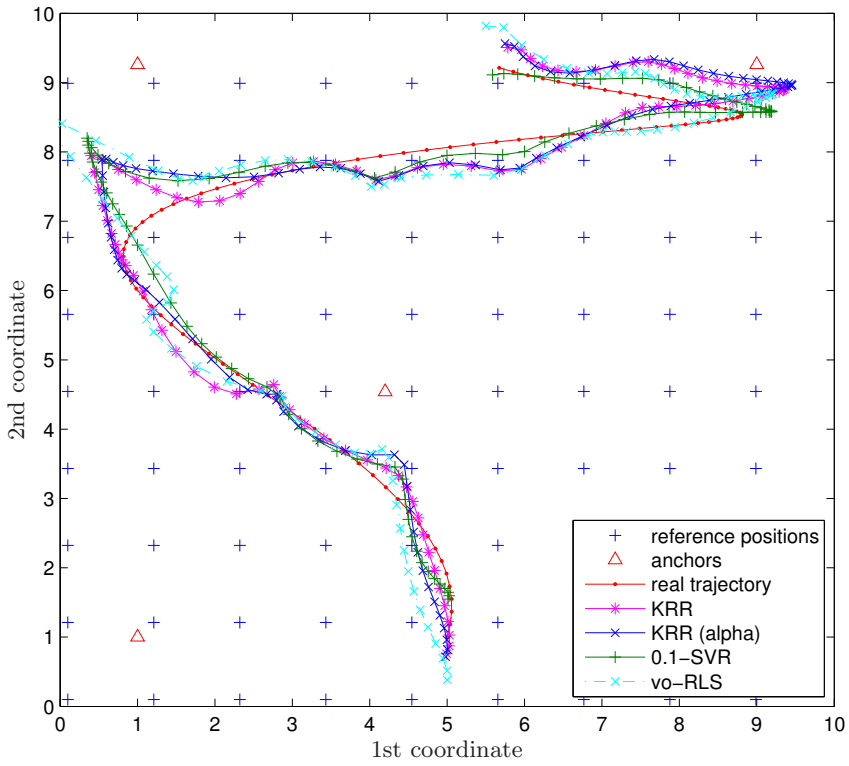


Figure 3.10: Estimation of the trajectory using real data

For the evaluation on simulated data, we first consider noiseless data and the same topology as in Figure 3.6. We generate the RSSI measures of the signals exchanged between the moving node and the anchors using (3.14), with the same parameters as in the beginning of Subsection 3.5.1. The RSSI measures of the signals exchanged between the moving node and the cluster heads, i.e., $\xi^{(z)}$, $z \in \{1, ..., Z\}$, are also generated using the defined RSSI model. The area is partitioned into four clusters (Z =4). The estimation error is equal to 0.14 m in the case of the clusterized approach, whereas with the centralized approach, it is equal to 0.17 m. We then add a zero-mean Gaussian white noise of standard deviation σ_{ρ} =1 dBm to the RSSI values. The estimation error is equal to 1.30 m for the clusterized approach, while the centralized version yields 1.23 m of errors. Both centralized and clusterized methods have almost the same efficiency when the noise is spread in the same way all over the network. In fact, in the absence of noise, the creation of several local maps, with a weighted fusion of the different estimations, yields better estimates compared to the centralized approach. However, in the case of noisy data, highly corrupted RSSIs will affect more than one cluster estimation (especially in the localization phase), leading to a spatial spreading of errors that will greatly affect the fusion, compared to a single estimation performed at one fusion center.

Not forgetting also that the weights used in the fusion (equation (3.12)) are based on power estimations that are also affected by noise. Besides its robustness, the clusterized approach outperforms the centralized one in terms of computation complexity. Indeed, while the clusterized scheme requires only $\mathcal{O}((N^{(z)})^3)$ for the matrix inversion, with $N^{(z)}$ =25 in this case, the centralized scheme considers all the reference positions at once with $\mathcal{O}({N_p}^3)$, where N_p =100.

Afterwards, we compare both centralized and clusterized schemes in the case of real collected data. We consider the same scenario as the one described in the Subsection 3.5.1. Figure 3.11 shows the considered topology, where the region of interest is partitioned into nine clusters, as well as the estimated trajectories using both approaches. The estimation error is equal to 0.28 m for the clusterized approach and to 0.34 m for the centralized approach. Since, in the practical setup, noise is not necessarily uniformly distributed among clusters (as in simulated data), dividing the region into clusters allows some of the local models to be less affected by errors than others. This form of diversity is absent in the centralized model where a single model is created from the noisy data to cover the whole region.

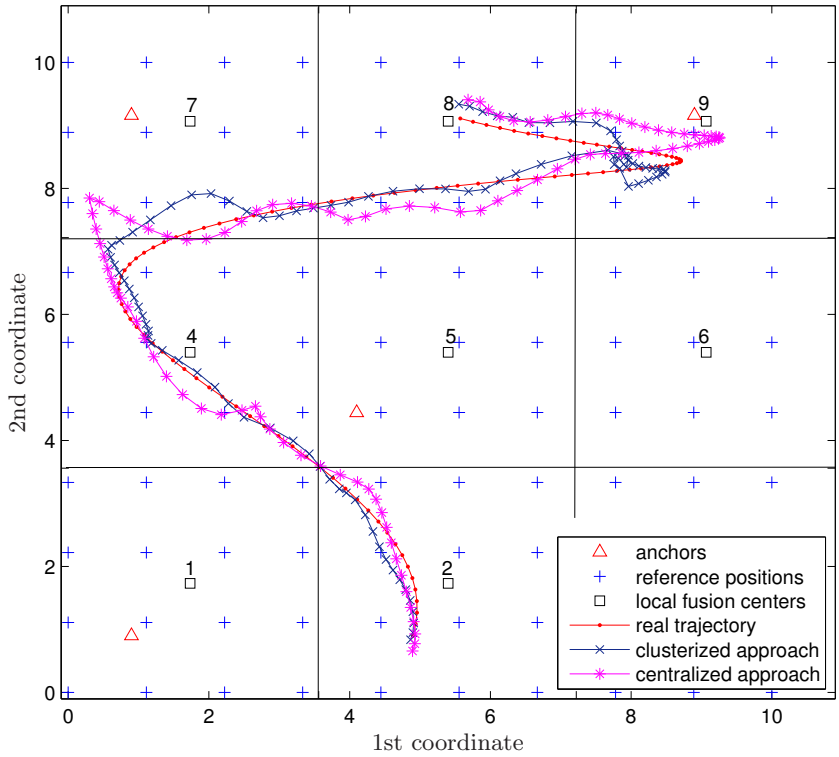


Figure 3.11: Estimation of the trajectory using real data with the clusterized approach

3.5.4 Comparison to the state-of-the-art

The objective of this subsection is to provide a comparison of the proposed centralized localization approach with respect to two well-known localization techniques: localization by connectivity and localization using the weighted K-nearest neighbor algorithm (WKNN) for different weighting methods. Comparisons are made with respect to the results obtained with the kernel ridge regression of Subsection 3.3.1, since it yields the best results, as shown previously.

Localization by connectivity [Y. Shang and Fromherz, 2003] is a simple technique that only uses neighboring anchors information and RSSI measures. All anchors are considered to have the same transmission range, which is an ideal disk with an equal transmission radius. A node's position is found by determining the intersection of all the range disks of the anchors detected by this node. However, this technique can only provide a coarse-grained estimate of each node's location. In addition, the localization error is highly dependent on the node density in the network, the number of anchors, and the network topology [Mao et al., 2007]. The results in Table 3.11 show the high influence of the number of anchors on the connectivity-based method, while our proposed method maintains a low level of estimation error in all cases. Moreover, our method outperforms the connectivity-based method in all situations, with a decrease of the estimation error by more than 75%. In terms of computation time, the advantage of the connectivity-based method is that it does not need training computations. However, in the case of 16 anchors, finding the position of a trajectory point with the kernel ridge regression takes about 0.3 milliseconds as we already stated in Table 3.6, whereas the connectivity algorithm takes around 3.8 milliseconds. Furthermore, when estimating the whole trajectory, that is 100 consecutive positions, the time needed for the connectivity is even higher than the total complexity of the training and localization phases together, using the kernel ridge regression. For all these reasons, the kernel ridge regression method seems to be the most convenient one to the considered context, as it globally presents the best performance with the lowest overall complexity. Moreover, when 196 anchors are deployed, the connectivity algorithm takes about 150 milliseconds to estimate a point of the trajectory, while the kernel ridge regression only needs 0.5 milliseconds. Therefore, we conclude that the kernel ridge regression method necessitates less computation time for the trajectory estimation, compared to connectivity, once the model has been defined.

Let us now consider the weighted K-nearest neighbor (WKNN) algorithm [Koyuncu and Yang, 2011]. This algorithm relies on the Euclidean distances between the RSSI values received by the node to be localized and the fingerprints in the database, to provide an estimation of the node's position. Let $\text{dist}_\ell = \|\boldsymbol{\rho} - \boldsymbol{\rho}_\ell\|$ be the Euclidean distance between

Table 3.11: Estimation errors of the proposed centralized method compared to the connectivity-based one, with noiseless data

No. of anchors (N_a)	16	64	81	100	196
Connectivity	7.44	3.20	3.02	2.61	1.92
Proposed method	0.17	0.51	0.61	0.65	0.22

the RSSI vector $\boldsymbol{\rho}$ of the mobile node and $\boldsymbol{\rho}_\ell$ of the database where $\ell \in \{1, \ldots, N_p\}$. Also, let $\mathcal{I}$ be the set of indices of $\boldsymbol{\rho}_\ell$ yielding the K smallest distances dist_ℓ (i.e., from the K nearest neighbors). Then, the node's position is estimated by:

$$\hat{\boldsymbol{x}} = \sum_{k \in \mathcal{I}} w_k \boldsymbol{p}_k,$$

where $\boldsymbol{p}_k$ is one of the nearest reference positions neighboring the node, and w_k is the corresponding normalized weighting factor. Note that the weighting factor highly influences the position accuracy; in fact, nearer neighbors should have higher weights, in order to contribute more to the position coordinates, compared to further neighbors (i.e., with lower weights). Therefore, weights should be chosen in a decreasing manner with respect to the distances. Weight values often considered in the literature are listed in Table 3.12, "A" to "E". Table 3.13 shows the estimation errors using WKNN for different weight functions and the proposed centralized approach using the kernel ridge regression, in the case of noiseless simulated data. On the other hand, in Table 3.14, the results shown are obtained in the case of RSSI measures corrupted by an additive zero-mean Gaussian white noise of standard deviation equal to 1 dBm. In Table 3.15, estimation errors are presented for the case of real data. Optimal values of K are found using the 10-fold cross validation, for $K \in \{1, 2, \ldots, 15\}$. Notice that in all the presented scenarios, the proposed localization approach outperforms the WKNN method, in terms of accuracy. However, estimating a trajectory point takes about 0.2 millisecond when using the WKNN; therefore, the WKNN algorithm slightly outperforms the kernel ridge regression in terms of time consumption, but with less estimation accuracy.

As for the localization method proposed in [Kushki et al., 2007], it also uses weighted combinations of a set of fingerprints from the training database. However, instead of using the K nearest neighbors, the authors present a new method for selecting this set of fingerprints. Indeed, they propose a spatially localized averaging strategy wherein anchor coverage information is used to retain a subset of spatially relevant reference positions. This is done based on the premise that reference positions close to the node, at a given step, are covered by a similar set of anchors. A real-time anchor selection algorithm is also proposed, where the anchors yielding the strongest RSSIs are first selected. Then, the correlation between RSSIs is considered in order to obtain a representation with minimal redundancy, reducing again the number of anchors. It is worth noting

that in [Kushki et al., 2007], many samples of RSSI are collected for each reference position. However, since in our simulations only one RSSI vector per reference position is considered, the number of samples becomes equal to 1. Then, the weight used in [Kushki et al., 2007] is given by "F" in Table 3.12, where S is the set of fingerprints obtained after the spatial filtering, τ is the number of selected anchors, and dist_k is the Euclidean distance between the RSSI vectors of the node and the considered reference position, while only taking into account the RSSI components corresponding to the selected anchors. The parameter ς is chosen according to the authors' definition. The anchor selection is performed on the five strongest anchors, then three of these anchors are chosen using the redundancy information. The smallest estimation errors obtained for the different proposed scenarios are given in Table 3.13, 3.14, and 3.15. One can see that our proposed method clearly outperforms the one in [Kushki et al., 2007]. As for the time complexity of the latter, we found that the elapsed time for the estimation of a trajectory point is about 1.9 milliseconds, which is much higher than the kernel ridge regression execution time.

Table 3.12: List of selected weight factors w_k

Type	Expression of w_k
"A"	$1/K$
"B"	$\frac{1/\text{dist}_k}{\sum_{v\in I} 1/\text{dist}_v}$
"C"	$\frac{1/\text{dist}_k^2}{\sum_{v\in I} 1/\text{dist}_v^2}$
"D"	$\frac{1/\text{dist}_k^3}{\sum_{v\in I} 1/\text{dist}_v^3}$
"E"	$\frac{\exp(-\text{dist}_k)}{\sum_{v\in I} \exp(-\text{dist}_v)}$
"F"	$\frac{\frac{1}{(\sqrt{2\pi}\varsigma)^\tau}\exp(-\frac{\text{dist}_k^2}{2\varsigma^2})}{\sum_{v\in S}\frac{1}{(\sqrt{2\pi}\varsigma)^\tau}\exp(-\frac{\text{dist}_v^2}{2\varsigma^2})}$

3.6 Conclusion

In this chapter, we proposed a new localization framework using radio-location fingerprinting and kernel-based machine learning. Different machine learning algorithms have been considered, namely the kernel ridge regression, the vector-output regularized least squares, and the support vector regression. We also proposed a clusterized version of this localization approach. The clusterized approach consists in dividing the network into several clusters, where different local position estimates are computed by the cluster heads, based on the centralized localization approach. Simulation results show that our approaches allow accurate localization in simulated and real data cases. Moreover, they outperform other localization techniques such as the WKNN algorithm, for different weight models, and the localization by connectivity, for different numbers of anchors.

Table 3.13: Estimation errors for different weight models, using noiseless simulated data

WKNN					"F"	Proposed method
"A" K=2	"B" K=8	"C" K=7	"D" K=9	"E" K=4		
4.36	2.74	2.13	2.34	3.61	4.66	0.17

Table 3.14: Estimation errors for different weight models, using noisy simulated data

WKNN					"F"	Proposed method
"A" K=3	"B" K=13	"C" K=7	"D" K=9	"E" K=4		
4.48	2.42	2.39	2.41	3.67	4.98	1.88

Table 3.15: Estimation errors for different weight models, using real data

WKNN					"F"	Proposed method
"A" K=2	"B" K=2	"C" K=11	"D" K=10	"E" K=7		
0.72	0.64	0.61	0.58	0.62	0.75	0.34

However, we will show in the next chapter that, when used alone, this RSSI-based framework is vulnerable to the presence of high additive noise. Therefore, in the succeeding chapter, we introduce further improvements of this method, by correcting the estimated trajectory using additional information, such as past known location information and instantaneous accelerations.

Chapter 4

Position-Based Tracking

Contents

Target tracking is an ongoing interesting research topic in wireless sensor networks. In this chapter, we introduce an original method for target tracking in WSNs. The proposed method combines machine learning with a Kalman filter to estimate instantaneous positions of a moving target. The target's accelerations, along with information from the network, are used to obtain an accurate estimation of its position. The performance of the method is studied for different scenarios and a thorough comparison to well-known algorithms is also provided.

4.1 Introduction

Target tracking [Li et al., 2002; Dallil et al., 2013] consists in estimating instantly the position of a moving target. As we already explained in Chapter 1, target tracking can be viewed as a sequential localization problem; therefore, it requires a real-time recursive location estimation algorithm. In Chapter 3, we performed localization using only RSSI information collected from the network. However, it is interesting to take advantage of additional information, when available, to correct the estimated trajectory. Such additional information could be inertial information, such as past location information and instantaneous accelerations of the target under investigation.

The foundation of tracking using inertial information is to recursively estimate the states of the target, i.e., the target's positions and the target's velocities [Svensson, 2010]. Using a model that describes the target's motion, one can first predict the future value of the state of the target, knowing the target's current state. Then, the predicted state is updated using observations from the network, i.e., any relevant measurement collected from the network. Several filters can be used to combine inertial information and observations, depending on the type of the observations. For instance, the Kalman filter (KF) [Kalman, 1960; Welch and Bishop, 2001] can be used in the case of a linear observation model. In the case of non-linearity, the extended Kalman filter (EKF) [Julier and Uhlmann, 1997] and unscented Kalman filter (UKF) [Julier and Uhlmann, 2004] could be used. However, as we stated in Chapter 1, such approaches perform linearization and approximations leading to sub-optimal performance and sometimes divergence of the filter [UmaMageswari et al., 2012]. The particle filter (PF) [Branko Ristic, 2004] is also used for tracking. Such a filter has more potential than the Kalman filter in the cases of non-Gaussian noises and non-linear models [Gustafsson et al., 2002]; however, algorithms employing this filter need more computations for the generation of samples and the resampling step than algorithms using the Kalman filter.

Many tracking techniques using inertial information have been proposed in the literature. In addition to RSSI measurements, these techniques employ a state-space model to refine the position estimation based on its previous position. For instance, a first-order model is used with a Kalman filter in [Li and Li, 2012], and with a particle filter in [Dias and Bruno, 2013], whereas [Jayamohan and Mathurakani, 2013] employs a second-order one. Another tracking technique is introduced in [Chan et al., 2009; Liu et al., 2011], where the authors estimate the positions of a target using the weighted K-nearest neighbor algorithm. Then, these WKNN estimates are combined to inertial information by means of a Kalman filter. However, all these models are only reliable for targets having slightly varying velocities or accelerations.

In this chapter, we adapt the localization technique introduced in Chapter 3 in order to perform tracking in WSNs. Compared to the previous chapter, we now take advantage of the target's mobility to enhance the obtained position estimate. In fact, the proposed tracking approach combines radio-fingerprinting and inertial information. It consists in setting reference positions along the network where RSSI measurements are collected, leading to a radio-fingerprint database. This database is used with machine learning algorithms to define a kernel-based model, whose input is the RSSI vector and whose output is the corresponding position. A moving target measures then its RSSIs and instantaneous accelerations. A first position estimate is obtained using the already-defined kernel-based model and the measured RSSIs, in a similar manner as in Chapter 3. Then, this estimate is combined with the acceleration information, by means of a Kalman filter, to achieve better accuracy. To this end, three different orders of the tracking models are examined. The proposed method outperforms existing methods, especially for hyperactive targets.

4.2 Problem Statement

In this chapter, we propose a centralized tracking approach, in which all collected data are transmitted to the central fusion station for processing. We use the centralized approach to simplify the explanation of the algorithm, since we showed in Section 3.5 of Chapter 3 that both centralized and clusterized approaches yield close results. Nevertheless, this tracking technique can be extended to a clusterized one based on the study of Section 3.4 of Chapter 3. In this section, we first give a description of the network's configuration. Then, in the second subsection, we describe three different mobility models, each based on different assumptions. Finally, we introduce the observation model, which is based on the localization framework described in Chapter 3.

4.2.1 Network configuration

We consider a WSN composed of N_s fixed sensors having known locations, denoted by $\boldsymbol{s}_i$, $i \in \{1, \ldots, N_s\}$. Let δ denote the dimension of the surveillance area ($\delta = 2$ or $\delta = 3$). For the sake of clarity, and without loss of generality, only one target with the unknown position $\boldsymbol{x}(k)$ is considered, k being the current time step. Nevertheless, the proposed method extends naturally to several moving targets, since they are tracked independently from each other, using their accelerations and information exchanged only with the fixed sensors. As we already explained in Chapter 1, targets can either be cooperative and participate intentionally in the tracking process, or non cooperative and do not exchange data for position estimation. The tracking approach presented in this chapter is used in the case of a cooperative target. Indeed, the target is assumed to continuously send to the fusion center its measured inertial information and the collected information from the network. Note that the target can perform auto-tracking if it has enough computational capacities. In the following, all coordinates are δ-dimension row vectors, and the variables that will be used in this chapter are listed in Table 4.1, along with their respective sizes.

Table 4.1: List of the used variables in Chapter 4, along with their respective sizes

Notation	Variable	Size
δ	environment dimension	1
N_s	number of fixed sensors	1
N_p	number of reference positions	1
k	time step	1
ϑ, φ, ϕ	rotation angle	1
$\boldsymbol{s}_i$	position of fixed sensor i	$1 \times \delta$
$\boldsymbol{x}$	target position	$1 \times \delta$
$\boldsymbol{\nu}$	target velocity	$1 \times \delta$
$\boldsymbol{\gamma}$	target acceleration	$1 \times \delta$
$\boldsymbol{p}_\ell$	reference position	$1 \times \delta$
$\boldsymbol{\psi}(\cdot)$	proposed position model	$1 \times \delta$
$\boldsymbol{z}$	position estimate	$1 \times \delta$
$\boldsymbol{\rho}$, $\boldsymbol{\rho}_\ell$	vector of RSSI measures	$N_s \times 1$
$\boldsymbol{n}$	observation noise	$\delta \times 1$
$\boldsymbol{X}$, $\widehat{\boldsymbol{X}}$, $\widehat{\boldsymbol{X}}^-$	target state	$2\delta \times 1$
$\boldsymbol{B}$, $\boldsymbol{B}_1$, $\boldsymbol{B}_2$	control-input vector	$2\delta \times 1$
$\boldsymbol{\epsilon}$	state noise	$2\delta \times 1$
$\boldsymbol{R}$	covariance matrix	$\delta \times \delta$
$\boldsymbol{C}$	observation matrix	$\delta \times 2\delta$
$\boldsymbol{Q}$, $\boldsymbol{T}$, $\boldsymbol{T}^-$, $\boldsymbol{V}$	covariance matrix	$2\delta \times 2\delta$
$\boldsymbol{A}$	state transition matrix	$2\delta \times 2\delta$

4.2.2 Target's motion

In order to perform tracking, it is interesting to take advantage of the target's motion, and thus its inertial information. Therefore, it is important to be able to describe the target's motion, i.e., to characterize the relationship between the target's previous state and its current state. This subsection describes three mobility models having different orders. The target is assumed to be equipped with an accelerometer, yielding at each time step its current δ accelerations. The target is assumed to be fixed at a known position $\boldsymbol{x}(0)$ at the beginning of the tracking. The objective consists then in relating the current position of the target $\boldsymbol{x}(k)$ to its previous position $\boldsymbol{x}(k-1)$, using its measured accelerations. To do this, four state-space models are described: (i) a first-order model, where the velocities are assumed to be constant during the tracking process, and thus the accelerations are assumed to be null; (ii) a hybrid first-order model where, consecutively, the accelerations then the velocities are assumed to be constant between any two consecutive time steps; (iii) a second-order one where the accelerations are assumed to be constant between any two consecutive time steps, with linearly varying velocities; and finally, (iv) a third-order one where the accelerations are assumed to vary linearly between any two consecutive time steps. In the following, $\boldsymbol{\nu}(k)$ denotes the estimated velocity vector of the target at the time step k, $\boldsymbol{\gamma}(k)$ denotes its measured acceleration vector at the time step k, and Δt is the tracking period, that is the time period separating two consecutive time steps $k-1$ and k. Note that an extension to the case of a rotating target will be provided in the last subsection.

4.2.2.1 First-order mobility model

The first-order mobility model assumes that the velocities are constant between any two consecutive times steps; moreover, it assumes that the velocities are constant during all the tracking process [Liu et al., 2011; Dias and Bruno, 2013], as follows:

$$\boldsymbol{\nu}(k) = \boldsymbol{\nu}(k-1),$$

where $\boldsymbol{\nu}(0)$ is considered to be known at the beginning of the tracking. The target's position is then given by:

$$\boldsymbol{x}(k) = \boldsymbol{x}(k-1) + \boldsymbol{\nu}(k)\,\Delta t. \tag{4.1}$$

One can see that this model assumes that the accelerations are null, which is not always the case. Consequently, it only works in limited conditions.

4.2.2.2 Hybrid first-order mobility model

We now propose to modify the first mobility model, in ordre to update the target's velocities by taking into account the measured accelerations. This mobility model makes two assumptions on the motion of the target. It first assumes that the acceleration vector of the target is constant between two consecutive time steps $k-1$ and k, and equal to $\boldsymbol{\gamma}(k)$. It thus computes the target's velocity vector iteratively by:

$$\boldsymbol{\nu}(k) = \boldsymbol{\nu}(k-1) + \boldsymbol{\gamma}(k)\,\Delta t, \tag{4.2}$$

where $\boldsymbol{\nu}(0)$ is null since the target is assumed to be fixed at the beginning of the tracking. Then, the velocities are assumed to be constant between $k-1$ and k and equal to $\boldsymbol{\nu}(k)$, leading to:

$$\boldsymbol{x}(k) = \boldsymbol{x}(k-1) + \boldsymbol{\nu}(k)\,\Delta t.$$

It is obvious that this model only works for slightly-varying-velocity targets (i.e., $\boldsymbol{\gamma} \simeq \mathbf{0}$). Because of its two assumptions, the first-order model degrades significantly when the target is more active.

4.2.2.3 Second-order mobility model

In the case of the second-order mobility model [Jayamohan and Mathurakani, 2013], the velocity vector is estimated as for the first-order model using equation (4.2). The second-order model assumes that the acceleration vector is constant between two consecutive time steps $k-1$ and k and equal to $\boldsymbol{\gamma}(k)$, as shown in blue in Figure 4.1. This assumption leads to the following:

$$\boldsymbol{x}(k) = \boldsymbol{x}(k-1) + \boldsymbol{\nu}(k-1)\,\Delta t + \boldsymbol{\gamma}(k)\,\frac{\Delta t^2}{2}. \tag{4.3}$$

This model outperforms the first-order one, since it considers less approximations and assumptions. It performs well with slightly varying accelerations motions. However, it is not well-adapted to trajectories with abrupt changes in accelerations, since estimates might significantly deviate from the exact trajectory due to cumulative model errors over time.

4.2.2.4 Third-order mobility model

Instead of assuming that the accelerations are constant between two consecutive time steps $k-1$ and k as in the case of the second-order mobility model, we now propose

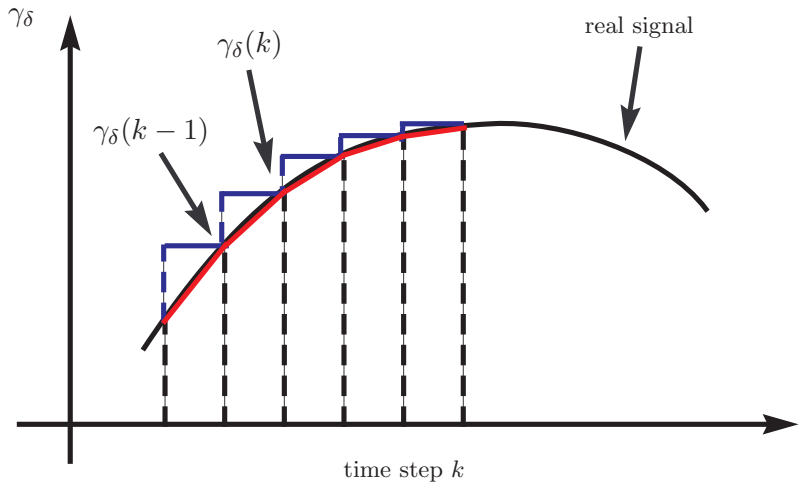


Figure 4.1: Estimated acceleration signal when using the second-order mobility model in blue and the third-order mobility model in red

to consider that the target's accelerations vary linearly between two consecutive time steps. Then, the acceleration vector varies from $\boldsymbol{\gamma}(k-1)$ at $k-1$ to $\boldsymbol{\gamma}(k)$ at k with a slope equal to $\frac{\boldsymbol{\gamma}(k)-\boldsymbol{\gamma}(k-1)}{\Delta t}$, as shown in red in Figure 4.1. According to this assumption, the velocity vector of the target at time step k is estimated recursively by:

$$\begin{aligned}\boldsymbol{\nu}(k) &= \boldsymbol{\nu}(k-1) + \boldsymbol{\gamma}(k-1)\,\Delta t + \frac{\boldsymbol{\gamma}(k)-\boldsymbol{\gamma}(k-1)}{\Delta t}\,\frac{\Delta t^2}{2} \\ &= \boldsymbol{\nu}(k-1) + \frac{\Delta t}{2}\boldsymbol{\gamma}(k-1) + \frac{\Delta t}{2}\boldsymbol{\gamma}(k),\end{aligned} \tag{4.4}$$

where the target is also assumed to be fixed at the beginning of the tracking (i.e., $\boldsymbol{\nu}(0) = \mathbf{0}$) with null acceleration (i.e., $\boldsymbol{\gamma}(0) = \mathbf{0}$), and at a known position $\boldsymbol{x}(0)$. Then, the target's position at time step k is given by:

$$\begin{aligned}\boldsymbol{x}(k) &= \boldsymbol{x}(k-1) + \boldsymbol{\nu}(k-1)\,\Delta t + \boldsymbol{\gamma}(k-1)\,\frac{\Delta t^2}{2} + \frac{\boldsymbol{\gamma}(k)-\boldsymbol{\gamma}(k-1)}{\Delta t}\,\frac{\Delta t^3}{6} \\ &= \boldsymbol{x}(k-1) + \boldsymbol{\nu}(k-1)\,\Delta t + \frac{\Delta t^2}{3}\boldsymbol{\gamma}(k-1) + \frac{\Delta t^2}{6}\boldsymbol{\gamma}(k),\end{aligned} \tag{4.5}$$

This model outperforms the other models, since it brings the estimated trajectory closer to the real one compared to the others, especially for hyperactive targets having highly varying accelerations. Indeed, Figure 4.1 shows the improvement in the accuracy of the estimated acceleration signals when using the third-order mobility model instead of the second-order one.

4.2.2.5 Extension to rotation

In the aforementioned mobility models, direct acceleration measurements made at the target are used. This means that the accelerations measured by the accelerometer in the target's coordinate system are used directly in the equations, as if they are measured in the global coordinate system. By doing so, the target is assumed to be rotationally constrained. However, the target could rotate during its motion in real applications, and its coordinate system, where the accelerations are given, might change. To overcome this problem, each target should be equipped with a gyroscope, that yields its orientations with respect to the global coordinate system.

Without loss of generality, consider that the localization is performed in a three-dimensional environment, i.e., $\delta = 3$. Consider that ϑ, φ, and ϕ are the angles of the counter-clockwise rotation of the target, given by its gyroscope at a given time step around the third coordinate axis of the global system, the first one and the second one respectively. The plots (a), (b) and (c) of Figure 4.2 illustrate the single rotations around the third, the first, and the second axes respectively, 1,2, and 3 being the global coordinate axes, and $1'$,$2'$, and $3'$ the target's ones. Let $\boldsymbol{\gamma} = (\gamma_1 \quad \gamma_2 \quad \gamma_3)$ be the acceleration vector of the target in the global coordinate system at a given time step and let $\boldsymbol{\gamma}' = (\gamma_{1'} \quad \gamma_{2'} \quad \gamma_{3'})$ be its measured one in its coordinate system. Then, having the rotation angles, $\boldsymbol{\gamma}$ is computed as follows:

$$\boldsymbol{\gamma} = \boldsymbol{\gamma}' \, \mathfrak{R},$$

where the first column of the three-dimensional rotation matrix $\mathfrak{R}$ is defined by

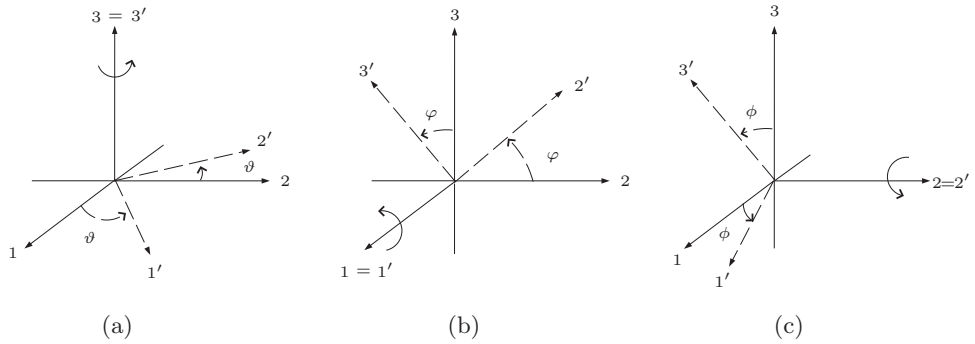
$$\begin{pmatrix} \cos\vartheta \cos\phi \\ -\sin\vartheta \cos\phi \\ \sin\phi \end{pmatrix},$$

its second column is defined by

$$\begin{pmatrix} \cos\vartheta \sin\varphi \sin\phi + \sin\vartheta \cos\varphi \\ -\sin\vartheta \sin\varphi \sin\phi + \cos\vartheta \cos\varphi \\ -\sin\varphi \cos\phi \end{pmatrix},$$

and its third column is defined by

$$\begin{pmatrix} -\cos\vartheta \cos\varphi \sin\phi + \sin\vartheta \sin\varphi \\ \sin\vartheta \cos\varphi \sin\phi + \cos\vartheta \sin\varphi \\ \cos\varphi \cos\phi \end{pmatrix}.$$



(a) (b) (c)

Figure 4.2: Rotations in a three-dimensional environment

In a two-dimensional environment (i.e., $\delta = 2$), where $\boldsymbol{\gamma} = (\gamma_1 \quad \gamma_2)$, the rotation is only possible in the plane with the rotation angle ϑ. By setting $\phi = \varphi = 0$, one gets the following transformation:

$$\boldsymbol{\gamma} = \boldsymbol{\gamma}' \begin{pmatrix} \cos\vartheta & \sin\vartheta \\ -\sin\vartheta & \cos\vartheta \end{pmatrix}. \tag{4.6}$$

During the tracking, the target measures its acceleration vector in its coordinate system at each time step, then finds its orientations using the gyroscope. Its accelerations in the global coordinate system can then be computed and used with one of the already stated mobility models in the localization algorithm.

4.2.3 The observation model

Now that the mobility models of the target are described, the idea of this section is to define its observation model to help improving its position estimate. Indeed, we can estimate the target's position from its previous position, using either of the four above described mobility models with the target's measured inertial information. However, measured inertial information, i.e., measured accelerations or velocities, are often noisy, leading to estimated positions that deviate from the real positions when this kind of measurements is taken alone. Therefore, it is important to combine the obtained estimation to additional information collected from the network.

The aim of this section is thus to set an observation model that could be used in the tracking process, based on the information gathered by the target from the fixed sensors in the network. In Chapter 1 and Chapter 3, we discussed the advantages of using the powers of the signals exchanged between sensors in a wireless sensor network. Indeed, RSSI-based techniques rely on the physical property of the signal, that states that its strength is attenuated with the traveled distance. Such techniques achieve acceptable performance, without the need for additional hardware. However, RSSI measures could

be significantly affected by several interferences and variables. For this reason, we proposed in the previous chapter a localization framework that employs the fingerprinting technique, that allows us to take into account many characteristics of the environment. Here as well, instead of using the exact values of RSSI as observations, we propose to exploit the fingerprinting technique in order to define our observation model.

In the following, let $\boldsymbol{p}_\ell$, $\ell \in \{1, \ldots, N_p\}$, denote the reference positions, with N_p being their number. These reference positions are generated over a fixed grid or randomly in the studied region. All fixed sensors continuously broadcast signals in the network at a fixed initial power, and a sensor is placed consecutively at the reference positions to detect the broadcasted signals and measure their RSSIs. Let $\boldsymbol{\rho}_\ell = (\rho_{\boldsymbol{s}_1,\boldsymbol{p}_\ell} \quad \ldots \quad \rho_{\boldsymbol{s}_{N_s},\boldsymbol{p}_\ell})^\top$ be the vector of RSSIs sent by all N_s sensors and received at the position $\boldsymbol{p}_\ell$, $\ell \in \{1, \ldots, N_p\}$. In this way, a radio-fingerprint database of N_p pairs $(\boldsymbol{\rho}_\ell, \boldsymbol{p}_\ell)$ is obtained. Then the objective is to find a relationship between the target's positions and the collected measurements.

While moving, the target collects the sensor signals and measures their RSSIs. Then, as explained in Chapter 3, we propose to use a model $\boldsymbol{\psi}\colon \mathbb{R}^{N_s} \to \mathbb{R}^{\delta}$ to estimate the target's position. Based on the constructed radio-fingerprint database, this model is defined in a way to associate to each RSSI vector $\boldsymbol{\rho}_\ell$ the corresponding position $\boldsymbol{p}_\ell$, with the advantage of not having to estimate the channel model. Several kernel-based machine learning algorithms are presented in Chapter 3 for the definition of this model, such as the kernel ridge regression, the support vector regression and the vector-output regularized least squares. For more details, refer to Section 3.3 on page 42. Once the model is defined, the target stores, at a given time step k, the measured RSSIs into a vector $\boldsymbol{\rho}(k)$, and then communicates this vector to the fusion center. Using the defined model $\boldsymbol{\psi}(\cdot)$, the estimated target's position at time step k is given by $\boldsymbol{\psi}(\boldsymbol{\rho}(k))$, namely

$$\boldsymbol{z}(k) = \boldsymbol{\psi}(\boldsymbol{\rho}(k)). \tag{4.7}$$

This estimated quantity $\boldsymbol{z}(k)$ is then taken as the observation value at each time step. By using this assumption, it is obvious that the observation model would be linear, which simplifies the computations later.

Once the mobility and observation models are defined, one can combine information from both models using an adequate filter to provide more accurate estimates. In the following section, we solve the tracking problem using the Kalman filter, that performs a recursive data processing to give an efficient position estimate by incorporating both types of information.

4.3 Resolution using the Kalman Filter

In this section, the combination of the inertial information and the observations is performed using the Kalman filter [Kalman, 1960; Welch and Bishop, 2001]. The Kalman filter is named after Rudolph E. Kalman, who published in 1960 his famous work about the linear filtering problems. Also known as linear quadratic estimation, the Kalman filter is a mathematical tool having a great importance in solving real world sensing problems. While this filter has been around for more than fifty years, it has recently been applied in a wide variety of wireless sensors applications [Li and Li, 2012]. Indeed, it is one of the best estimators for such applications in the case of linear problems, due to its effectiveness and the simplicity of its implementation. The Kalman filter is a kind of predictor-corrector estimator that minimizes the estimated error covariance. By applying it to the target tracking problem, the Kalman filter first predicts the unknown state of the target using the previous estimated state and its mobility equation. Then, in the following step, the predicted state is corrected using observations from the network. Let $\boldsymbol{X}(k)$ denote the unknown state of the target at time step k. Since both the target's position $\boldsymbol{x}(k)$ and velocity $\boldsymbol{\nu}(k)$ are estimated in the four mobility equations described above, the unknown state $\boldsymbol{X}(k)$ is then a $2\delta \times 1$ vector including both vectors $\boldsymbol{x}(k)$ and $\boldsymbol{\nu}(k)$, such that $\boldsymbol{X}(k) = (\boldsymbol{x}(k)\ \boldsymbol{\nu}(k))^\top$. The state-space equation for Kalman is then based on one of the mobility models and is given by the following:

$$\boldsymbol{X}(k) = \boldsymbol{A}\ \boldsymbol{X}(k-1) + \boldsymbol{B}(k) + \boldsymbol{\epsilon}(k), \tag{4.8}$$

where $\boldsymbol{\epsilon}(k) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{V})$ is the state equation error whose probability distribution is assumed to be normal, having zero mean and covariance matrix $\boldsymbol{V}$ of size $2\delta \times 2\delta$. The matrix $\boldsymbol{A}$ is the state transition matrix that relates the current state of the target to its previous one, and $\boldsymbol{B}(k)$ is a noisy control-input vector depending on the accelerations. Based on the estimation obtained using the measured RSSIs and the kernel-based model $\boldsymbol{\psi}(\cdot)$, we can define afterwards the observation equation as follows:

$$\boldsymbol{z}(k)^\top = \boldsymbol{C}\ \boldsymbol{X}(k) + \boldsymbol{n}(k), \tag{4.9}$$

where $\boldsymbol{C}$ is the observation matrix that relates the state $\boldsymbol{X}(k)$ to the already estimated position $\boldsymbol{z}(k)$, and $\boldsymbol{n}(k) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R})$ is the observation noise with normal distribution, zero mean and covariance matrix $\boldsymbol{R}$ of size $\delta \times \delta$.

4.3.1 Definition of the filter's parameters

In this paragraph, we define the parameters of equations (4.8) and (4.9). The state-space equation (4.8) is defined with respect to the mobility model. Now if we consider the mobility model of order one given in (4.1), we can write the Kalman state-space model as follows:

$$\begin{aligned}\boldsymbol{X}(k) &= \begin{pmatrix} \boldsymbol{x}(k)^\top \\ \boldsymbol{\nu}(k)^\top \end{pmatrix} = \boldsymbol{A} \begin{pmatrix} \boldsymbol{x}(k-1)^\top \\ \boldsymbol{\nu}(k-1)^\top \end{pmatrix} + \boldsymbol{B}(k) + \boldsymbol{\epsilon}(k) \\ &= \begin{pmatrix} \mathbf{I}_\delta & \Delta t \mathbf{I}_\delta \\ \mathbf{0}_{\delta\times\delta} & \mathbf{I}_\delta \end{pmatrix} \boldsymbol{X}(k-1) + \boldsymbol{\epsilon}(k), \end{aligned} \tag{4.10}$$

where $\mathbf{I}_\delta$ is the $\delta \times \delta$ identity matrix, and $\mathbf{0}_{\delta\times\delta}$ is the $\delta \times \delta$ matrix of zeros. Then using the hybrid first-order mobility model, the state-space model is written as follows:

$$\boldsymbol{X}(k) = \begin{pmatrix} \mathbf{I}_\delta & \Delta t \mathbf{I}_\delta \\ \mathbf{0}_{\delta\times\delta} & \mathbf{I}_\delta \end{pmatrix} \boldsymbol{X}(k-1) + \begin{pmatrix} \boldsymbol{\gamma}(k)^\top \Delta t^2 \\ \boldsymbol{\gamma}(k)^\top \Delta t \end{pmatrix} + \boldsymbol{\epsilon}(k). \tag{4.11}$$

By considering the second-order mobility equation given in (4.3), we can write the second-order state-space model as follows:

$$\boldsymbol{X}(k) = \begin{pmatrix} \mathbf{I}_\delta & \Delta t \mathbf{I}_\delta \\ \mathbf{0}_{\delta\times\delta} & \mathbf{I}_\delta \end{pmatrix} \boldsymbol{X}(k-1) + \begin{pmatrix} \boldsymbol{\gamma}(k)^\top \frac{\Delta t^2}{2} \\ \boldsymbol{\gamma}(k)^\top \Delta t \end{pmatrix} + \boldsymbol{\epsilon}(k). \tag{4.12}$$

Finally, the third-order state-space model is given by the following:

$$\begin{aligned}\boldsymbol{X}(k) &= \boldsymbol{A}\ \boldsymbol{X}(k-1) + \boldsymbol{B}(k) + \boldsymbol{\epsilon}(k) = \boldsymbol{A}\ \boldsymbol{X}(k-1) + \boldsymbol{B}_1(k-1) + \boldsymbol{B}_2(k) + \boldsymbol{\epsilon}(k) \\ &= \begin{pmatrix} \mathbf{I}_\delta & \Delta t \mathbf{I}_\delta \\ \mathbf{0}_{\delta\times\delta} & \mathbf{I}_\delta \end{pmatrix} \boldsymbol{X}(k-1) + \begin{pmatrix} \boldsymbol{\gamma}(k-1)^\top \frac{\Delta t^2}{3} \\ \boldsymbol{\gamma}(k-1)^\top \frac{\Delta t}{2} \end{pmatrix} + \begin{pmatrix} \boldsymbol{\gamma}(k)^\top \frac{\Delta t^2}{6} \\ \boldsymbol{\gamma}(k)^\top \frac{\Delta t}{2} \end{pmatrix} + \boldsymbol{\epsilon}(k). \end{aligned} \tag{4.13}$$

Note that the covariance matrix $\boldsymbol{V}$ of the noise term $\boldsymbol{\epsilon}(k)$ depends of the error on the trajectory. It can be estimated in an empirical manner, by simulating several trajectories with several acceleration signals. The estimation of these trajectories using the four state-space models allow us to obtain an estimation of the covariance matrix $\boldsymbol{V}$ relative to each one of the models. Next, we define the parameters of (4.9). Since the observations are the position estimates given by the model $\boldsymbol{\psi}(\cdot)$, the $\delta \times 2\delta$ matrix $\boldsymbol{C}$ is then given by the identity matrix $\mathbf{I}_\delta$ completed by zeros, that is $\boldsymbol{C} = (\mathbf{I}_\delta \ \ \mathbf{0}_{\delta\times\delta})$. The covariance matrix $\boldsymbol{R}$ of the noise term $\boldsymbol{n}(k)$ is also approximated in an empirical manner. We first generate a new set of reference pairs of RSSI values and their corresponding positions. These positions are then estimated using the defined model $\boldsymbol{\psi}(\cdot)$ and the estimation error

is computed stored into a vector. Finally, the matrix $\boldsymbol{R}$ is determined by computing the covariance of this error vector. This matrix is considered to be constant over time and for all targets.

4.3.2 Algorithm using the Kalman filter

The solution of the tracking problem using the Kalman filter consists of two phases. The filter first predicts the unknown state using the previous estimated state and the state-space equation (4.8). Then, the predicted state is corrected using the observation model given in (4.9). Now let $\widehat{\boldsymbol{X}}(k-1)$ denote the target's state estimated with the Kalman filter at time step $k-1$. Using the state-space equation, the predicted state can be written as:

$$\widehat{\boldsymbol{X}}^{-}(k) = \boldsymbol{A}\,\widehat{\boldsymbol{X}}(k-1) + \boldsymbol{B}(k),$$

where $\boldsymbol{B}(k)$ is computed at each iteration as shown previously, and $\widehat{\boldsymbol{X}}(0)$ is assumed to be known. Then, the predicted estimation covariance reflecting the accuracy of the state estimate is updated as follows:

$$\boldsymbol{T}^{-}(k) = \boldsymbol{A}\,\boldsymbol{T}(k-1)\boldsymbol{A}^{\top} + \boldsymbol{Q}(k),$$

where $\boldsymbol{T}(k-1)$ is the final covariance estimation at time step $k-1$ and $\boldsymbol{T}(0)$ is null since the initial state is known. The matrix $\boldsymbol{Q}(k)$ is the covariance matrix of $\boldsymbol{X}(k)$ given $\boldsymbol{X}(k-1)$, namely

$$\begin{aligned}\boldsymbol{Q}(k) &= \mathrm{Cov}(\boldsymbol{X}(k)|\boldsymbol{X}(k-1) = \mathrm{Cov}(\boldsymbol{B}(k) + \boldsymbol{\epsilon}(k)) \\ &= \mathrm{Cov}(\boldsymbol{B}(k)) + \mathrm{Cov}(\boldsymbol{\epsilon}(k)) \\ &= \mathrm{Cov}(\boldsymbol{B}(k)) + \boldsymbol{V},\end{aligned}$$

where $\mathrm{Cov}(\cdot)$ computes the covariance matrix of its argument. Each coordinate acceleration noise is assumed to be independent with zero-mean normal distribution, having known variances $\sigma^2_{\gamma,d}$, $d = 1, ..., \delta$. Their values can be estimated by performing a calibration of the accelerometer before the tracking stage. We also assume that all the δ coordinates are statistically independent. In the case of the first-order state-space model, the covariance matrix $\boldsymbol{Q}(k)$ is given by $\boldsymbol{V}$ since $\boldsymbol{B}(k)$ is null. As for the case of

the hybrid first-order state-space model, its covariance matrix $\boldsymbol{Q}(k)$ is given by:

$$\begin{aligned}\boldsymbol{Q}(k) &= \boldsymbol{V} + \mathrm{Cov}\begin{pmatrix} \boldsymbol{\gamma}(k)^\top \Delta t^2 \\ \boldsymbol{\gamma}(k)^\top \Delta t \end{pmatrix} \\ &= \boldsymbol{V} + \begin{pmatrix} \Delta t^4\ \mathrm{Diag}\left(\boldsymbol{\sigma}_{\boldsymbol{\gamma}}^2\right) & \Delta t^3\ \mathrm{Diag}\left(\boldsymbol{\sigma}_{\boldsymbol{\gamma}}^2\right) \\ \Delta t^3\ \mathrm{Diag}\left(\boldsymbol{\sigma}_{\boldsymbol{\gamma}}^2\right) & \Delta t^2\ \mathrm{Diag}\left(\boldsymbol{\sigma}_{\boldsymbol{\gamma}}^2\right) \end{pmatrix},\end{aligned}$$

where $\mathrm{Diag}\left(\boldsymbol{\sigma}_{\boldsymbol{\gamma}}^2\right)$ is the $\delta \times \delta$ diagonal matrix with entries $\sigma_{\gamma,d}^2,\ d = 1, ..., \delta$. Then, for the second-order state-space model, the matrix $\boldsymbol{Q}(k)$ is given by the following:

$$\begin{aligned}\boldsymbol{Q}(k) &= \boldsymbol{V} + \mathrm{Cov}\begin{pmatrix} \boldsymbol{\gamma}(k)^\top \frac{\Delta t^2}{2} \\ \boldsymbol{\gamma}(k)^\top \Delta t \end{pmatrix} \\ &= \boldsymbol{V} + \begin{pmatrix} \frac{\Delta t^4}{4}\ \mathrm{Diag}\left(\boldsymbol{\sigma}_{\boldsymbol{\gamma}}^2\right) & \frac{\Delta t^3}{2}\ \mathrm{Diag}\left(\boldsymbol{\sigma}_{\boldsymbol{\gamma}}^2\right) \\ \frac{\Delta t^3}{2}\ \mathrm{Diag}\left(\boldsymbol{\sigma}_{\boldsymbol{\gamma}}^2\right) & \Delta t^2\ \mathrm{Diag}\left(\boldsymbol{\sigma}_{\boldsymbol{\gamma}}^2\right) \end{pmatrix}.\end{aligned}$$

Finally, for the third-order state-space model, the matrix $\boldsymbol{Q}(k)$ is given by:

$$\begin{aligned}\boldsymbol{Q}(k) &= \boldsymbol{V} + \mathrm{Cov}\left(\begin{pmatrix} \boldsymbol{\gamma}(k-1)^\top \frac{\Delta t^2}{3} \\ \boldsymbol{\gamma}(k-1)^\top \frac{\Delta t}{2} \end{pmatrix} + \begin{pmatrix} \boldsymbol{\gamma}(k)^\top \frac{\Delta t^2}{6} \\ \boldsymbol{\gamma}(k)^\top \frac{\Delta t}{2} \end{pmatrix}\right) \\ &= \boldsymbol{V} + \begin{pmatrix} \frac{5\Delta t^4}{36}\ \mathrm{Diag}\left(\boldsymbol{\sigma}_{\boldsymbol{\gamma}}^2\right) & \frac{\Delta t^3}{4}\ \mathrm{Diag}\left(\boldsymbol{\sigma}_{\boldsymbol{\gamma}}^2\right) \\ \frac{\Delta t^3}{4}\ \mathrm{Diag}\left(\boldsymbol{\sigma}_{\boldsymbol{\gamma}}^2\right) & \frac{\Delta t^2}{2}\ \mathrm{Diag}\left(\boldsymbol{\sigma}_{\boldsymbol{\gamma}}^2\right) \end{pmatrix}.\end{aligned}$$

Note that this computation is valid when the target is rotationally constrained or when its orientation angles are noiseless. However, in the other cases, the difference in the covariance computation could be balanced by the use of the modeling error with its covariance. Then, the predicted quantities $\widehat{\boldsymbol{X}}^-(k)$ and $\boldsymbol{T}^-(k)$ are corrected using the observation equation (4.9) as follows:

$$\widehat{\boldsymbol{X}}(k) = \widehat{\boldsymbol{X}}^-(k) + \boldsymbol{G}_{\mathrm{KF}}(k)\left(\boldsymbol{z}(k)^\top - \boldsymbol{C}\widehat{\boldsymbol{X}}^-(k)\right) \tag{4.14}$$

$$\boldsymbol{T}(k) = \left(\mathbf{I}_{2\delta} - \boldsymbol{G}_{\mathrm{KF}}(k)\,\boldsymbol{C}\right)\boldsymbol{T}^-(k), \tag{4.15}$$

where $\boldsymbol{G}_{\mathrm{KF}}(k)$ is the optimal Kalman gain given by:

$$\boldsymbol{G}_{\mathrm{KF}}(k) = \boldsymbol{T}^-(k)\,\boldsymbol{C}^\top\left(\boldsymbol{C}\,\boldsymbol{T}^-(k)\,\boldsymbol{C}^\top + \boldsymbol{R}\right)^{-1}. \tag{4.16}$$

4.4 Practical Simulations and Results

In this section, we evaluate the performance of the proposed tracking method on simulated data. In the first subsection, several trajectories with different orders for the

state-space model are examined. In the second subsection, we study the influence of the noise standard deviations $\boldsymbol{\sigma_\gamma}$ and $\sigma_{\boldsymbol{\rho}}$ on the estimation error. In the third subsection, we study the influence of the number of fixed sensors and the number of reference positions on the estimation error. The target being rotationally-constrained in the first three subsections, we highlight in the fourth subsection, the importance of taking into account its rotations. Finally, results are compared to the ones obtained with the WKNN algorithm combined with a Kalman filter [Chan et al., 2009; Liu et al., 2011] and with a tracking algorithm using particle filtering [Dias and Bruno, 2012].

For the two following subsections, we consider the same practical setup that consists of a 100 m $\times$ 100 m area, 16 fixed sensors and 100 reference positions distributed over a fixed grid. We generate the RSSI values in a similar manner as in Section 3.5 of Chapter 3, that is using the path-loss propagation model [Medeisis and Kajackas, 2000] given by:

$$\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell} = \rho_0 - 10\, n_P \, \log_{10} \|\boldsymbol{s}_i - \boldsymbol{p}_\ell\| + \varepsilon_{i,\ell}, \tag{4.17}$$

where $\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}$ (in dBm) is the power received from the sensor at position $\boldsymbol{s}_i$ by the node at position $\boldsymbol{p}_\ell$, ρ_0 is the power at a distance of 1 m set to 1 dBm, n_P is the path-loss exponent set to 4 here as well, $\|\boldsymbol{s}_i - \boldsymbol{p}_\ell\|$ is the Euclidian distance between $\boldsymbol{p}_\ell$ and $\boldsymbol{s}_i$, and $\varepsilon_{i,\ell}$ is the zero-mean noise affecting the RSSI measures, with $\sigma_{\boldsymbol{\rho}}$ its standard deviation. This model is also used to calculate the RSSI values collected by the target during its movement.

As we already explained, the estimated position of the target using the kernel-based model $\boldsymbol{\psi}(\cdot)$ will be taken as the observation. To define the model $\boldsymbol{\psi}(\cdot)$, one can use any of the machine learning algorithms described in Section 3.3 of Chapter 3. However, in the comparison between the learning algorithms provided in Subsection 3.5.1 of Chapter 3, we showed that the best estimations are obtained when using the kernel ridge regression and the vo-RLS. We also concluded that the ϵ-SVR has the worst performance when it comes to time complexity and estimation error. Consequently, we will only consider in the simulations of this chapter the kernel ridge regression and the vo-RLS. As for the choice of the kernel, we also consider the Gaussian kernel given in (3.13) on page 53. The 10-fold cross-validation technique [Stone, 1974] is used here as well to determine the value of the kernel's bandwidth σ and the value of the regularization parameter η.

4.4.1 Evaluation of the tracking method on three trajectories

In order to evaluate the accuracy of the proposed tracking scheme, we first consider three different trajectories of 100 points with $\Delta t = 1$s. For the trajectory illustrated in Figure 4.3, the accelerations are assumed to be equal to zero, leading to constant

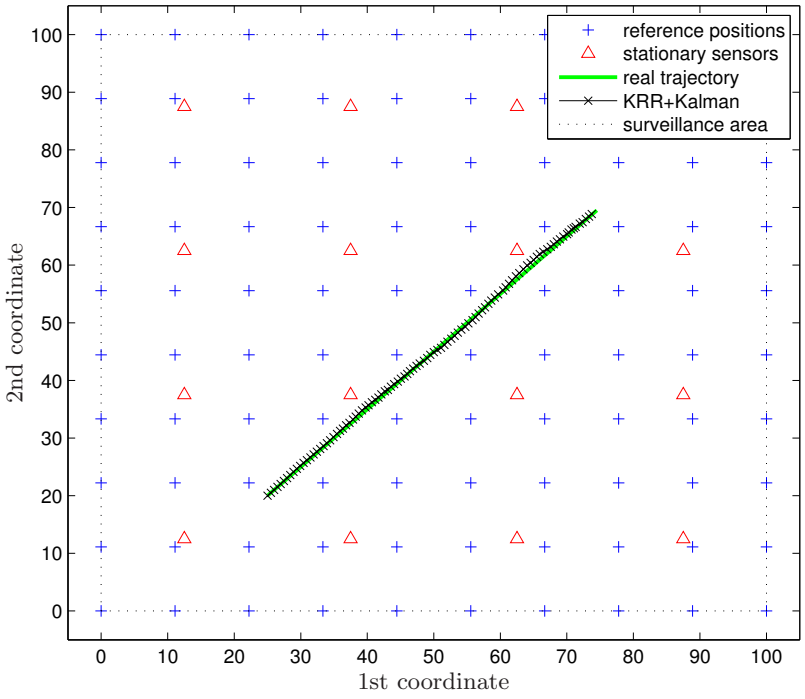


Figure 4.3: Estimation of the first trajectory

velocities. As for the second and the third trajectories of Figure 4.4 and Figure 4.5, their respective accelerations are given in the top plots and in the bottom plots of Figure 4.6, γ_1 and γ_2 being the first and the second acceleration coordinates respectively. One can see that the accelerations of the third trajectory have more variations than the accelerations of the second trajectory. The coordinate expressions are obtained by taking twice the primitive integral of the accelerations. By taking these three trajectories, the performance of the proposed method is evaluated for different types of scenarios, considering first a monotonously moving target, then more hyperactive ones.

In order to simulate a noisy environment, both components of $\boldsymbol{\sigma_\gamma}$ are taken equal to 0.01 m/s^2, and $\sigma_{\boldsymbol{\rho}}$ is taken equal to 1 dBm. The covariance matrix $\boldsymbol{V}$ of the state equation noise is set to zero for the hybrid first-order, the second-order and the third-order state-space models. It is taken equal to the covariance of the hybrid first-order model in the case of the first-order one. Then, let the estimation error be evaluated by the root mean squared distance between the exact positions and the estimated ones. Figures 4.3, 4.4, and 4.5 show the estimated trajectories when using the proposed method with the kernel ridge regression (KRR) and the third-order state-space equation (4.13). Table 4.2 shows the average over 50 simulations of the estimation errors in meters for the three trajectories and the three different state-space models, using the kernel ridge regression

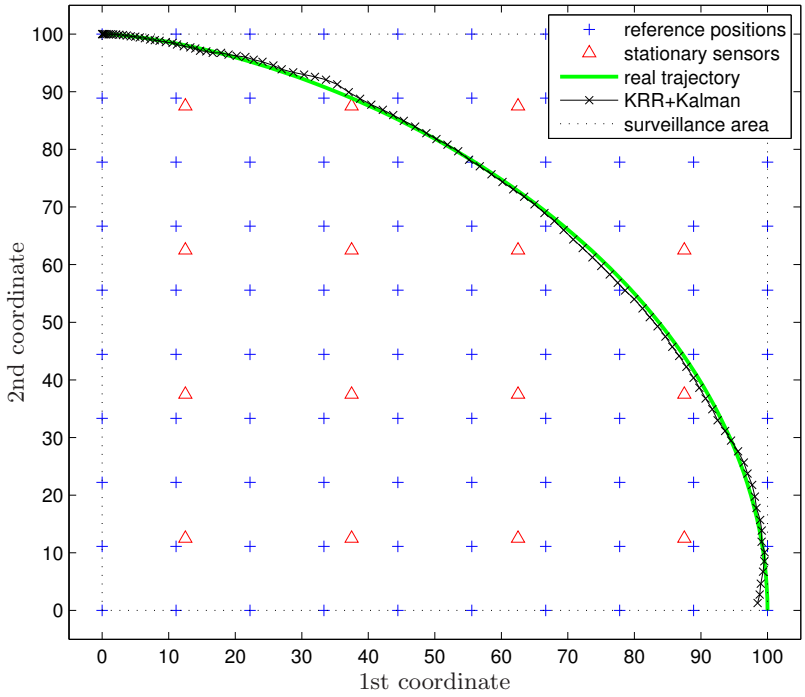


Figure 4.4: Estimation of the second trajectory

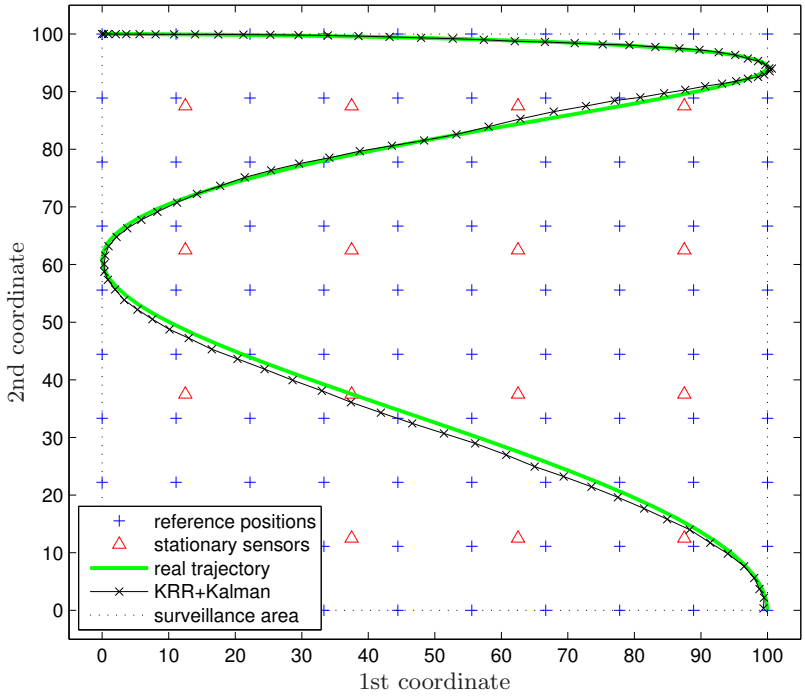


Figure 4.5: Estimation of the third trajectory

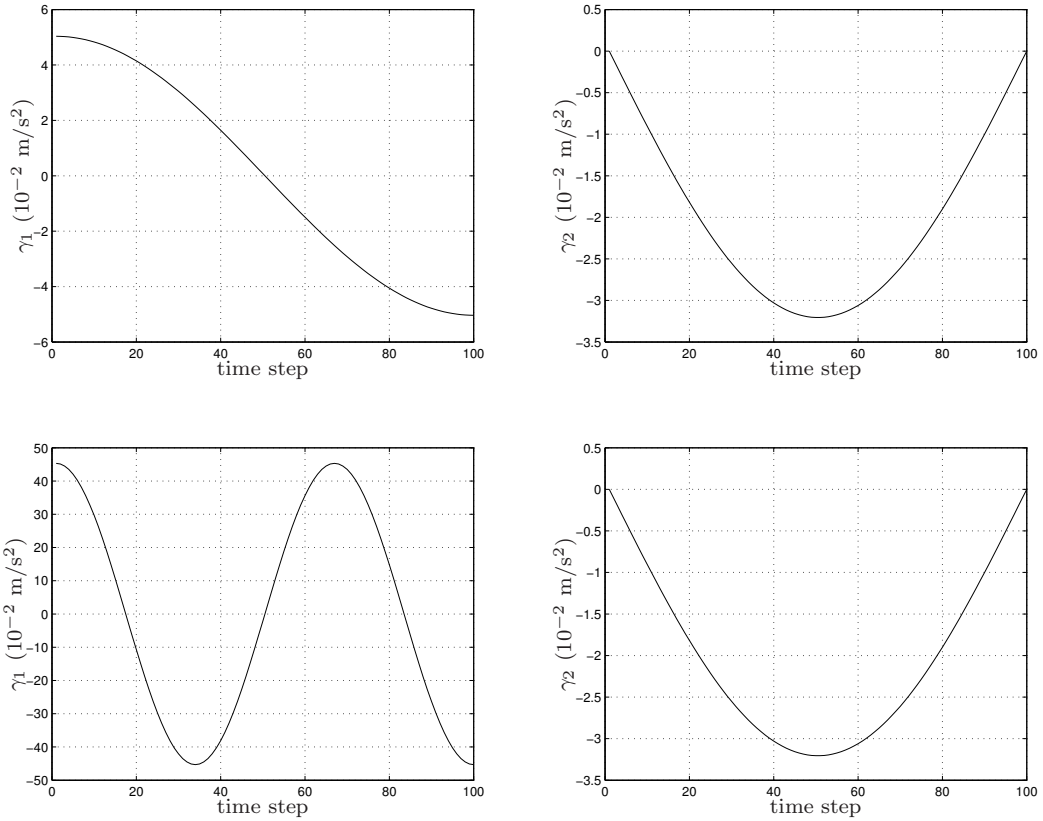


Figure 4.6: Acceleration signals for the second trajectory in the top plots and for the third trajectory in the bottom plots, γ_1 and γ_2 being the first and the second acceleration coordinates respectively

and the vo-RLS in the learning process. The four models yield close results for the first trajectory. However, for the second and third trajectories, the first-order model yields the highest estimation error. This result can be explained by the fact that this model assumes that the velocities are constant; however, the accelerations are not null and thus the velocities are not constant in the case of these two trajectories. Note that for the third trajectory, the smallest estimation error is obtained when using the third-order state-space model. This result is expected since the accelerations in this trajectory have high variations, and as explained in Subsection 4.2.2, the third-order state-space model is well suited for such cases.

4.4.2 Influence of $\boldsymbol{\sigma_\gamma}$ and σ_ρ

In this subsection, we consider the trajectory of Figure 4.5, where the general case of a hyperactive target is considered. The third-order state-space model of Section 4.3 is used since it yields the best results as shown in the previous subsection. Indeed,

Table 4.2: Estimation errors for different state-space models and the three trajectories

Tracking algorithm	Trajectory 1	Trajectory 2	Trajectory 3
KRR + first-order model	0.83	6.35	28.89
KRR + hybrid first-order model	0.91	1.15	2.26
KRR + second-order model	0.90	1.11	1.51
KRR + third-order model	0.85	1.01	0.86
vo-RLS + first-order model	1.21	7.21	31.12
vo-RLS + hybrid first-order model	1.29	1.34	2.44
vo-RLS + second-order model	1.28	1.30	1.71
vo-RLS + third-order model	1.18	1.18	1.01

even though the hybrid first-order model and the second-order model yield good results for the trajectories of Figures 4.3 and 4.4, the estimation error increases significantly compared to the third-order model when the target is hyperactive (Figure 4.5), as shown in Table 4.2.

Let us now study the influence of the noise standard deviations $\boldsymbol{\sigma_\gamma}$ and $\sigma_{\boldsymbol{\rho}}$ on the estimation error. We first take different percentages of the standard deviation of the acceleration, going from 1% to 10%, along with a fixed $\sigma_{\boldsymbol{\rho}}$ equal to 5% of the standard deviation of the RSSI measurements. The estimation errors are averaged over 50 Monte-Carlo simulations. It is worth noting that the standard deviation of the RSSIs is equal to 10.79 dBm; therefore, $\sigma_{\boldsymbol{\rho}}$ is equal to 0.54 dBm. Figure 4.7 shows the influence of $\boldsymbol{\sigma_\gamma}$ on the estimation error. One can see that the results obtained in this figure with the kernel ridge regression and the vo-RLS are independent from the acceleration noise, whereas estimations using only accelerometer information are highly affected by the variations of $\boldsymbol{\sigma_\gamma}$. One can also see that the kernel ridge regression combined with the Kalman filter yields the best results. In fact, the filter corrects the results, and the error is always smaller than the error in the case of the kernel ridge regression alone. Note that in Figure 4.7, we consider that when $\boldsymbol{\sigma_\gamma}$ is varying, the proposed tracking approach takes this variation into account. However, in practical scenarios, the acceleration noise could be higher than what the algorithm expects or even lower. In order to evaluate the robustness of our tracking approach in such cases, we consider the following example, where the used learning algorithm is the kernel ridge regression, and $\sigma_{\boldsymbol{\rho}}$ is also set to 5% of the standard deviation of the RSSI measures. In the tracking process, we consider for the trajectory several values of $\boldsymbol{\sigma_\gamma}$ different than the values used for the Kalman filter. Table 4.3 shows the considered combinations along with the obtained estimation errors in meters. One can see that the best estimation errors are obtained when we have good knowledge of the standard deviation of the additive noise. When the value of $\boldsymbol{\sigma_\gamma}$ used in the Kalman filter is different than the one used for the trajectory, the estimation error increases; however, the results are still satisfactory. Also notice that the increase in the

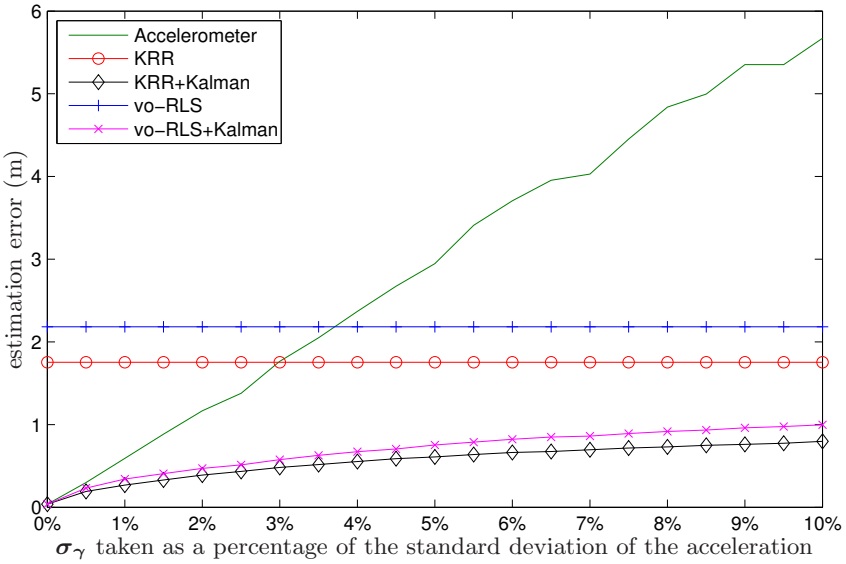


Figure 4.7: Estimation error as a function of the noise on the accelerations

Table 4.3: Estimation errors for different Kalman and trajectory values of $\boldsymbol{\sigma_\gamma}$

$(\boldsymbol{\sigma_\gamma})_{\text{trajectory}}$ \ $(\boldsymbol{\sigma_\gamma})_{\text{Kalman}}$	1%	5%	10%
1%	0.26	0.52	0.59
5%	1.01	0.48	0.58
10%	1.90	0.81	0.79

estimation errors is more important when the value of $\boldsymbol{\sigma_\gamma}$ is underestimated than when it is overestimated.

We then fix the value of $\boldsymbol{\sigma_\gamma}$ to 1% of the standard deviation of the acceleration, and we consider several percentages of the standard deviation of the RSSI measures, going from 0% to 50%; in other words, $\sigma_{\boldsymbol{\rho}}$ varies from 0 to 5.40 dBm. The estimation errors are also averaged over 50 Monte-Carlo simulations. Figure 4.8 shows the influence of $\sigma_{\boldsymbol{\rho}}$ on the estimation error. One can see that, as expected, tracking using only accelerometer information is independent from the noise on the RSSIs. The kernel ridge regression and the vo-RLS are highly affected by the noise variations, since they use these RSSI measurements for the estimation. As for the method combining the kernel ridge regression with the Kalman filter, it outperforms the method using only accelerations. It is interesting here to see the effectiveness of the Kalman filter. Indeed, one can see in Figure 4.8 that the kernel ridge regression used alone yields better results than the vo-RLS also used alone; however, after adding the Kalman filter, the results of the two techniques become very similar and the error becomes almost constant for

Table 4.4: Estimation errors for different training and testing values of $\sigma_{\boldsymbol{\rho}}$

$(\sigma_{\boldsymbol{\rho}})_{\text{trajectory}}$ \ $(\sigma_{\boldsymbol{\rho}})_{\text{training}}$	5%	25%	50%
5%	0.53	1.00	1.21
25%	0.97	1.04	1.42
50%	1.90	1.63	**1.58**

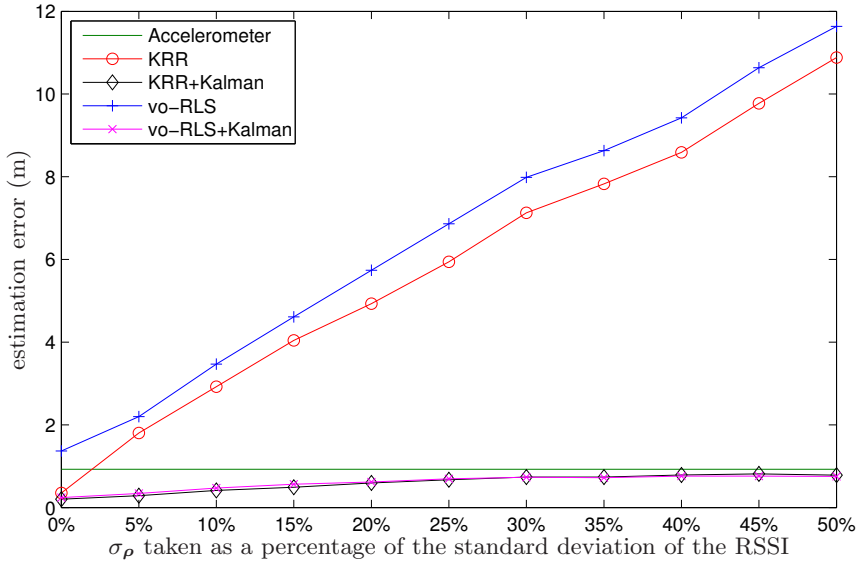


Figure 4.8: Estimation error as a function of the noise on the RSSI

both methods when $\sigma_{\boldsymbol{\rho}}$ exceeds 30% of the standard deviation of the RSSI measures. Note here as well that, in Figure 4.8, we consider the same noise standard deviation $\sigma_{\boldsymbol{\rho}}$ for the generation of the target's RSSIs along the trajectory and for the RSSIs of the training phase. However, $\sigma_{\boldsymbol{\rho}}$ which affects the trajectory might be different than the one considered for the training phase. We now set $\boldsymbol{\sigma_{\gamma}}$ to 5% of the standard deviation of the acceleration; then, we consider several combinations of $\sigma_{\boldsymbol{\rho}}$ for the trajectory and for the training. Table 4.4 shows the obtained estimation errors in meters, when using the kernel ridge regression with Kalman. One can see that the estimation error increases when considering a training value for $\sigma_{\boldsymbol{\rho}}$ different than the trajectory value. Indeed, for a given noise standard deviation of the trajectory, the best estimation is obtained when the training phase is using the same amount of noise. However, notice that for a given training value of $\sigma_{\boldsymbol{\rho}}$, the smaller $\sigma_{\boldsymbol{\rho}}$ on the trajectory is, the smaller the estimation error is. One can conclude here that the best estimation is obtained when using the exact parameters of the tracking noises, if they are known. Otherwise, it is better to overestimate the noises parameters than to underestimate them. Nevertheless, the method remains robust in all cases.

Table 4.5: Estimation errors for a random distribution of fixed sensors and reference positions

Tracking algorithm	Mean error	σ_{MSE}
KRR + third-order model	1.47	0.44
vo-RLS + third-order model	1.56	0.46

4.4.3 Influence of the fixed sensors and the reference positions

In this subsection, we also consider the trajectory of Figure 4.5, with both components of $\boldsymbol{\sigma_\gamma}$ set to 0.01 m/s^2 and $\sigma_{\boldsymbol{\rho}}$ set to 1 dBm. We first study the influence of the distribution of the 16 fixed sensors and the 100 reference positions on the performance of the tracking method. In the previous subsections, we considered a uniform grid distribution of the fixed sensors and the reference positions. We now consider a random distribution, instead of the uniform grid, to see the influence of such a choice on the tracking method. We repeat the experiment 50 times for the kernel ridge regression and the vo-RLS, using the third-order state-space model. The mean estimation error is shown in Table 4.5, where σ_{MSE} is the standard deviation of the mean estimation error. One can see that the obtained estimation errors are higher than the ones in Table 4.2, where a distribution over a uniform grid is considered. Nevertheless, the results are still very accurate. Indeed, as we already explained in Subsection 3.5.1 of Chapter 3, the reason behind the increase of the estimation error is that a distribution over a uniform grid allows a better coverage of the surveillance area, while a random distribution does not always guarantee a good coverage of the area.

We now study the influence of the number of fixed sensors N_s and the number of reference positions N_p on the performance of the tracking method. We only use the kernel ridge regression in the following. Nevertheless, it is worth noting that varying N_s and N_p has the same influence on the tracking method if we use the vo-RLS in the learning process. We first vary the number of fixed sensors, such that $N_s = 1^2, 2^2, \ldots, 15^2$, while keeping a fixed number for the reference positions $N_p = 100$. Figure 4.9 shows the evolution of the estimation error in terms of the number of fixed sensors. We then set the number of fixed sensors N_s to 16, and we vary the number of reference positions, such that $N_p = 5^2, 6^2, \ldots, 25^2$. Figure 4.10 shows the evolution of the estimation error in terms of the number of reference positions. By comparing the obtained results, one can notice that when increasing the number of fixed sensors or the number of reference positions, we obtain a better estimation of the target's positions. Indeed, Figure 4.9 shows that when using 16 fixed sensors, the average over 50 simulations of the estimation error is 0.87 m compared to an error of 0.66 m when using 11^2=121 or 12^2=144 fixed sensors. On the other hand, Figure 4.10 shows that for $N_p = 100$, the average over 50 simulations

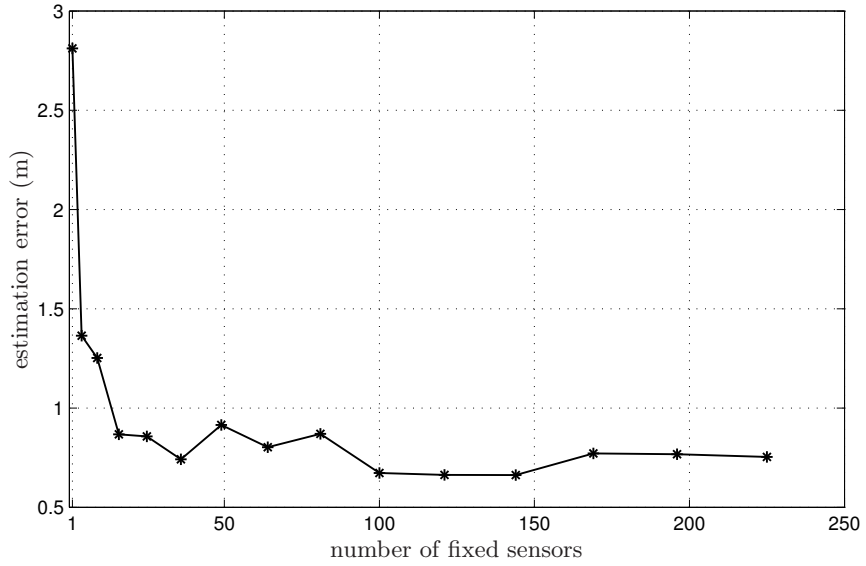


Figure 4.9: Estimation error as a function of the number of fixed sensors

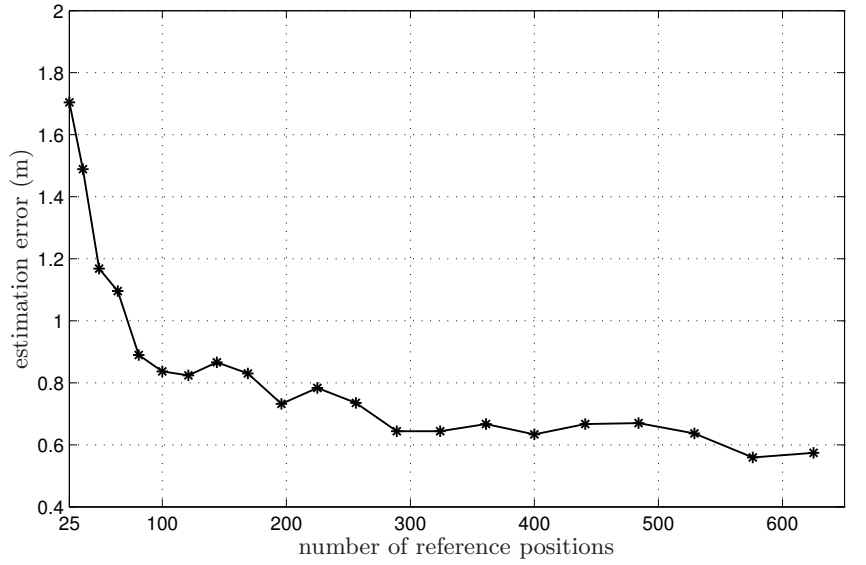


Figure 4.10: Estimation error as a function of the number of reference positions

of the estimation error is 0.84 m compared to an error of 0.56 m when increasing N_p to 24^2=576. In fact, with a higher number of fixed sensors and reference positions, we get better coverage and knowledge of the environment, which explains the improvement in the results. However, increasing the number of fixed sensors increases the total cost in material, while increasing the number of reference positions induces a significant increase in the algorithm's complexity.

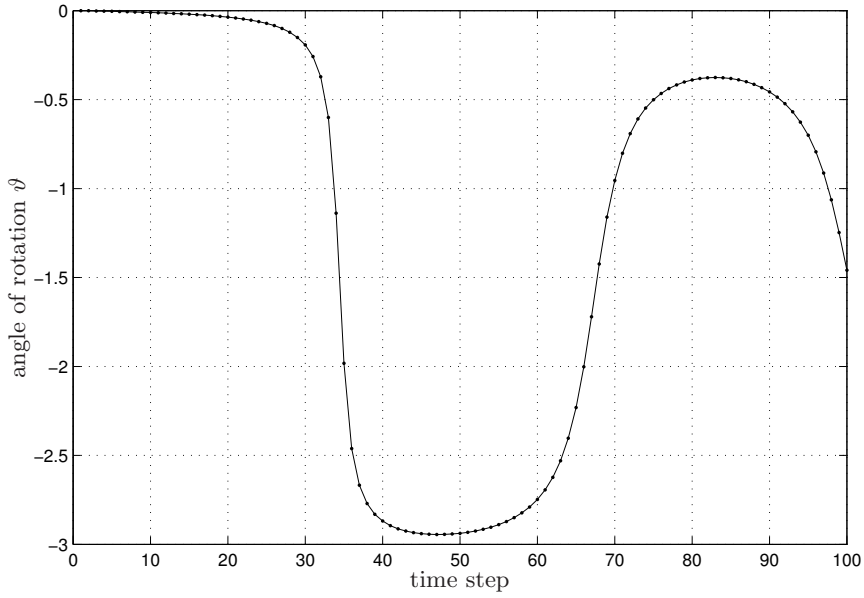


Figure 4.11: Angle of rotation as a function of time

4.4.4 Extension to a rotating target

In all the above subsections, we considered that the target is rotationally constrained. However, as we explained in Subsection 4.2.2, the target could rotate during its motion, and its coordinate system, where the accelerations are given, might change. In order to study the effect of the rotation on the tracking, we consider the two-dimensional trajectory of Figure 4.5 and we assume that the target starts with an orientation equal to 0, then it rotates following its trajectory. Figure 4.11 shows the target's angle of rotation ϑ, in radians, at each time step.

Now let both components of $\boldsymbol{\sigma_\gamma}$ be equal to 0.01 m/s^2 and $\sigma_{\boldsymbol{\rho}}$ to 1 dBm. First, we estimate the target's trajectory using its measured accelerations, without taking its rotations into account. In this case, the estimation error is around 15 m. As expected, the use of the measured accelerations of a rotating target highly degrades the estimation when the rotation is not taken into account. Then, having the target's accelerations in its coordinate system, we compute its accelerations in the initial coordinate system using the transformation given in (4.6) and the target's measured angle of rotation ϑ. In the case of a noiseless measured angle ϑ, the mean estimation error averaged over 50 simulations is equal to 0.91 m. We can obviously see that in the case of a rotating target, finding its accelerations in the initial coordinate system leads to an accurate tracking process.

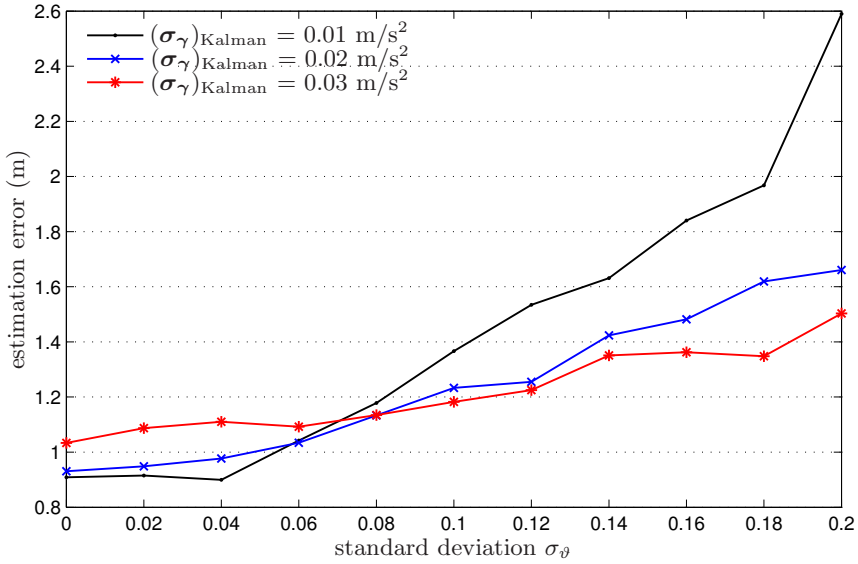


Figure 4.12: Estimation error as a function of the noise on the target's orientation

Now let σ_ϑ denote the standard deviation of the zero-mean gaussian noise that affects the measured target's orientation. We aim at studying the influence of σ_ϑ on the estimation error. Figure 4.12 shows the estimation error (in meters) when σ_ϑ is varying between 0 rad and 0.2 rad, while considering several values of $\boldsymbol{\sigma_\gamma}$. The value of the noise affecting the trajectory is set to 0.01 m/s^2. On the other hand, when using the Kalman filter in the tracking process, we consider three values for $\boldsymbol{\sigma_\gamma}$: 0.01 m/s^2, 0.02 m/s^2, and 0.03 m/s^2. One can see for all three plots that the estimation error increases with the increase of σ_ϑ. This increase of the estimation error is essentially due to the fact that the Kalman filter does not take into account the noise over orientations. However, the increase is significantly more important for the first plot, where $(\boldsymbol{\sigma_\gamma})_{\text{Kalman}}$ is equal to $\boldsymbol{\sigma_\gamma}$ on the trajectory, that is 0.01 m/s^2. The error is smaller in the other two plots, namely when σ_ϑ is increased. Indeed, this can be explained by the fact that the Kalman filter is given more flexibility when increasing $(\boldsymbol{\sigma_\gamma})_{\text{Kalman}}$ without increasing $\boldsymbol{\sigma_\gamma}$ on the trajectory.

4.4.5 Comparison to other tracking techniques

In this subsection, the objective is to compare the proposed method to two recently proposed tracking methods. Note that for simplicity, the target is again assumed rotationally constrained. For the first comparison, we use the method proposed in [Chan et al., 2009], which also makes use of the Kalman filter to correct the trajectory estimated by radio-fingerprints. We then compare our method to the centralized version of

the method described in [Dias and Bruno, 2012], which involves the use of a particle filter and RSSI measurements. We consider the three trajectories described in Subsection 4.4.1. In order to have a fair comparison of our technique towards these two methods, we consider a setup that is the closest possible to the one the authors use in their papers. For this purpose, we take the number of fixed sensors $N_s = 4$, even though taking $N_s = 16$ gives better results in the case of our method as one can see from Figure 4.9. Several values of σ_ρ are considered, with both components of $\boldsymbol{\sigma}_{\boldsymbol{\gamma}}$ set to 0.01 m/s^2.

We proceed by briefly describing the method in [Chan et al., 2009]. It consists of estimating the position using the weighted K-nearest neighbor (WKNN) algorithm, then applying the Kalman filter to enhance the estimation. As we previously explained in Subsection 3.5.4 of Chapter 3, the target's first position estimate using WKNN is given by a weighted combination of the K nearest neighboring positions from the training database, with the nearness indicator being based on the Euclidean distance between RSSIs. The weights used for the WKNN algorithm in [Chan et al., 2009] are given by:

$$w_n = \frac{1/\text{dist}_n}{\sum_{z\in\mathcal{I}} 1/\text{dist}_z}, \tag{4.18}$$

where dist_n is the Euclidean distance between the RSSI vector $\boldsymbol{\rho}(k)$ of the target at time step k and $\boldsymbol{\rho}_n$, $n \in \mathcal{I}$, and $\mathcal{I}$ is the set of indices of $\boldsymbol{\rho}_\ell$ of the database yielding the K smallest distances dist_ℓ (i.e., the K nearest neighbors) at time step k. The estimated target's position is then given by: $\sum_{n\in\mathcal{I}} w_n \boldsymbol{p}_n$. The number of neighbors K is taken equal to 8 as in the simulations of Chan et al. [2009]. As for the correction using the Kalman filter, we use a second-order state-space model similar to the one in Section 4.3. The estimation errors (in meters) obtained when using this algorithm for the three trajectories of Subsection 4.4.1 and for σ_ρ =1 dBm are computed 50 times for each case, and their averages are shown in Table 4.6. The estimation errors using our method with the second-order and the third-order state-space models are also stored in Table 4.6. Table 4.6 also shows the mean estimation error obtained when using the third trajectory, for different values of σ_ρ and for both components of $\boldsymbol{\sigma}_{\boldsymbol{\gamma}}$ taken equal to 0.01 m/s^2. Our method clearly outperforms the WKNN-based one. Indeed, the estimation error obtained with the proposed method, using a second-order or a third-order state-space model, is significantly smaller than the one obtained with the WKNN algorithm followed by the Kalman filter, for all three types of trajectories.

As for the second method used for our comparison, it employs a particle filter (PF) along with RSSI measurements and a first-order state-space model [Dias and Bruno, 2012]. As we already explained in Chapter 1, the particle filter approximates the minimum mean-square error (MMSE) estimate of the emitter state given all present and past observations, i.e., RSSI measurements. The objective of this filter is to represent the

Table 4.6: Estimation errors for different trajectories and different values of σ_ρ

Tracking algorithm	Trajectory 1	Trajectory 2	Trajectory 3		
	$\sigma_\rho = 1$	$\sigma_\rho = 1$	$\sigma_\rho = 1$	$\sigma_\rho = 2$	$\sigma_\rho = 3$
PF + second-order model	1.32	6.09	7.79	7.26	7.86
WKNN + second-order model	1.33	4.67	3.98	4.83	5.54
KRR + second-order model	1.31	2.04	2.96	4.23	4.83
KRR + third-order model	1.26	1.71	1.41	2.09	2.13

posterior distribution of the hidden states by a properly weighted set of time-varying random samples. Given all present and past network measurements, the weighted average of these samples converges at each time step, in some statistical sense, to the true global MMSE estimate of the current unknown states [Dias and Bruno, 2012]. Note that we used at first the first-order state-space model as described in the authors work. However, this model did not work well for the second and third trajectories, due to the abrupt variations in the target's motion. Therefore, we used the second-order state-space model for all three trajectories. Table 4.6 shows the mean estimation errors obtained with the method in [Dias and Bruno, 2012], when using different trajectories and different values of σ_ρ in dBm. Both components of $\boldsymbol{\sigma_\gamma}$ are taken equal to 0.01 m/s^2. One can see that our tracking method also outperforms this tracking technique based on particle filtering.

4.5 Conclusion

We proposed in this chapter a new method for target tracking in wireless sensor networks by combining machine learning and Kalman filtering. For the learning process, we used two kernel-based machine learning algorithms: the kernel ridge regression and the vector-output regularized least squares. We also described three different orders for the state-space models to be used in the Kalman filtering, and highlighted the difference between them and how they can affect the performance of the tracking procedure. In addition, we provided an extension to the case of a rotating target. Simulation results showed that the proposed method outperforms recently developed methods. The method allows accurate tracking, and is proved to be robust in the case of noisy data, whether the noise affects the acceleration information or the RSSI measures. It also proved to be robust when the target's orientation changes during its motion.

The main limitation of the proposed method is that the sensors should remain fixed at the same locations during the tracking process, which can be binding for some applications. The next chapter introduces a position-free tracking method, where a solution is provided to the stationarity problem of the sensors.

Chapter 5

Position-Free Tracking

Contents

In this chapter, we propose to model the relationship between the distances separating sensors and the RSSIs of the signals exchanged by these sensors. To this end, we define two kernel-based regression models; the first one is non-parametric, without any prior knowledge of the underlying physical system, while the second one is semi-parametric, combining the well-known log-distance propagation model with a non-linear fluctuation term. We then propose two tracking approaches that use either one of the defined RSSI/distance models. These models allow us to estimate the distances separating the considered target from the sensors in the network. The target's position is then estimated by combining the inertial information with the estimated distances, using either the Kalman filter or the particle filter. The accuracy of the proposed distance models is discussed, as well the effectiveness of the two proposed tracking methods. Comparisons to recently proposed methods are also provided.

5.1 Introduction

In Chapter 3 and Chapter 4, we took advantage of kernel-based machine learning to define a model that takes as input the RSSIs of the signals exchanged between a moving target and some fixed sensors in the network, and yields as output the position of the target. In this chapter, instead of using the defined kernel-based position model, we propose two kernel-based models that compute the distances separating the target from the sensors. Moreover, the most important reason for switching to distance models is that, by doing so, the sensors are allowed to move in the network, without affecting the tracking, as it will be shown in the following.

As we already mentioned in Chapter 1, distance estimation using RSSI can be really challenging, since the measurements of signals' powers could be significantly altered by the presence of additive noise, multipath fading, shadowing, and other interferences. Therefore, it becomes important to find a mathematical model which can accurately describe the relationship between the RSSI values and the distances. Several models have been proposed in the literature to characterize the relationship between the RSSIs and the distances. A popular one is the Okumura-Hata model, also known as the log-distance propagation model [Medeisis and Kajackas, 2000; Patwari et al., 2005; Zanella and Bardella, 2014]. Even though this model has many limitations, it is still widely used because of its simplicity. However, this model is basically for outdoors, since it predicts the signal strength without taking the surrounding environment into account, such as multipath propagation and attenuations due to walls. Therefore, it becomes inaccurate in cases where there is no line of sight between sensors. Different models have been proposed to overcome this problem, such as modified versions of the log-distance model

[Seidel and Rappaport, 1992; Sandeep et al., 2008], in which the attenuations due to floors and walls are explicitly included. Several researchers have determined a mathematical relationship between RSSIs and distances, without taking physical properties into consideration [Wang et al., 2003; Yang and Chen, 2009]. To this end, an empirical model is estimated with a polynomial regression. Depending on the applications and the considered environment, several other models can also be found in the literature [Neskovic et al., 2000; Sarkar et al., 2003; Andrade and Hoefel, 2010].

Once the distances are estimated from the RSSIs, they are combined using for instance trilateration [Manolakis, 1996] to find the sensors' positions. For illustration, refer to Figure 1.4 on page 11. One can also use methods based on filtering, since they can help in smoothing the target's trajectory, reducing thus the estimation error. For instance, a tracking approach using the extended Kalman filter with the distances and acceleration information is proposed in [Shareef and Zhu, 2009; Correa et al., 2014]. However, as we previously explained for this filter, the linearization and the approximations might lead to inaccurate estimation [UmaMageswari et al., 2012]. The particle filter can also be used for tracking in this context. For example, the tracking methods described in [Gustafsson et al., 2002; Wang et al., 2007; Farmani et al., 2012] take advantage of such a filter, by using the target's motion and the distances as well.

In this chapter, we propose two original regression models that estimate the distances separating sensors using their received signal strength indicators, with a non-parametric formulation and a semi-parametric one. By investigating these models, we propose to solve the tracking problem using two new methods. Our approach relies on a training phase and a tracking phase. In the first phase, we start by constructing a training database composed of RSSI measures and distances based on the radio-fingerprinting concept. Then, the distance models are defined using the collected database. Defining the distance models is an essential step in the tracking process; in fact, in the following, the estimated distances obtained using one of the proposed distance models will be used to determine the target's position. The first proposed distance model is non-parametric, not needing any prior knowledge of the physical system, while the second one is a semi-parametric regression model, combining the well-known log-distance theoretical propagation model with a non-linear fluctuation term, estimated within the framework of kernel-based machines. Each model takes the RSSIs as input and yields the distance to the corresponding sensor as output. Once the training phase is achieved, the localization and tracking can begin. In the tracking phase, a moving target collects RSSI measures and uses them with one of the computed kernel-based models to estimate the distances separating it from the sensors having known locations. Based on these distances, the target's position is estimated using two different methods. The first one consists in combining these distance estimates with the acceleration information, by

means of a Kalman filter, to determine the target's position. The second one consists in localizing the target using a particle filter with the distance estimates and the acceleration information. In the simulations, we examine the performance of the two proposed distance models, and we evaluate the two proposed tracking methods, while providing a comparison to state-of-the-art tracking methods.

5.2 Problem Statement

The tracking approach proposed in this chapter is centralized like the previous chapter. Nevertheless, one could easily consider a clusterized strategy as we explained in Section 3.4 of Chapter 3. In this section, we start by describing the network's configuration; then, in the following subsections, we provide descriptions of both training and tracking phases of the proposed tracking approach.

5.2.1 Network configuration

Consider an environment of δ dimensions, with $\delta = 2$ for a two-dimensional environment, and N_s sensors having known locations. Unlike the tracking approach presented in Chapter 4, the sensors are now allowed to move in the network. Their locations are considered to be known at any given time step, and are denoted by $\boldsymbol{s}_i(k) = (s_{i,1}(k) \dots s_{i,\delta}(k)),\ i \in \{1, \dots, N_s\}$, k being the current time step. The objective is to track a moving target with the unknown position $\boldsymbol{x}(k) = (x_1(k) \dots x_\delta(k))$. Here as well, we will only consider the case of one cooperative moving target. Nevertheless, the tracking approach could be applied in the case of several moving targets. Table 5.1 lists all the variables that will be used in this chapter, along with their respective sizes.

5.2.2 The training phase

The aim of the training phase is to determine the distance models. These models will then be used in the tracking phase to estimate the distances separating the target from the sensors in the network. The first step of this phase consists in defining a training set of N_p radio-fingerprints, where the distances are associated with the measured RSSIs. To this end, N_p reference positions, denoted by $\boldsymbol{p}_\ell = (p_{\ell,1} \dots p_{\ell,\delta}),\ \ell \in \{1, \dots, N_p\}$, are either placed on a grid or randomly in the studied environment. The N_s sensors continuously broadcast signals in the network at a fixed initial power, and a sensor is placed consecutively at the reference positions to detect the broadcasted signals and measure their RSSIs. Let $\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell},\ i \in \{1, \dots, N_s\}$, denote the power of the signal received

Table 5.1: List of the used variables in Chapter 5, along with their respective sizes

Notation	Variable	Size
δ	environment dimension	1
N_s	number of fixed sensors	1
N_p	number of reference positions	1
k	time step	1
$\chi(\cdot)$	proposed distance model	1
$\boldsymbol{s}_i$	position of fixed sensor i	$1 \times \delta$
$\boldsymbol{x}$	target position	$1 \times \delta$
$\boldsymbol{\nu}$	target velocity	$1 \times \delta$
$\boldsymbol{\gamma}$	target acceleration	$1 \times \delta$
$\boldsymbol{p}_\ell$	reference position	$1 \times \delta$
$\boldsymbol{x}^m$	position of particle m	$1 \times \delta$
$\boldsymbol{\nu}^m$	velocity of particle m	$1 \times \delta$
$\boldsymbol{\epsilon}_{\mathrm{PF}}$	error over particles positions	$1 \times \delta$
$\boldsymbol{z}_{\mathrm{KF}}$	Kalman filter observation	$(N_s - 1) \times 1$
$\boldsymbol{n}$	observation noise	$(N_s - 1) \times 1$
$\boldsymbol{z}_{\mathrm{PF}}$	particle filter observation	$N_s \times 1$
$\boldsymbol{\Lambda}$	log distance vector	$N_s \times 1$
$\boldsymbol{\alpha}_i$, $\boldsymbol{\beta}_i$	learning coefficients	$N_p \times 1$
$\boldsymbol{X}$, $\widehat{\boldsymbol{X}}$, $\widehat{\boldsymbol{X}}^-$	target state	$2\delta \times 1$
$\boldsymbol{B}$	control-input vector	$2\delta \times 1$
$\boldsymbol{\epsilon}_{\mathrm{KF}}$	state noise	$2\delta \times 1$
$\boldsymbol{H}$	covariance matrix	$\delta \times \delta$
$\boldsymbol{Q}$, $\boldsymbol{T}$, $\boldsymbol{T}^-$, $\boldsymbol{V}$	covariance matrix	$2\delta \times 2\delta$
$\boldsymbol{A}$	state transition matrix	$2\delta \times 2\delta$
$\boldsymbol{C}$	observation matrix	$(N_s - 1) \times 2\delta$
$\boldsymbol{R}$	covariance matrix	$(N_s - 1) \times (N_s - 1)$
$\boldsymbol{K}$	Gram matrix	$N_p \times N_p$

from the sensor at position $\boldsymbol{s}_i$ by the sensor at reference position $\boldsymbol{p}_\ell$, and let $d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}$, $i \in \{1, \ldots, N_s\}$, denote the distance separating these sensors. The distance $d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}$ is then given by:

$$d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell} = \sqrt{\sum_{v=1}^{\delta} \left(s_{i,v} - p_{\ell,v}\right)^2}. \tag{5.1}$$

In this way, N_s training sets of N_p pairs $(\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}, d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell})$ are obtained. The training phase consists then in finding a set of N_s models

$$\chi_i \colon \mathbb{R} \to \mathbb{R}, \ i \in \{1, \ldots, N_s\}.$$

Each model χ_i takes as input the RSSI received from the sensor $\boldsymbol{s}_i$ at a reference position $\boldsymbol{p}_\ell, \ell \in \{1, \ldots, N_p\}$, and yields as output the distance $d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}$ separating $\boldsymbol{s}_i$ from $\boldsymbol{p}_\ell$. In this chapter, we propose two different definitions for these distance models: a non-parametric

one and a semi-parametric one. More details will be provided in Section 5.3.

5.2.3 The tracking phase

In the tracking phase, a moving target, at an unknown position $\boldsymbol{x}(k)$, collects the RSSIs of the signals received from the N_s sensors, at a given time step k. Let $\rho_{\boldsymbol{s}_i(k),\boldsymbol{x}(k)}$ denote the RSSI of the signal received from the sensor at position $\boldsymbol{s}_i(k)$ by the target at position $\boldsymbol{x}(k)$. Then, an estimate of the distance $\widehat{d}_{\boldsymbol{s}_i(k),\boldsymbol{x}(k)}$, $i \in \{1,\ldots,N_s\}$, separating the sensor at position $\boldsymbol{s}_i(k)$ from the target is then obtained using the already-defined distance model χ_i and the collected RSSIs, as follows:

$$\widehat{d}_{\boldsymbol{s}_i(k),\boldsymbol{x}(k)} = \chi_i(\rho_{\boldsymbol{s}_i(k),\boldsymbol{x}(k)}). \tag{5.2}$$

The target is assumed to be equipped with a gyroscope and an accelerometer that yield respectively its instantaneous orientation and accelerations. Combining this information leads to the instantaneous accelerations of the target over the δ coordinates in the global coordinate system. We propose then to solve the tracking problem in two ways. The first solution is given in Section 5.4, where the distance estimates are combined with the accelerometer information to obtain an estimate of the target's positions using the Kalman filter. The second solution is given in Section 5.5, where the target's position estimates are obtained by combining the accelerations and the distance estimates using the particle filter.

5.3 Definition of χ_i using Kernel Methods

The objective of this section is to define the set of models $\chi_i,\ i \in \{1,\ldots,N_s\}$, using the constructed training database. Each model χ_i is defined in such a way that it associates to each RSSI measure $\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell},\ \ell \in \{1,\ldots,N_p\}$, the corresponding distance $d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}$. To this end, we propose two distance models. In the first subsection, we define the set of models $\chi_i,\ i \in \{1,\ldots,N_s\}$, as non-parametric models, i.e., they are taken as linear combinations of kernels centered on the training samples. The second models, described in the second paragraph of this section, are semi-parametric regression models that combine the physical log-distance propagation model of [Medeisis and Kajackas, 2000; Patwari et al., 2005] with a non-linear fluctuation term, defined in a reproducing kernel Hilbert space. This non-linear term compensates for the missing factors in the log-distance model, therefore allowing a better modeling of the RSSI/distance relationship.

5.3.1 Non-parametric regression models

In this subsection, the distance model is derived using only the training database. No assumptions on the model or prior knowledge of the model are considered. Therefore, determining χ_i requires solving a non-linear regression problem. To this end, we take advantage of kernel methods by considering the kernel ridge regression to determine the functions χ_i, as studied in details in Section 2.4 on page 2.4.

Each function χ_i is determined by minimizing the mean quadratic error between the model's outputs $\chi_i(\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell})$ and the desired outputs $d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}$, as follows:

$$\chi_i = \arg\min_{\underline{\chi}_i \in \mathcal{H}} \frac{1}{N_p} \sum_{\ell=1}^{N_p} (d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell} - \underline{\chi}_i(\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}))^2 + \eta_i \|\underline{\chi}_i\|_{\mathcal{H}}^2, \tag{5.3}$$

where $\mathcal{H}$ is a reproducing kernel Hilbert space, and η_i is a regularization parameter that controls the tradeoff between the training error and the complexity of the solution. According to the representer theorem, given in Section 2.3 on page 30, the optimal function can be written as follows:

$$\chi_i(\cdot) = \sum_{\ell=1}^{N_p} \alpha_{\ell,i}\, \kappa(\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}, \cdot), \tag{5.4}$$

where $\kappa : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ is a kernel, and $\alpha_{\ell,i}, \ell \in \{1,\dots,N_p\}$, are parameters to be determined. Now let $\boldsymbol{\alpha}_i$ be the $N_p\times 1$ vector whose ℓ-th entry is $\alpha_{\ell,i}$. By substituting (5.4) in (5.3), a dual optimization problem in terms of $\boldsymbol{\alpha}_i$ is obtained, whose solution is given by taking the derivative of the corresponding cost function with respect to $\boldsymbol{\alpha}_i$ and setting it to zero. The solution is then given by the following:

$$\boldsymbol{\alpha}_i = (\boldsymbol{K}_i + \eta_i N_p \mathbf{I}_{N_p})^{-1} \boldsymbol{d}_{\boldsymbol{s}_i}, \tag{5.5}$$

where $\mathbf{I}_{N_p}$ is the $N_p\times N_p$ identity matrix, $\boldsymbol{d}_{\boldsymbol{s}_i} = (d_{\boldsymbol{s}_i,\boldsymbol{p}_1} \dots d_{\boldsymbol{s}_i,\boldsymbol{p}_{N_p}})^\top$, $i \in \{1,\dots,N_s\}$, and $\boldsymbol{K}_i$ is the $N_p\times N_p$ matrix whose (u,u')-th entry is $\kappa(\rho_{\boldsymbol{s}_i,\boldsymbol{p}_u}, \rho_{\boldsymbol{s}_i,\boldsymbol{p}_{u'}})$, for $u,u' \in \{1,\dots,N_p\}$.

Notice that this non-parametric model does not consider any prior knowledge on the physical properties of the relationship between the RSSIs and the distances. Indeed, the model does not take into account the fact that this RSSI/distance relationship is logarithmic. In the following subsection, we propose a semi-parametric regression model in such a way that the prior knowledge of the logarithmic RSSI/distance relationship is taken into consideration.

5.3.2 Semi-parametric regression models

Both theoretical and measurements-based studies indicate that the received signal strength decreases logarithmically with distance. However, even though the log-distance signal propagation model takes this fact into account, it is still insufficient to characterize the RSSI/distance relationship. In fact, this model assumes that the latter is fully parametric, neglecting then all other factors in the environment, such as physical obstacles, multipath propagation, interferences, etc. It therefore characterizes the RSSI/distance relationship as follows:

$$\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell} = \rho_{0i} - 10\, n_P\, \log_{10} \frac{d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}}{d_{0i}}, \tag{5.6}$$

where ρ_{0i} is the power at the reference distance d_{0i} associated with the sensor at position $\boldsymbol{s}_i$, and n_P is the path-loss exponent. Note that the quantities ρ_{0i}, d_{0i} and n_P are considered unknown. In this subsection, we propose a new propagation model by adding to this model a non-linear fluctuation term φ_i, that represents a combination of all unknown factors affecting the RSSI measures, as follows:

$$\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell} = \rho_{0i} - 10\, n_P\, \log_{10} \frac{d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}}{d_{0i}} + \varphi_i, \tag{5.7}$$

This term provides the log-distance physical model with further flexibility, resulting in a more accurate model. Rewriting (5.7) yields the following:

$$\log_{10} d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell} = \frac{\rho_{0i}}{10\, n_P} + \log_{10} d_{0i} - \frac{\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}}{10\, n_P} + \frac{\varphi_i}{10\, n_P}. \tag{5.8}$$

One can see that (5.8) is a combination of a linear model in terms of $\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}$ and a non-linear model. Let $\psi_i(\cdot)$ denote the model that associates to each RSSI $\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}$ the logarithm of the distance $\log_{10}(d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell})$. This model $\psi_i(\cdot)$ can be decomposed into two terms: a linear term and a non-linear fluctuation term, such that:

$$\psi_i(\cdot) = \psi_{i,\mathrm{lin}}(\cdot) + \psi_{i,\mathrm{nlin}}(\cdot),$$

where we have the following:

$$\begin{cases} \psi_{i,\mathrm{lin}}(\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}) = a_i\, \rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell} + b_i, \\ \psi_{i,\mathrm{nlin}}(\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}) = \frac{\varphi_i}{10\, n_P}, \end{cases} \tag{5.9}$$

with $a_i = -\frac{1}{10\, n_P}$ and $b_i = \frac{\rho_{0i}}{10\, n_P} + \log_{10} d_{0i}$, a_i and b_i being unknown variables that need to be determined. We now consider that the non-linear term, given by $\psi_{i,\mathrm{nlin}}(\cdot)$, lies in a reproducing kernel Hilbert space denoted by $\mathcal{H}$, and induced by a positive definite kernel function $\kappa(\cdot,\cdot)$. It is estimated by minimizing the regularized mean quadratic error on

the training set, given by:

$$\frac{1}{N_p}\sum_{\ell=1}^{N_p}\varepsilon_{\ell,i}^2+\eta_i\|\psi_{i,\text{nlin}}\|_{\mathcal{H}}^2, \tag{5.10}$$

where the quantity η_i is a regularization parameter that controls the tradeoff between the training error and the complexity of the solution, and the cost function $\varepsilon_{\ell,i}$ is given by:

$$\varepsilon_{\ell,i}=\log_{10} d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}-a_i\,\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}-b_i-\psi_{i,\text{nlin}}(\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}),$$

where $\ell\in\{1,\ldots,N_p\}$. From the representer theorem, $\psi_{i,\text{nlin}}(\cdot)$ can be written as a linear combination of kernels as follows:

$$\psi_{i,\text{nlin}}(\cdot)=\sum_{\ell=1}^{N_p}\beta_{\ell,i}\,\kappa(\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell},\cdot),$$

where $\kappa:\mathbb{R}\times\mathbb{R}\to\mathbb{R}$, and $\beta_{\ell,i}$, with $\ell\in\{1,\ldots,N_p\}$, are parameters to be estimated.

By multiplying (5.10) by N_p for convenience and writing it in matrix form, one gets the following cost function:

$$\begin{aligned}&\boldsymbol{\Lambda}_i^\top\boldsymbol{\Lambda}_i+a_i^2\,\boldsymbol{\rho}_i^\top\boldsymbol{\rho}_i+N_pb_i^2+\boldsymbol{\beta}_i^\top\boldsymbol{K}_i^\top\boldsymbol{K}_i\boldsymbol{\beta}_i-2a_i\boldsymbol{\Lambda}_i^\top\boldsymbol{\rho}_i-2b_i\mathbf{1}_{N_p\times1}^\top\boldsymbol{\Lambda}_i\\&-2\boldsymbol{\Lambda}_i^\top\boldsymbol{K}_i\boldsymbol{\beta}_i+2a_ib_i\mathbf{1}_{N_p\times1}^\top\boldsymbol{\rho}_i+2a_i\boldsymbol{\rho}_i^\top\boldsymbol{K}_i\boldsymbol{\beta}_i+2b_i\mathbf{1}_{N_p\times1}^\top\boldsymbol{\beta}_i+\eta_i\,N_p\boldsymbol{\beta}_i^\top\boldsymbol{K}_i\boldsymbol{\beta}_i,\end{aligned} \tag{5.11}$$

where $\boldsymbol{\beta}_i=(\beta_{1,i}\ldots\beta_{N_p,i})^\top$, $\boldsymbol{K}_i$ is the $N_p\times N_p$ Gram matrix whose (u,u')-th entry is $\kappa(\rho_{\boldsymbol{s}_i,\boldsymbol{p}_u},\rho_{\boldsymbol{s}_i,\boldsymbol{p}_{u'}})$, for $u,u'\in\{1,...,N_p\}$, $\boldsymbol{\rho}_i$ is the $N_p\times1$ vector whose ℓ-th entry is equal to $\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}$, $\ell\in\{1,\ldots,N_p\}$, $\mathbf{1}_{N_p\times1}$ is the $N_p\times1$ vector of ones, and $\boldsymbol{\Lambda}_i$ is the $N_p\times1$ vector whose ℓ-th entry is equal to $\log_{10} d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}$. By taking the partial derivatives in matrix form of (5.11) with respect to a_i, b_i and $\boldsymbol{\beta}_i$ and setting them to zero, we get the following linear system having the form $\boldsymbol{U}_i=\boldsymbol{W}_i\boldsymbol{Y}_i$, where

$$\boldsymbol{Y}_i=\begin{bmatrix}a_i\\b_i\\\boldsymbol{\beta}_i\end{bmatrix},\quad\boldsymbol{U}_i=\begin{bmatrix}\mathbf{1}_{N_p\times1}^\top\boldsymbol{\Lambda}_i\\\boldsymbol{\Lambda}_i\\\boldsymbol{\Lambda}_i^\top\boldsymbol{\rho}_i\end{bmatrix},$$

$$\boldsymbol{W}_i=\begin{bmatrix}\mathbf{1}_{N_p\times1}^\top\boldsymbol{\rho}_i & N_p & \mathbf{1}_{N_p\times1}^\top\boldsymbol{K}_i\\\boldsymbol{\rho}_i & \mathbf{1}_{N_p\times1} & \boldsymbol{K}_i+\eta_iN_p\mathbf{I}_{N_p\times N_p}\\\boldsymbol{\rho}_i^\top\boldsymbol{\rho}_i & \mathbf{1}_{N_p\times1}^\top\boldsymbol{\rho}_i & \boldsymbol{\rho}_i^\top\boldsymbol{K}_i\end{bmatrix} \tag{5.12}$$

The solution is then given by:

$$\boldsymbol{Y}_i=(\boldsymbol{W}_i^\top\boldsymbol{W}_i)^{-1}\boldsymbol{W}_i^\top\boldsymbol{U}_i. \tag{5.13}$$

The model parameters are then computed using (5.12) and (5.13). Now using only the RSSI information, one can find the logarithm of any distance separating a sensor at position $\boldsymbol{s}_i(k)$ from the target at position $\boldsymbol{x}(k)$, as follows:

$$\begin{aligned}\log_{10} d_{\boldsymbol{s}_i(k),\boldsymbol{x}(k)} &= \psi_i\left(\rho_{\boldsymbol{s}_i(k),\boldsymbol{x}(k)}\right) \\ &= a_i\,\rho_{\boldsymbol{s}_i(k),\boldsymbol{p}_\ell} + b_i + \sum_{\ell=1}^{N_p} \beta_{\ell,i}\,\kappa\left(\rho_{\boldsymbol{s}_i(k),\boldsymbol{p}_\ell}, \rho_{\boldsymbol{s}_i(k),\boldsymbol{x}(k)}\right).\end{aligned}$$

Then, the model $\chi_i(\cdot)$ yields the distance as follows:

$$\chi_i\left(\rho_{\boldsymbol{s}_i(k),\boldsymbol{x}(k)}\right) = 10^{\log_{10} d_{\boldsymbol{s}_i(k),\boldsymbol{x}(k)}} = 10^{\psi_i\left(\rho_{\boldsymbol{s}_i(k),\boldsymbol{x}(k)}\right)}. \tag{5.14}$$

5.4 Resolution using the Kalman Filter

In this section, we propose to find the target position by combining its instantaneous accelerations with the distances separating it from the sensors in the network, by means of a Kalman filter. These distances are estimated using any of the kernel-based distance models defined in Section 5.3. As we already explained, the Kalman filter first predicts the unknown state of the target using the previous estimated state and its mobility equation. The predicted state is then corrected using observations from the network. Therefore, in order to use the Kalman filter, we need first to define the state-space equation. Then, we need to define the observations.

5.4.1 Definition of the filter's equations

In the previous chapter, we introduced three orders of state-space equations, and we proved that using a third-order state-space model gives the best results, especially in cases where the target is hyperactive. Consequently, in this chapter, we will only consider the third-order state-space equation from Section 4.3 on page 79. In the following, the target is assumed to be rotationally constrained. If this is not the case, the coordinate system where the accelerations are given changes. The solution to this problem has been discussed in Subsection 4.2.2 on page 76, where we showed that the target accelerations in the initial coordinate system can be computed using the target's measured orientations and a rotation matrix. Now let $\boldsymbol{X}(k) = (\boldsymbol{x}(k) \;\; \boldsymbol{\nu}(k))^\top$ denote the unknown state of the target at time step k, with $\boldsymbol{\nu}(k)$ being the estimated velocity vector of the target at the time step k. In addition, let $\boldsymbol{\gamma}(k)$ denote the target measured acceleration vector at time

step k. The considered state-space equation is then given by the following:

$$\begin{aligned}\boldsymbol{X}(k) &= \boldsymbol{A}\ \boldsymbol{X}(k-1) + \boldsymbol{B}(k) + \boldsymbol{\epsilon}_{\mathrm{KF}}(k),\\ &= \begin{pmatrix} \mathbf{I}_\delta & \Delta t \mathbf{I}_\delta \\ \mathbf{0}_{\delta\times\delta} & \mathbf{I}_\delta \end{pmatrix} \boldsymbol{X}(k-1) + \begin{pmatrix} \boldsymbol{\gamma}(k-1)^\top \frac{\Delta t^2}{3} \\ \boldsymbol{\gamma}(k-1)^\top \frac{\Delta t}{2} \end{pmatrix} + \begin{pmatrix} \boldsymbol{\gamma}(k)^\top \frac{\Delta t^2}{6} \\ \boldsymbol{\gamma}(k)^\top \frac{\Delta t}{2} \end{pmatrix} + \boldsymbol{\epsilon}_{\mathrm{KF}}(k),\end{aligned} \tag{5.15}$$

where $\boldsymbol{A}$ is the state transition matrix, $\boldsymbol{B}(k)$ is a noisy control-input vector depending on the accelerations, Δt is the tracking period, that is the time period separating two consecutive time steps $k-1$ and k, $\mathbf{I}_\delta$ is the $\delta\times\delta$ identity matrix, and $\mathbf{0}_{\delta\times\delta}$ is the $\delta\times\delta$ matrix of zeros. As for the quantity $\boldsymbol{\epsilon}_{\mathrm{KF}}(k)$, it is the state equation error, whose probability distribution is assumed to be normal, having zero mean and covariance matrix $\boldsymbol{V}$ of size $2\delta \times 2\delta$.

Having defined the state-space model, we now need to define the observation in order to use the Kalman filter. Now, if we use the equations of the estimated distances separating the target from the sensors, the obtained observation model is non-linear. To overcome this problem, we propose an alternative linear observation model based on the distances. Indeed, using (5.1), one can write the following:

$$\begin{aligned} d^2_{\boldsymbol{s}_i,\boldsymbol{x}(k)} &= \sum_{v=1}^{\delta} (s_{i,v} - x_v(k))^2 \\ &= \sum_{v=1}^{\delta} s^2_{i,v} + \sum_{v=1}^{\delta} x^2_v(k) - 2\sum_{v=1}^{\delta} s_{i,v}\ x_v(k). \end{aligned}$$

By subtracting $d^2_{\boldsymbol{s}_{N_s},\boldsymbol{x}(k)}$ from $d^2_{\boldsymbol{s}_i,\boldsymbol{x}(k)}$, $i \in \{1, ..., N_s - 1\}$, we get the following:

$$d^2_{\boldsymbol{s}_i,\boldsymbol{x}(k)} - d^2_{\boldsymbol{s}_{N_s},\boldsymbol{x}(k)} = \sum_{v=1}^{\delta} s^2_{i,v} - \sum_{v=1}^{\delta} s^2_{N_s,v} + 2\sum_{v=1}^{\delta} x_v(k)(s_{N_s,v} - s_{i,v}).$$

After rearranging the terms in this expression, it becomes possible to set a linear observation model as follows:

$$\boldsymbol{z}_{\mathrm{KF}}(k) = \boldsymbol{C}\boldsymbol{X}(k) + \boldsymbol{n}(k), \tag{5.16}$$

where $\boldsymbol{n}(k) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{R})$ is an additive noise, assumed to have a normal distribution, with zero mean and covariance matrix $\boldsymbol{R}$, and the observation vector $\boldsymbol{z}_{\mathrm{KF}}(k)$ can be written as follows:

$$\boldsymbol{z}_{\mathrm{KF}}(k) = \begin{pmatrix} d^2_{\boldsymbol{s}_1,\boldsymbol{x}(k)} - d^2_{\boldsymbol{s}_{N_s},\boldsymbol{x}(k)} + \sum_{v=1}^{\delta} s^2_{N_s,v} - \sum_{v=1}^{\delta} s^2_{1,v} \\ \vdots \\ d^2_{\boldsymbol{s}_{N_s-1},\boldsymbol{x}(k)} - d^2_{\boldsymbol{s}_{N_s},\boldsymbol{x}(k)} + \sum_{v=1}^{\delta} s^2_{N_s,v} - \sum_{v=1}^{\delta} s^2_{N_s-1,v} \end{pmatrix}.$$

The observation matrix $\boldsymbol{C}$, of size $(N_s - 1) \times 2\delta$ is then given by:

$$\boldsymbol{C} = 2 \begin{pmatrix} s_{N_s,1} - s_{1,1} & \dots & s_{N_s,\delta} - s_{1,\delta} & \\ \vdots & & \vdots & \mathbf{0}_{(N_s-1)\times\delta} \\ s_{N_s,1} - s_{N_s-1,1} & \cdots & s_{N_s,\delta} - s_{N_s-1,\delta} & \end{pmatrix},$$

where $\mathbf{0}_{(N_s-1)\times\delta}$ is the $(N_s - 1) \times \delta$ matrix of zeros. In order to approximate the covariance matrix $\boldsymbol{R}$, a new set of reference positions is generated, and their distances to the sensors are estimated using χ_i. Then, the differences between their observation vectors, using their estimated distances, and the product of the matrix $\boldsymbol{C}$ by their exact positions are computed and stored into a vector, that is the error vector. The matrix $\boldsymbol{R}$, assumed to be constant over time, is then determined by computing the covariance of the error vector.

5.4.2 Algorithm using the Kalman filter

Now that we have defined the state-space equation (5.15) and the observation equation (5.16), we can use the Kalman filter to find the target position at each time step k. The Kalman filter first predicts the unknown position using the previous estimated position with the state-space equation (5.15), as follows:

$$\widehat{\boldsymbol{X}}^{-}(k) = \boldsymbol{A}\,\widehat{\boldsymbol{X}}(k-1) + \boldsymbol{B}(k), \tag{5.17}$$

where $\widehat{\boldsymbol{X}}(0)$ is assumed to be known. Then, the predicted estimation covariance is updated by the following:

$$\boldsymbol{T}^{-}(k) = \boldsymbol{A}\,\boldsymbol{T}(k-1)\boldsymbol{A}^{\top} + \boldsymbol{Q}(k),$$

where $\boldsymbol{T}(k-1)$ is the covariance estimation at time step $k-1$, and $\boldsymbol{T}(0)$ is null since the initial state is known. As for the matrix $\boldsymbol{Q}(k)$, i.e., the covariance matrix of $\boldsymbol{X}(k)$ given $\boldsymbol{X}(k-1)$, in the case of the third-order state-space model, it is given as in Chapter 4 by:

$$\begin{aligned} \boldsymbol{Q}(k) &= \boldsymbol{V} + \mathrm{Cov}\left(\begin{pmatrix} \boldsymbol{\gamma}(k-1)^{\top} \frac{\Delta t^2}{3} \\ \boldsymbol{\gamma}(k-1)^{\top} \frac{\Delta t}{2} \end{pmatrix} + \begin{pmatrix} \boldsymbol{\gamma}(k)^{\top} \frac{\Delta t^2}{6} \\ \boldsymbol{\gamma}(k)^{\top} \frac{\Delta t}{2} \end{pmatrix} \right) \\ &= \boldsymbol{V} + \begin{pmatrix} \frac{5\Delta t^4}{36}\,\mathrm{Diag}\left(\boldsymbol{\sigma}_{\boldsymbol{\gamma}}^2\right) & \frac{\Delta t^3}{4}\,\mathrm{Diag}\left(\boldsymbol{\sigma}_{\boldsymbol{\gamma}}^2\right) \\ \frac{\Delta t^3}{4}\,\mathrm{Diag}\left(\boldsymbol{\sigma}_{\boldsymbol{\gamma}}^2\right) & \frac{\Delta t^2}{2}\,\mathrm{Diag}\left(\boldsymbol{\sigma}_{\boldsymbol{\gamma}}^2\right) \end{pmatrix}, \end{aligned}$$

where $\sigma^2_{\gamma_v}$, $v \in \{1, \ldots, \delta\}$ are the variances of the acceleration noises. As we already mentioned, their values can be estimated by performing a calibration of the accelerometer before the tracking.

The final step of the filter is the correction of the predicted quantities $\widehat{\boldsymbol{X}}^{-}(k)$ and $\boldsymbol{T}^{-}(k)$ using the observation equation (5.16) as follows:

$$\begin{aligned}\widehat{\boldsymbol{X}}(k) &= \widehat{\boldsymbol{X}}^{-}(k) + \boldsymbol{G}_{\mathrm{KF}}(k)\,(\boldsymbol{z}_{\mathrm{KF}}(k)^{\top} - \boldsymbol{C}\widehat{\boldsymbol{X}}^{-}(k)),\\ \boldsymbol{T}(k) &= (\mathbf{I}_{2\delta} - \boldsymbol{G}_{\mathrm{KF}}(k)\,\boldsymbol{C})\,\boldsymbol{T}^{-}(k),\end{aligned}$$

where $\boldsymbol{G}_{\mathrm{KF}}(k)$ is the optimal Kalman gain given by:

$$\boldsymbol{G}_{\mathrm{KF}}(k) = \boldsymbol{T}^{-}(k)\,\boldsymbol{C}^{\top}\,(\boldsymbol{C}\,\boldsymbol{T}^{-}(k)\,\boldsymbol{C}^{\top} + \boldsymbol{R})^{-1}.$$

5.5 Resolution using the Particle Filter

The objective of this section is to find the target position estimates using the particle filter, the distances estimated by either one of the kernel-based distance models of Section 5.3 and the instantaneous accelerations of the target. The particle fillter is an implementation of the formal recursive Bayesian filter using sequential Monte Carlo methods [Chen, 2003]. The aim of a Bayesian filter is to construct the posterior probability density function (pdf) of the required state vector using all available information. Then, this filter provides a formal mechanism for propagating and updating the pdf when new information (i.e., measurements) is received. However, in a particle filter scheme, instead of describing the required pdf as a functional form, it is represented approximately as a set of random samples of the pdf. As the number of samples tends to infinity, one converges to an exact equivalent of the functional form. These samples, called the particles of the filter, are propagated and updated according to the dynamics and measurement models.

5.5.1 Definition of the filter's equations

Unlike the Kalman filter, the particle filter makes no restrictive assumption about the dynamics of the state-space or the density function. The observation model can be nonlinear, while the initial state and noise distributions can take any required form. For this reason, the observations here are taken equal to the estimated distances separating the target from the sensors, that is:

$$\boldsymbol{z}_{\mathrm{PF}}(k) = \boldsymbol{d}_{x(k)},$$

where the vector of distances $\boldsymbol{d}_{x(k)} = (d_{\boldsymbol{s}_1,\boldsymbol{x}(k)} \dots d_{\boldsymbol{s}_{N_s},\boldsymbol{x}(k)})^\top$, is obtained using any one of the aforementioned kernel-based models. Like the Kalman-based method, the tracking problem is also defined, using accelerations, by the third-order state-space model.

5.5.2 Algorithm using the particle filter

In order to solve the problem using the particle filter, N_{PF} particles are first generated at the initial position of the target $\boldsymbol{x}(0)$, considered to be known. All the particles are given equal weights of $\frac{1}{N_{\text{PF}}}$. It is important to note here that the higher N_{PF}, the more computations we need. Let $\boldsymbol{x}^m(0), m \in \{1, \dots, N_{\text{PF}}\}$, denote the initial particles positions, and $\boldsymbol{\nu}^m(0)$ their initial null velocities; then the particles positions at time step k are denoted by $\boldsymbol{x}^m(k), m \in \{1, \dots, N_{\text{PF}}\}$, and their velocities by $\boldsymbol{\nu}^m(k)$. Starting from the initial position, the particles positions and their velocities at time step k are recursively predicted using (4.5) as follows:

$$\begin{aligned}
\boldsymbol{\nu}^m(k) &= \boldsymbol{\nu}^{m'}(k-1) + \boldsymbol{\gamma}(k-1)\,\Delta t + \frac{\boldsymbol{\gamma}(k) - \boldsymbol{\gamma}(k-1)}{\Delta t}\,\frac{\Delta t^2}{2} \\
&= \boldsymbol{\nu}^{m'}(k-1) + \frac{\Delta t}{2}\boldsymbol{\gamma}(k-1) + \frac{\Delta t}{2}\boldsymbol{\gamma}(k), \\
\boldsymbol{x}^m(k) &= \boldsymbol{x}^{m'}(k-1) + \boldsymbol{\nu}^{m'}(k-1)\,\Delta t + \boldsymbol{\gamma}(k-1)\,\frac{\Delta t^2}{2} + \frac{\boldsymbol{\gamma}(k) - \boldsymbol{\gamma}(k-1)}{\Delta t}\,\frac{\Delta t^3}{6} + \boldsymbol{\epsilon}_{\text{PF}}(k) \\
&= \boldsymbol{x}^{m'}(k-1) + \boldsymbol{\nu}^{m'}(k-1)\,\Delta t + \frac{\Delta t^2}{3}\boldsymbol{\gamma}(k-1) + \frac{\Delta t^2}{6}\boldsymbol{\gamma}(k) + \boldsymbol{\epsilon}_{\text{PF}}(k),
\end{aligned}$$

where $\boldsymbol{x}^{m'}(k-1)$ is one of the particles of time step $k-1$, selected randomly according to the discrete distribution of their weights, $\boldsymbol{\nu}^{m'}(k-1)$ is the velocity associated to $\boldsymbol{x}^{m'}(k-1)$, and $\boldsymbol{\epsilon}_{\text{PF}}(k) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{H})$ is the state equation error whose probability distribution is assumed to be normal, having zero mean and covariance matrix $\boldsymbol{H}$ of size $\delta \times \delta$. The particles' weights $\omega^m(k)$ are updated according to the observed distances. Indeed, a particle is more valuable if its distances to the sensors are closer to the observed ones [Farmani et al., 2012]. This update rule is formulated as follows:

$$\omega^m(k) = \frac{1}{\|\boldsymbol{d}_{\boldsymbol{x}^m(k)} - \boldsymbol{z}_{\text{PF}}(k)\|}\,\omega^{m'}(k-1),$$

where $\boldsymbol{d}_{\boldsymbol{x}^m(k)} = (d_{\boldsymbol{s}_1,\boldsymbol{x}^m(k)} \dots d_{\boldsymbol{s}_{N_s},\boldsymbol{x}^m(k)})^\top$, and $\|\boldsymbol{d}_{\boldsymbol{x}^m(k)} - \boldsymbol{z}_{\text{PF}}(k)\|$ is the distance between the vectors $\boldsymbol{z}_{\text{PF}}(k)$ and $\boldsymbol{d}_{\boldsymbol{x}^m(k)}$. Then, the weights are normalized using the following:

$$\omega^m(k) = \frac{\omega^m(k)}{\sum_{m=1}^{N_{\text{PF}}} \omega^m(k)}.$$

Finally, the last step in the filtering process is the resampling. In this step, the particles with negligible weights are replaced by new particles in the proximity of the particles with

higher weights. It is applied only when the effective number of particles N_{eff} becomes less than a threshold value N_{th}, with N_{eff} given by:

$$N_{\text{eff}} = \frac{1}{\sum_{m=1}^{N_{\text{PF}}} (\omega^m(k))^2}.$$

The threshold N_{th} is usually taken equal to 10% of the particles number N_{PF}. Having calculated the weights of the particles, the target's state at time step k is then given using the following:

$$\widehat{\boldsymbol{x}}(k) = \sum_{m=1}^{N_{\text{PF}}} \omega^m(k)\ \boldsymbol{x}^m(k).$$

Compared to the Kalman-based method, this algorithm needs more computations, due to the generation and resampling of the particles; however, it is more robust when the distances errors are non Gaussian.

5.6 Evaluation of the accuracy of the distance models

In this section, we propose to evaluate the accuracy of the two proposed distance models, with simulated and real data. Note that the Gaussian kernel is used here; in addition, the kernel bandwidths and the regularization parameters are estimated using the cross-validation technique. The proposed models are compared to the log-distance propagation model, given in (5.6), and to the polynomial model introduced in [Yang and Chen, 2009]. The polynomial regression in [Yang and Chen, 2009] determines a mathematical relation between RSSIs and distances, without taking physical properties into consideration. In such a case, the signal propagation model is the q-th degree polynomial given by the following:

$$d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell} = a_0 + a_1\, \rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell} + a_2\, \rho^2_{\boldsymbol{s}_i,\boldsymbol{p}_\ell} + \cdots + a_q\, \rho^q_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}, \tag{5.18}$$

where a_j, $j \in \{0, \ldots, q\}$, are the polynomial coefficients to be determined. These parameters a_j are chosen in a way to fit the q-th degree polynomial through the training set, using the least squares method.

5.6.1 Evaluation of the distance models on simulated data

Now let us evaluate the accuracy of the distance models on simulated data, in the case of two different scenarios. For the first scenario, we consider an area without walls, and we estimate the distances using the proposed models. Then, we compare the obtained results to the ones obtained with the log-distance propagation model. Next, we consider a different scenario, where the signals are attenuated because of the presence of walls

Table 5.2: Distance estimation errors for simulated data in the case of the first scenario, where NPM, SPM, PM, PolyM denote respectively the non-parametric model, the semi-parametric one, the log-distance propagation one and the polynomial one

	PM	PolyM $q=2$	PolyM $q=3$	PolyM $q=4$	NPM	SPM
Mean training error	2.26	2.74	2.23	2.22	2.18	2.09
Mean test error	2.59	2.94	2.57	2.60	2.57	2.59

in the proposed topology. As it will be shown in the following, such topology allows a better comparison between models.

We start with the first scenario, where a 100 m $\times$ 100 m area is considered, with $N_s = 16$ sensors and $N_p = 100$ reference positions distributed over the area. Figure 5.1 illustrates the considered topology, where no walls or obstacles are present. The RSSIs for the training phase are obtained using the log-distance propagation model given in (5.6), with n_P set to 4 as often given in the literature, and ρ_0 set to 1 dBm at the reference distance d_0 set to 1 m. As for the test phase, 100 positions are randomly generated in the studied area, and their RSSIs are also obtained using (5.6). Finally, a zero mean additive white noise $o_{i,\ell}$ is added to all the RSSIs, with $\sigma_{\boldsymbol{\rho}}$ being its standard deviation. Here, we take $\sigma_{\boldsymbol{\rho}}$ equal to 1 dBm. Based on the study given in Section 5.3, we define $N_s = 16$ non-parametric distance models and $N_s = 16$ semi-parametric distance models. Then, in the test phase, we estimate the distances using these models, the log-distance propagation model, and the polynomial model with $q = 2$, $q = 3$ and $q = 4$. The mean estimation errors (in meters) are stored in Table 5.2. For convenience, in the comparison tables, we denote the non-parametric model of Subsection 5.3.1 by NPM, and the semi-parametric model of Subsection 5.3.2 by SPM. We also denote the log-distance propagation model, that is known for being a physical model, by PM and the polynomial model by PolyM. One can see from this table that the results are really close, especially for the test error, which is of much great importance than the training error. This result was expected, since we are generating the RSSIs using the log-distance propagation model, and the noise is a zero mean additive noise. For all considered models, the objective is to minimize the error on the training set, that is the difference between the real distances and the estimated distances. Therefore, since the noise is of zero mean, all estimated models will yield the same optimal solution, i.e., the noiseless initial model given in (5.6).

We now propose another scenario given in Figure 5.2, where we consider a 25 m $\times$ 5 m area, two fixed sensors and 45 known positions for the training phase. This figure shows that there are five rooms in the studied area, meaning that the signal penetrates a

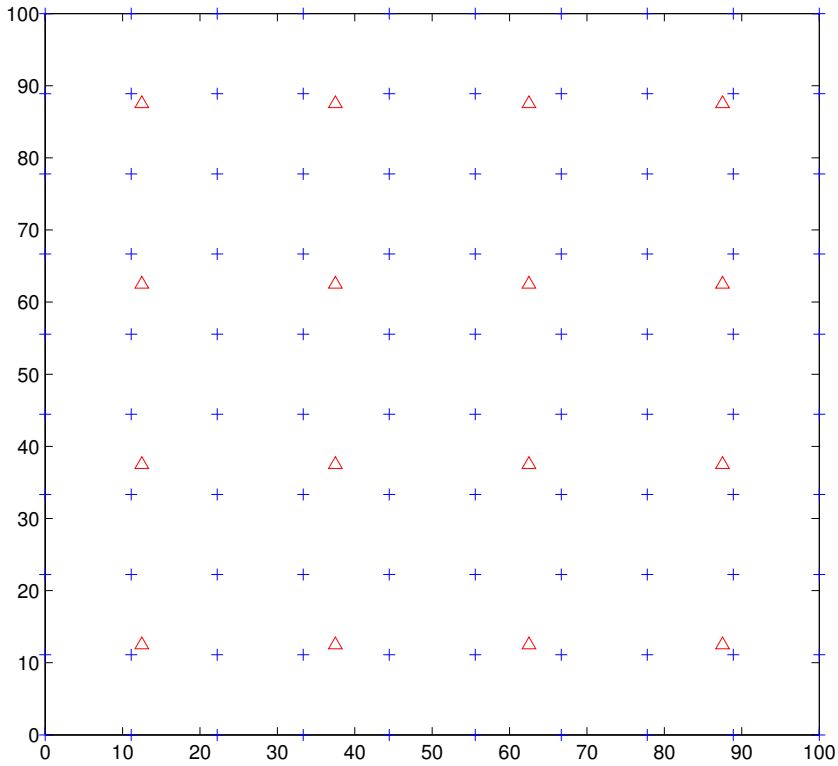


Figure 5.1: Topology of the simulated area for the first scenario, where Δ represents the sensors and + represents the training positions

maximum of four walls during its propagation. Therefore, we consider the average walls model described in [Andrade and Hoefel, 2010] to generate the RSSI measures. This model is a modified version of the log-distance model that explicitly takes into account the attenuations due to walls. The received signal strength indicator is then given by the following:

$$\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell} = \rho_0 - 10\, n_P\, \log_{10} d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell} - N_{w_i} L_{w_i} + o_{i,\ell}, \tag{5.19}$$

where the quantities L_{w_i} and N_{w_i} denote respectively the loss due to walls and the number of penetrated walls. The quantity L_{w_i} is taken equal to 6.9 dBm, since we consider the case of heavy thick walls [Andrade and Hoefel, 2010]. Now for the test phase, 100 positions are randomly generated in the studied area, and their RSSIs are obtained using (5.19). Then, the distances are estimated using the two proposed models, the log-distance propagation model and the polynomial models. The estimation errors for these models are stored in Table 5.3, when the standard deviation of $o_{i,\ell}$ is taken equal to 0.5 dBm. Table 5.4 yields the estimation errors when the noise's standard deviation is increased to 1 dBm. One can see from both tables that the two proposed

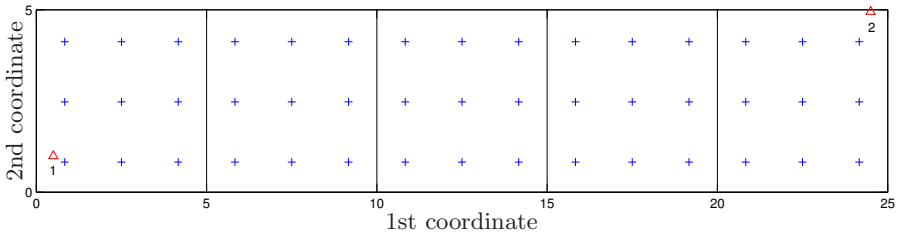


Figure 5.2: Topology of the simulated area for the second scenario, where Δ represents the sensors and $+$ represents the training positions

Table 5.3: Distance estimation errors for simulated data in the case of the second scenario with $\sigma_{\boldsymbol{\rho}}$ equal to 0.5 dBm, where NPM, SPM, PM, PolyM denote respectively the non-parametric model, the semi-parametric one, the log-distance propagation one and the polynomial one

	PM	PolyM $q=2$	PolyM $q=3$	PolyM $q=4$	NPM	SPM
Mean training error	1.26	0.56	0.55	0.54	0.20	0.22
Mean test error	1.37	0.62	0.61	0.61	0.32	0.34

Table 5.4: Distance estimation errors for simulated data in the case of the second scenario with $\sigma_{\boldsymbol{\rho}}$ equal to 1 dBm, where NPM, SPM, PM, PolyM denote respectively the non-parametric model, the semi-parametric one, the log-distance propagation one and the polynomial one

	PM	PolyM $q=2$	PolyM $q=3$	PolyM $q=4$	NPM	SPM
Mean training error	1.29	0.68	0.67	0.67	0.39	0.43
Mean test error	1.39	0.66	0.65	0.66	0.51	0.52

distance models outperforms the log-distance propagation model and the polynomial one in terms of accuracy. Moreover, it is the non-parametric model that yields the best results.

5.6.2 Evaluation of the distance models on real data

For the evaluation of the proposed distance models on real data, we use the same collected measurements considered in Section 3.5 of Chapter 3. This set of collected data is available from [Zanca et al.]; the measurements are carried out in a room of approximately 10 m $\times$ 10 m, where 48 EyesIFX sensor nodes are deployed over a uniform grid. We consider that there are five fixed sensors at known positions, and 43 other sensors with known positions for the training and test phases. Figure 5.3 shows the topology of the testbed. It is important to note that the average values over time of the RSSIs are

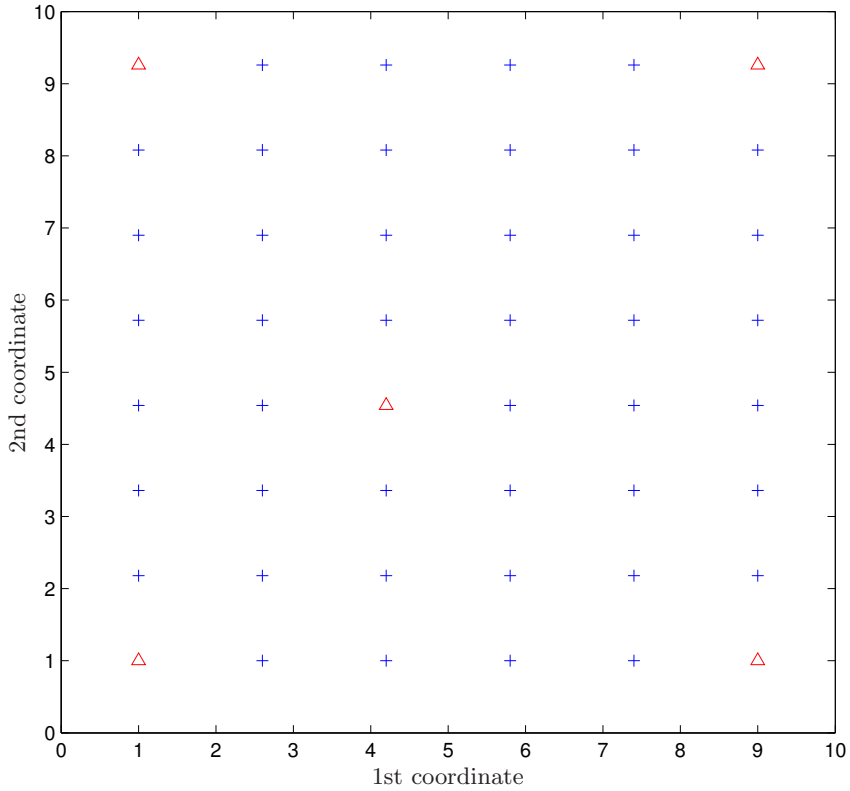


Figure 5.3: Topology of the testbed (real data), where Δ represents the sensors and + represents the training positions

used here. In fact, the RSSIs vary significantly with respect to time and movements, as one can see in Figure 5.4. These variations are known as short-term or multi-path fading. On the other hand, the local average of the signal varies slowly. These slow fluctuations depend mostly on environmental characteristics, and they are known as long-term fading. Therefore, it is more suitable to use the average values of the RSSIs than to use all the collected values [Neskovic et al., 2000].

Now the objective is to find five distance models, i.e., one model per sensor. Each model has a different set of training data to be used for the estimation of its parameters. As we already explained in Subsection 5.2.2, each model training set consists, on one hand, of the RSSIs of the signals exchanged between the considered fixed sensor and the 43 other sensors and, on the other hand, of the corresponding distances separating these sensors. The mean value of the training error (in meters) for the computed models is given in Table 5.5, along with the mean error on the training set for the physical log-distance propagation model used in [Medeisis and Kajackas, 2000; Patwari et al., 2005; Zanella and Bardella, 2014] and for the polynomial model of [Wang et al., 2009], for

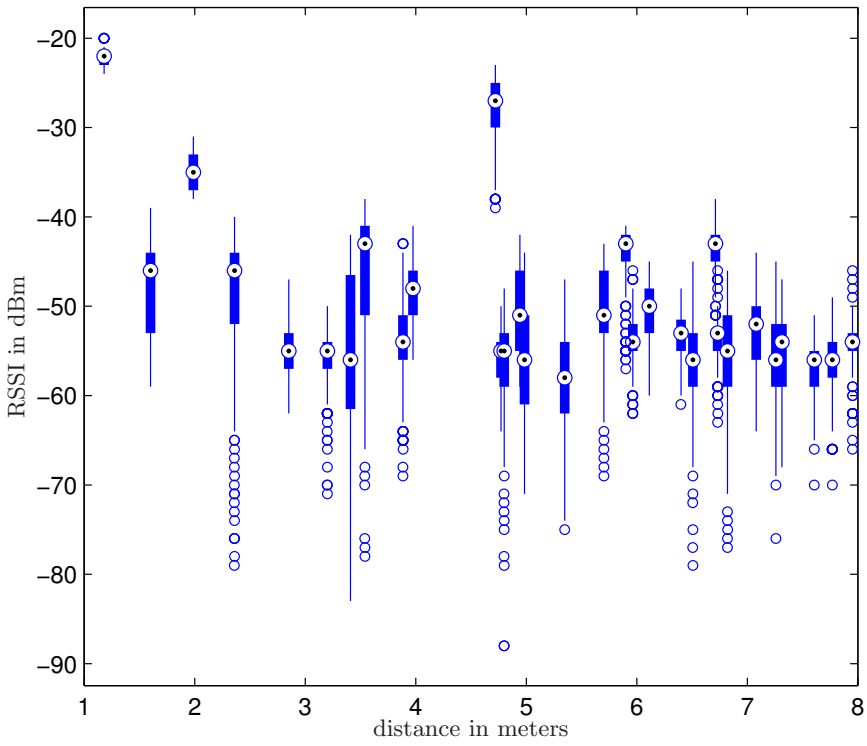


Figure 5.4: RSSIs measured at the training positions as a function of the distances separating these positions from a fixed sensor

several degrees q. We also use the leave-one-out (LOO) technique in order to evaluate the performance of the proposed model in the case of data that are not part of the training set. The LOO technique involves using a single observation from the collected data as the validation data, and the remaining observations as the training data. This is repeated 43 times, such that each observation is used once as the validation data. This technique is interesting because it allows us to compare the proposed models to the log-distance model, even though the set of collected data is not really large. Finally, the mean estimation error (in meters) is stored in Table 5.5 for the computed models. One can see from this table that the proposed models yield better results than the log-distance model and the polynomial for the different degrees q, when comparing the mean estimation error. Moreover, the two proposed models yield really close results.

Now, in a similar manner to Section 3.5 of Chapter 3, we generate additional reference positions in order to get a total of 100 reference positions. This is done, here as well, using a weighting function that relies on the distances between the existing points and the new ones. Then, for the test phase, we generate in the same way 100 random test positions. Table 5.6 shows the mean training and test errors (in meters) for the computed models. One can see that the estimation error decreases for all models; indeed, this is

Table 5.5: Distance estimation errors for the different models in the case of real data, where NPM, SPM, PM, PolyM denote respectively the non-parametric model, the semi-parametric one, the log-distance propagation one and the polynomial one

	PM	PolyM $q=2$	PolyM $q=3$	PolyM $q=4$	NPM	SPM
Mean training error	1.36	1.30	1.29	1.29	1.21	1.17
Mean LOO error	1.41	1.40	1.48	1.74	1.33	1.33

Table 5.6: Distance estimation errors for the different models in the case of real weighted data, where NPM, SPM, PM, PolyM denote respectively the non-parametric model, the semi-parametric one, the log-distance propagation one and the polynomial one

	PM	PolyM $q=2$	PolyM $q=3$	PolyM $q=4$	NPM	SPM
Mean training error	1.17	1.11	1.10	1.06	0.95	0.93
Mean test error	1.29	1.18	1.14	1.10	1.01	1.02

due to the weighting process, that smoothes the RSSI/distance function. Nevertheless, the two proposed models still outperform the other models in terms of accuracy.

5.7 Performance evaluation of the tracking methods

This section evaluates the performance of the two proposed tracking methods on simulated data. In the first subsection, we evaluate the performance of the proposed methods in terms of accuracy for fixed values of the noise standard deviations $\boldsymbol{\sigma_\gamma}$ and $\sigma_{\boldsymbol{\rho}}$. Next, in the second subsection, we study the influence of the noise standard deviations $\boldsymbol{\sigma_\gamma}$ and $\sigma_{\boldsymbol{\rho}}$ on the estimation error. Then, in the third subsection, we compare our results to ones obtained with the WKNN algorithm combined with a Kalman filter [Chan et al., 2009]; we also study the influence of the number of sensors N_s on the estimation error for the proposed methods and the method in [Chan et al., 2009]. Finally, we compare the accuracy of the proposed methods to the localization method proposed in Chapter 3 and to the tracking method proposed in Chapter 4 in both cases of fixed and moving sensors. In the following, we consider the setup of Figure 5.1, and the RSSI measures are generated using (5.6), with n_P set to 4 and ρ_0 set to 1 dBm. Now for the choice of the distance model, one can see from the results of Section 5.6 that both proposed models yield really close results in the case of simulated data. Consequently, we will use the non-parametric model in this section. As we already explained, the kernel parameters η_i and σ_i are chosen in such a way to minimize the error on the training set. As for the

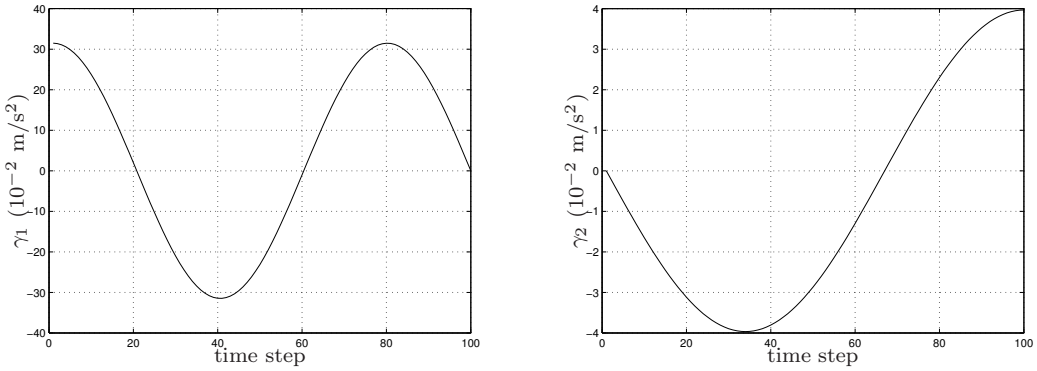


Figure 5.5: Acceleration signals of the target, with γ_1 on the left and γ_2 on the right being the first and the second acceleration coordinates respectively

application of the particle filter in all our simulations, the number of particles N_{PF} is set to 50.

5.7.1 Evaluation of the proposed methods

We consider a moving target in the defined area. The target accelerations are given in Figure 5.5, γ_1 and γ_2 being the first and the second acceleration coordinates respectively. By taking twice the primitive integrals of the accelerations, we compute the coordinates of the target. The obtained trajectory contains 100 points with $\Delta t = 1$ s, and is illustrated in Figure 5.6. We consider that noises are present in all scenarios, since a noiseless setup is not realistic in a practical environment. To this end, we take $\sigma_{\boldsymbol{\rho}}$ equal to 10% of the standard deviation of the RSSI measures, i.e., $\sigma_{\boldsymbol{\rho}} = 1.08$ dBm, and we take $\boldsymbol{\sigma_\gamma}$ equal to 5% of the standard deviation of the accelerations. Let the estimation error be evaluated by the root mean squared distance between the exact positions and the estimated ones. Figure 5.6 shows the estimated trajectories when using the two proposed tracking methods. The results are averaged over 50 Monte-Carlo simulations. The mean error obtained when using the Kalman filter of Section 5.4 is equal to 1.03 m. As for the mean error obtained with the particle filter of Section 5.5, it is equal to 0.68 m. One can see that both methods allow an accurate tracking of the target, with a better estimation error when considering the particle filter with this setup.

5.7.2 Influence of $\boldsymbol{\sigma_\gamma}$ and $\sigma_{\boldsymbol{\rho}}$

Now we study the influence of the noise standard deviations $\boldsymbol{\sigma_\gamma}$ and $\sigma_{\boldsymbol{\rho}}$ on the estimation error. First, we take a fixed value for $\sigma_{\boldsymbol{\rho}}$, equal to 10% of the standard deviation of the

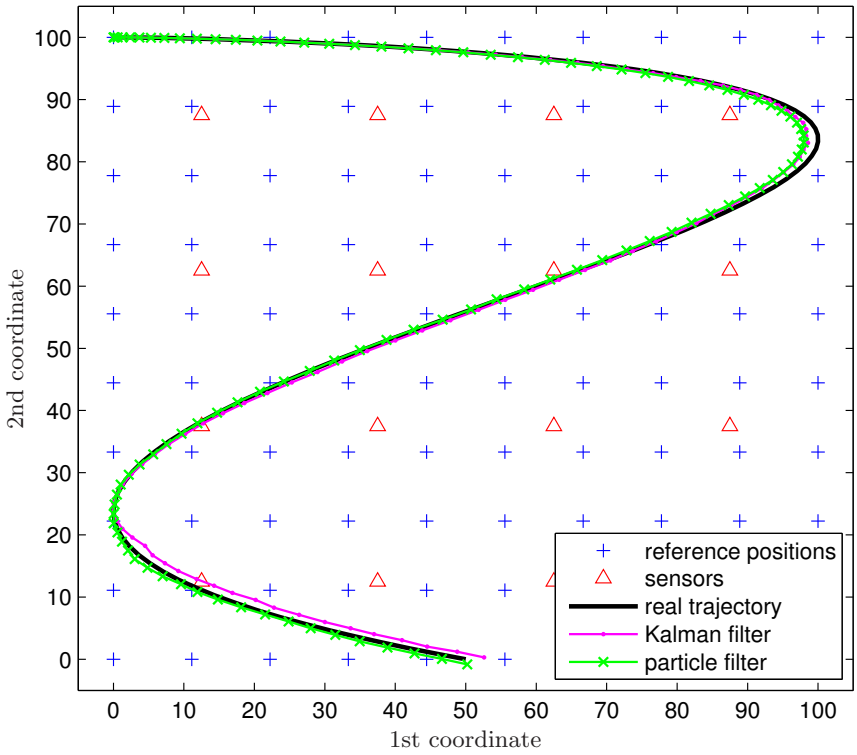


Figure 5.6: Estimation of the trajectory: simulated data

RSSI measures. Then, different percentages of the standard deviation of the acceleration are considered, ranging from 1% to 10%. The estimation errors are, here as well, averaged over 50 Monte-Carlo simulations. Figure 5.7 shows the influence of $\boldsymbol{\sigma_\gamma}$ on the estimation error. This figure shows that both filters have similar performances when the noise on the accelerations is small; however, with higher noise values, the particle filter outperforms the Kalman filter.

Finally, we take a fixed value for $\boldsymbol{\sigma_\gamma}$ equal to 5% of the standard deviation of the accelerations, with several percentages of the standard deviation of the RSSI measures, going from 0% to 50%. Figure 5.8 shows the influence of the variation of σ_ρ on the estimation error for both methods. One can see here as well that the particle filter yields better results than the Kalman filter. This is due to the distribution of the observation errors, assumed to be Gaussian in the Kalman filter. Indeed, small RSSI errors yield slightly varying observation errors. However, with higher RSSI errors, the noise distribution gets farther from a Gaussian one, and the particle filter performs better. Nevertheless, one can see that both methods have relatively small estimation errors, compared to the space dimensions.

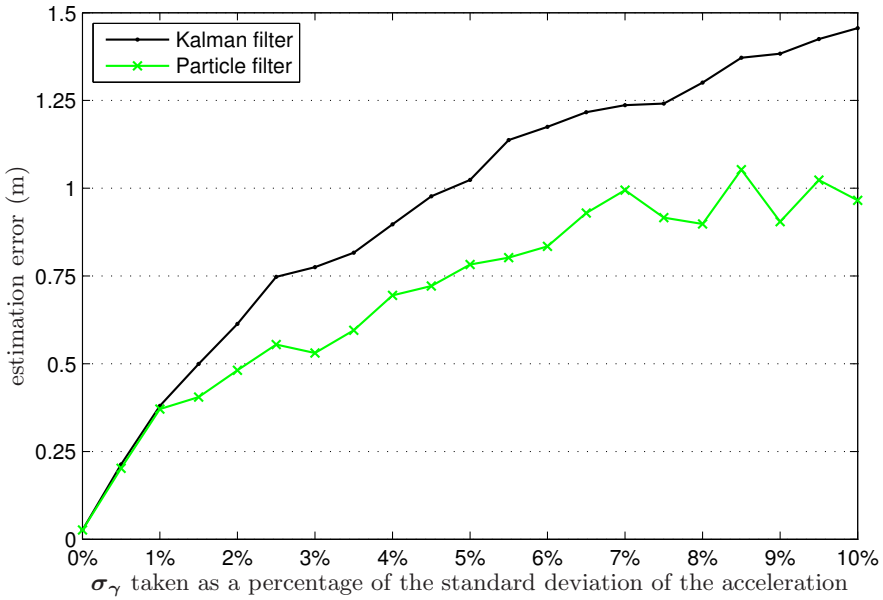


Figure 5.7: Estimation error as a function of the noise on the accelerations, with σ_ρ equal to 10% of the standard deviation of the RSSIs

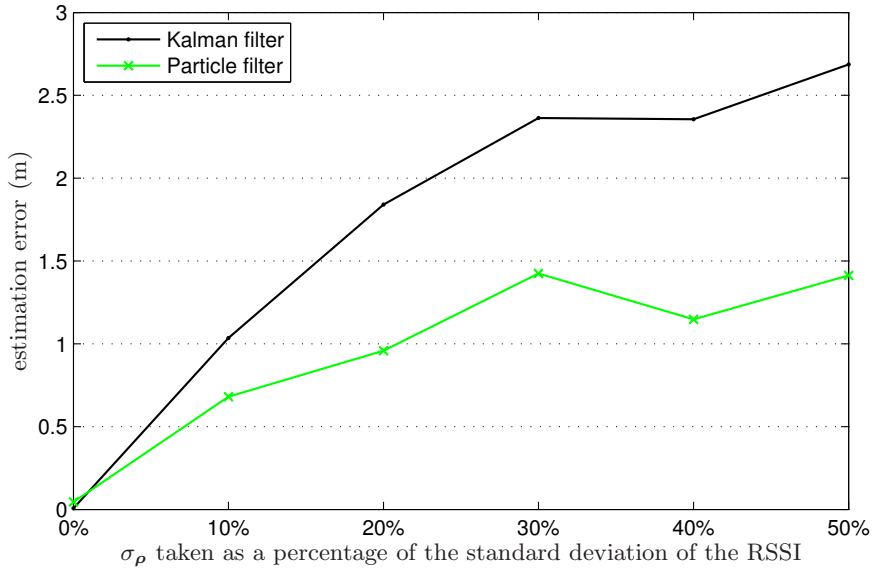


Figure 5.8: Estimation error as a function of the noise on the RSSI

Table 5.7: Estimation errors for the different tracking techniques

Tracking method	Mean estimation error
Proposed Kalman filter	**1.03**
Proposed particle filter	**0.68**
WKNN	4.57
WKNN + Kalman	2.15

5.7.3 Comparison to other techniques

The objective now is to first compare the proposed approach to the method in [Chan et al., 2009; Liu et al., 2011]. We consider the same setup as the one in Subsection 5.7.1, and the same values for $\boldsymbol{\sigma_\gamma}$ and $\sigma_{\boldsymbol{\rho}}$. Then, taking also the same setup, we compare the tracking method proposed in this chapter to the methods proposed in Chapter 3 and Chapter 4, in both cases of fixed and moving sensors.

As we already explained in Subsection 4.4.5 on page 93, the method in [Chan et al., 2009] consists in estimating the position using the weighted K-nearest neighbor algorithm, then applying the Kalman filter with a second-order state-space model to enhance the estimation. We use the same weight as in Chapter 4, given in (4.18), with the number of neighbors set to 8, as in [Chan et al., 2009]. The estimation errors (in meters) obtained when using this method are computed 50 times, and the mean estimation error is stored in Table 5.7, where σ_{MSE} is the standard deviation of the mean estimation error. This table also shows the mean estimation error of this algorithm without using the acceleration information (WKNN). One can see that the two proposed methods outperform the WKNN-based methods in terms of accuracy.

Next, several percentages of the standard deviation of the RSSI measures are taken using the methods in [Chan et al., 2009]. The value of $\boldsymbol{\sigma_\gamma}$ is taken equal to 5% of the standard deviation of the accelerations as in the scenario of Figure 5.8. The estimation error obtained in this case is shown in Figure 5.9. This figure shows that the two proposed methods outperform the other methods for all values of $\sigma_{\boldsymbol{\rho}}$, compared to the results obtained in Figure 5.8.

We now compare these methods for several values of the number of sensors N_s, for a fixed number of reference positions $N_p = 100$. The value of $\sigma_{\boldsymbol{\rho}}$ is taken equal to 10% of the standard deviation of the RSSI measures, and $\boldsymbol{\sigma_\gamma}$ is equal to 5% of the standard deviation of the accelerations. Figure 5.10 shows the estimation error as a function of N_s. Note that here the sensors are assumed fixed, since this assumption is essential for a good performance of the WKNN-based methods. One can see that the two proposed methods yield the smallest errors for all values of the number of sensors; moreover, at

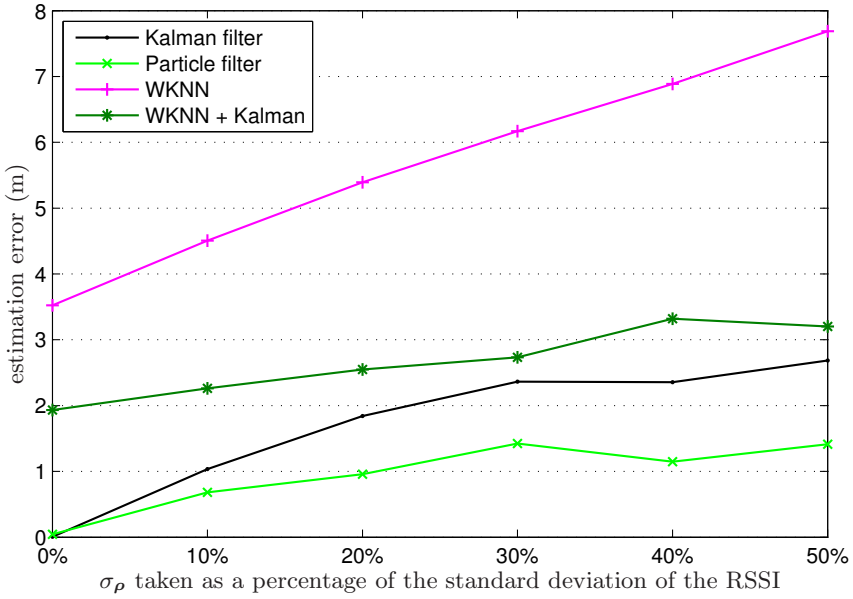


Figure 5.9: Estimation error as a function of the noise on the RSSI, using different methods

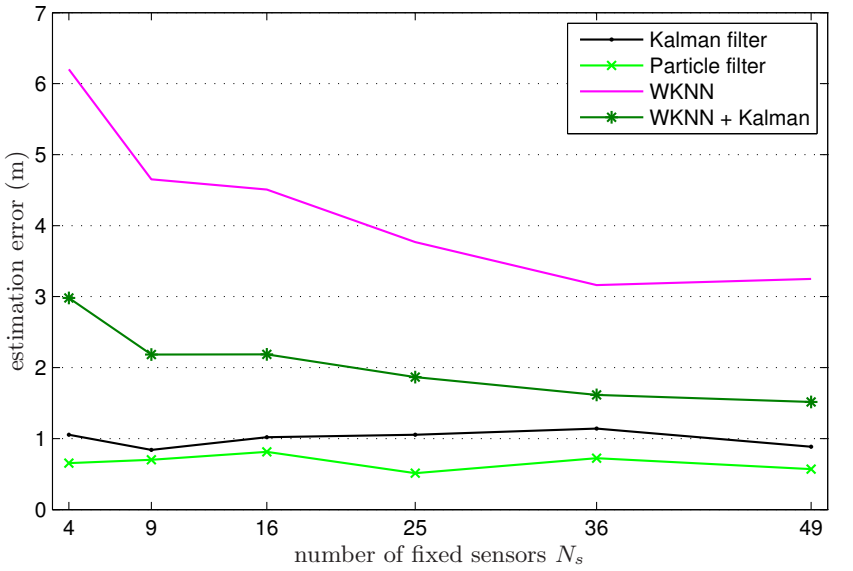


Figure 5.10: Estimation error as a function of the number of fixed sensors N_s

$N_s = 4$, the other methods do not give such accurate results. Another important thing to notice is that the proposed methods are more robust to the changes in the number of sensors, and they present less variations than the other methods.

Finally, let us consider the localization method presented in Chapter 3 and the tracking method presented in Chapter 4, both using the kernel ridge regression learning algorithm.

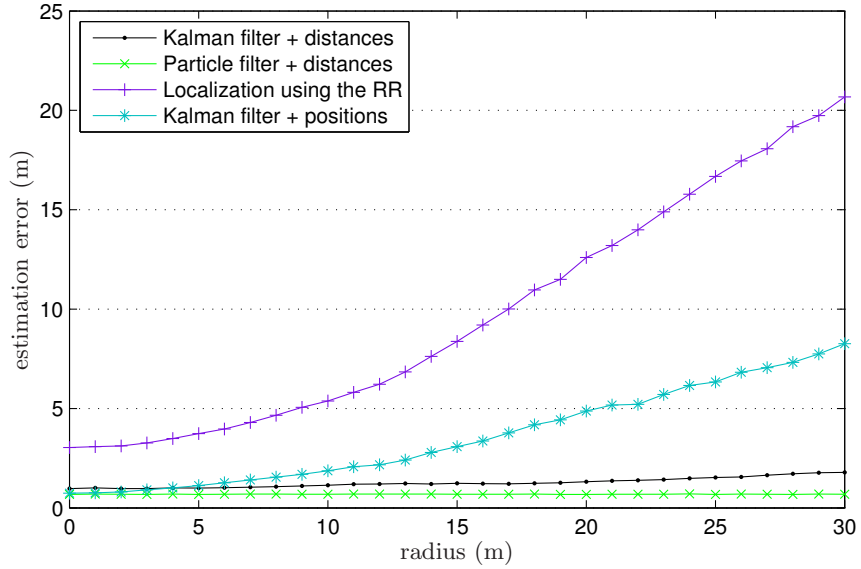


Figure 5.11: Estimation error as a function of the radius defining the movement of the sensors

We consider the scenario described at the beginning of this subsection, with the same value for the noises standard deviations, and we estimate the positions of the target. To compare the methods, consider that the $N_s = 16$ sensors are moving in the surveillance area, with circular movements. The trajectory of Figure 5.6 is estimated using all proposed methods, for different radii ranging from 0 to 30 m and for random angles going from 0 to 2π radians. The results are then averaged over 50 Monte-Carlo simulations. Figure 5.11 shows the mean estimation error as a function of the radius for the tracking methods. One can see that all acceleration-based methods yield close results at the beginning, when the radius is equal to zero, that is when the sensors remain fixed. This condition is essential to obtain optimal results when using the previously proposed methods, since the sensors' positions are included in the learning process. The advantage of the proposed methods in this chapter is that we find the RSSI/distance relationship, and thus find directly the distance from the RSSI measures. When the sensors change their positions, this relationship does not change, thus we do not need to reconfigure the model, i.e. a new training phase is not needed. One can see from Figure 5.11 that when the value of the radius increases, the estimation error also increases for both methods of Chapter 3 and Chapter 4, while the estimation error almost remains the same for the two proposed methods in this chapter. This result proves that the new proposed methods are more robust to changes in the initial configuration of the surveillance area than the previous methods.

5.8 Conclusion

In this chapter, we proposed two original regression models that relate the received signal strength indicators to the distances separating sensors in the network. Then, we solved the tracking problem using the estimated distances and two new methods that take into account the target's motion, using either the Kalman filter or the particle filter. We provided a fully comprehensive study of the proposed distance models and their performances. Simulation results show that our models yield accurate distance estimation. Results also show that our tracking methods allow accurate position estimation, and are proved to be robust in the case of noisy data. Moreover, when the sensors are no longer fixed, both proposed tracking methods outperform the methods proposed in the previous chapters.

Chapter 6

Parameter Estimation of Gas Diffusion Sources

Contents

In this chapter, we introduce an original clusterized framework for the detection and estimation of the parameters of multiple gas sources in wireless sensor networks. Each cluster head processes the concentration measures collected regularly by the sensors within its cluster in order to detect gas diffusions, by means of a kernel-based detector. Once an alert is raised in a cluster, the collected concentrations are processed locally in order to estimate the gas diffusion parameters, such as the source's location and the release rate. First estimates of the parameters are obtained using a kernel-based model that takes as input the collected concentrations. Next, an enhanced estimate of the source's location is provided using the estimated parameters and the measures concentrations with a second kernel-based model. We then evaluate the performance of the proposed method in the case of a single source, as well as for multiple sources. Moreover, a comparison to state-of-the-art techniques is also given.

6.1 Introduction

In the previous chapters, we provided several localization and tracking techniques for wireless sensor networks. Once their positions are known, the sensors are able to collect physical measurements in order to monitor the environment where they are deployed. WSNs are being widely used for the detection and estimation of pollutions caused by chemical, biological, radiological or nuclear (CBRN) attacks, in particular for extreme environmental conditions. In the following, we focus particularly on the problem of detecting and estimating gas diffusions. In fact, as we already explained in Chapter 1, gas releases might occur either accidentally, such as with the Union Carbide release in Bhopal, India in 1984 [Kalelkar, 1988], or deliberately, such as with the Sarin gas attack on Tokyo, Japan in 1995 [Tu, 1999]. Considering the potentially catastrophic consequences of such diffusions on environment and on human life, it is important to detect the gas diffusion and estimate its parameters, including the source location and the release rate. These computations require the collection of gas concentration measurements from the area, which necessitates specially trained people with appropriate protective equipments. Alternatively, WSNs have proven to be very useful in such extreme scenarios, where human intervention is risky and expensive. Typically, sensors are deployed in the area to be monitored and are regularly and continuously collecting measurements from the area. The collected information is then processed in order to estimate the source parameters.

Several methods have been proposed in the literature for the estimation of the source parameters. For instance, Kathirgamanathan et al. [2002] developed an inverse model

for inferring the parameters of an instantaneous point source from gas concentration measurements. The method solves a non-linear least squares estimation problem. Christopoulos and Roumeliotis [2005] determined the source parameters using mobile robots that collect concentration measurements. The study was focused on the selection of the sequence of locations where each robot should be moved in order to obtain accurate real-time estimates. Another method was proposed in [Delmaire and Roussel, 2012], where the problem of pollutant source localization and flow estimation is addressed in a one-dimensional context, using a single remote sensor. The pollutant is assumed to be generated by one out of several possible sources, and the task is viewed as a conditional deconvolution which requires a priori knowledge. In the end, a joint estimation decision is derived in a Bayesian framework. Ickowicz et al. [2012] also proposed a method for the prediction and estimation of the concentration of pollutants in complex environments. The authors take benefit of a semi-parametric formulation of the problem and Gaussian processes to model the interactions between position and time given some observations, and then obtain a confidence region of the position of the source and the release time. This method and all the aforementioned ones demand an extensive understanding of the WSN topology and require a priori knowledge of the underlying diffusion process. To overcome these issues, one can clearly take advantage of the flexibility of kernel-based methods for non-parametric estimation as we will show in the following.

In this chapter, we propose a new clusterized framework for the detection and estimation of gas diffusions from multiple sources in wireless sensor networks. Sensors are deployed in the region of interest and collect concentration measurements at regular and short sampling intervals. The region is divided into clusters, each having its own cluster head. Consequently, information processing is done locally. The proposed framework operates in two phases: the detection phase and the estimation phase. In the first one, a kernel-based detector is defined using the support vector data description (SVDD) [Tax and Duin, 2004]. The detector is used by each cluster head in order to identify any gas diffusion in the specified cluster using the collected concentration measurements. When an anomaly is spotted, the concentrations that first triggered the alert are treated in the estimation phase to determine the gas release parameters, such as the source location and the release rate. A first estimate of the parameters is obtained using a non-linear model, which is also defined within the framework of kernel methods. Then, part of the estimated parameters are processed as internal feedback, along with the measured concentrations, in order to provide a more accurate estimate of the source location. Simulations show that the proposed estimation method yields accurate results in the case of a single source, as well as in the case of multiple sources.

6.2 Problem Statement

In this chapter, we consider a clusterized approach because of the nature of the considered application. Indeed, a large number of sensors needs to be deployed in order to monitor a gas diffusion in a large region of interest. Therefore, the amount of information that needs to be processed is high, and it becomes then more suitable to partition the area into several smaller areas, and then assign a cluster head to each one.

6.2.1 Network configuration

Consider N sensors deployed in the area to be monitored, at fixed locations (x_n, y_n, z_n), $n \in \{1, \ldots, N\}$. These sensors measure, at each time t, the gas concentrations at their locations. The area is partitioned into Z distinct clusters, as often recommended for WSNs, in order to make the computations more appropriate for WSNs and the method more robust to transmission impairments and network failures. Each cluster is managed by a cluster head, that is a smart central processing unit (CPU), responsible of handling (gathering and synchronizing) data, performing calculations, and exchanging information with the sensors. Note that this device could also be one of the sensors of the network, and it can be placed anywhere in its specified cluster. In addition, all cluster heads can communicate with each other, and clusters could have any shape or dimension. Without loss of generality, we consider here that clusters are rectangular, with the cluster heads located at their centers, as illustrated in Figure 6.1. The variables that will be used in the following are listed in Table 6.1, along with their respective sizes.

6.2.2 Description of the proposed approach

In this subsection, we give a brief description of the proposed approach. The objective is to monitor a region of interest in order to detect one or multiple gas diffusions using wireless sensors networks, and then estimate the gas source parameters, such as the source location and the release rate. Accordingly, the approach consists of two phases: the detection phase and the estimation phase. At all times, the deployed sensors are constantly collecting concentration measurements at short sampling intervals. Note that sensors that belong to the same cluster only send their collected data to their cluster head. Each cluster head uses then a kernel-based detector in order to identify any gas diffusion in its cluster. More details about the detection phase will be given in Section 6.3, where we also define the kernel-based detector, using the support vector data description.

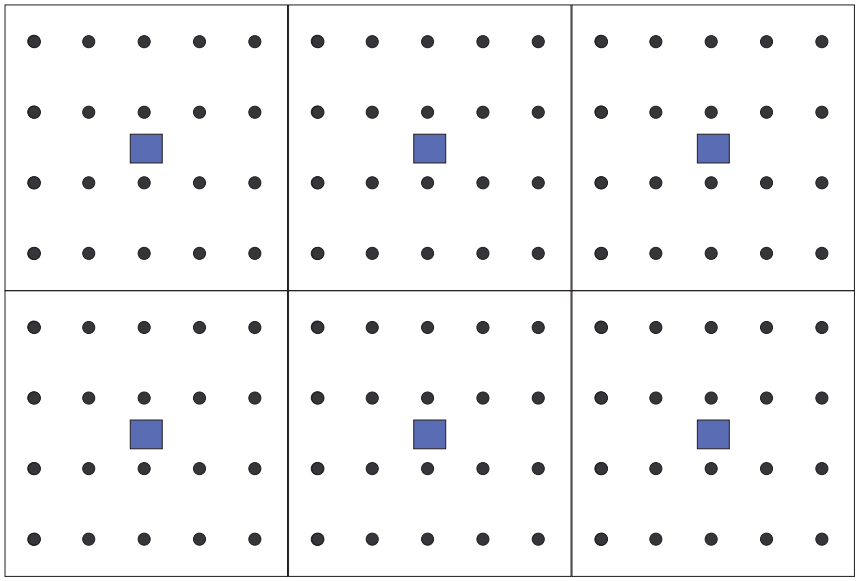


Figure 6.1: Region topology, where ■ represents the cluster heads and • represents the deployed sensors

The gas concentrations that first triggered the alert are now treated to estimate the gas diffusion parameters. Concentration measures from adjacent clusters in alert are merged together in order to estimate the number of possible sources. Note that not all the concentrations will be processed in the estimation phase, since some are irrelevant for parameters estimation. Details about the fusion of clusters and the selection process will be given in Section 6.4. For the estimation of the source's parameters, we use a kernel-based model, that takes as input the collected concentrations. Next, using these measured concentrations and part of the estimated parameters as internal feedback, we provide an enhanced estimate of the source location. The definition of the two models will be provided in Section 6.4 as well. Figure 6.2 illustrates the scheme of the proposed approach. This scheme shows the different checkpoints of the proposed approach, starting with the acquisition of the concentrations, continuing to the detection phase, followed by a fusion of the information of the concerned clusters with a selection of the relevant concentrations, and finishing with the estimation of the source's parameters.

6.2.3 The advection-diffusion model

In the following, we propose a generalized version of the advection-diffusion model described in [Kathirgamanathan et al., 2002]. To this end, consider an instantaneous gas release of a mass Q, assumed to occur at time t_0 and at location (x_0, y_0, z_0). A wind with mean velocity $\boldsymbol{U} = (u_X, u_Y, 0)$ spreads the gas particle in the region. The mass concentration C of the released gas, at an arbitrary location (x, y, z) and at time t, is

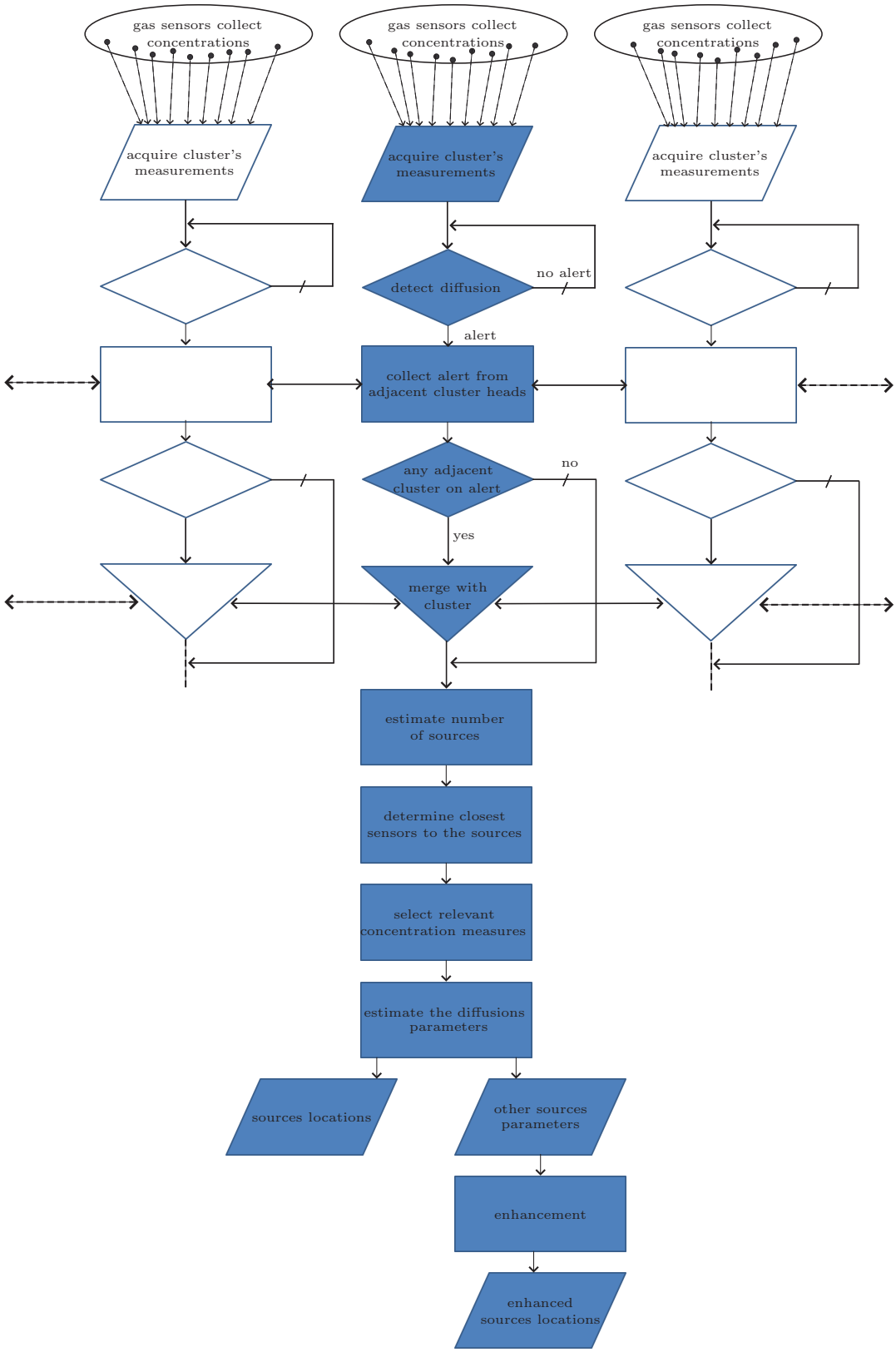


Figure 6.2: Scheme of the proposed method

Table 6.1: List of the used variables in Chapter 6, along with their respective sizes

Notation	Variable	Size
N	number of deployed sensors	1
N_s	number of sources	1
N_{det}	size of the training set for the detection	1
N_{reg}	size of the training set for the regression	1
x_n, y_n, z_n	coordinates of sensor n	1
C	concentration measure	1
K_X, K_Y, K_Z	eddy diffusivities	1
Q	release rate	1
t_0	time of the gas release	1
x_0, y_0, z_0	coordinates of the source	1
Z	number of clusters	1
$N^{(z)}$	number of sensors in cluster z	1
α_i, υ_i	Lagrangian multipliers	1
$\boldsymbol{\psi}_B$	enhancement model	1×2
$\boldsymbol{U} = (u_X, u_Y, 0)$	wind velocity	1×3
$\boldsymbol{\psi}_A$	estimation model	1×5
$\boldsymbol{\theta}_\ell$, $\boldsymbol{\Theta}_{\ell,*}$	source parameters	1×5
$\boldsymbol{c}^{(s)}$, $\boldsymbol{c}_\ell$	local concentration vector	$V \times 1$
$\boldsymbol{W}_\ell$, $\boldsymbol{W}$	input of $\boldsymbol{\psi}_B$	$(V+3) \times 1$
$\boldsymbol{C}^{(z)}$, $\boldsymbol{C}$	concentration vector collected from cluster z	$N^{(z)} \times 1$
$\boldsymbol{x}_0$	sources' first coordinates	$N_{\text{reg}} \times 1$
$\boldsymbol{y}_0$	sources' second coordinates	$N_{\text{reg}} \times 1$
$\boldsymbol{\gamma}$	matrix of learning coefficients	$N_{\text{reg}} \times 2$
$\boldsymbol{\Theta}$	source parameters matrix	$N_{\text{reg}} \times 5$
$\boldsymbol{\beta}$	matrix of learning coefficients	$N_{\text{reg}} \times 5$
$\boldsymbol{K}_A$, $\boldsymbol{K}_B$	Gram matrix	$N_{\text{reg}} \times N_{\text{reg}}$

governed by the equation of mass conservation given by the following:

$$\frac{\partial C}{\partial t} = -\nabla \boldsymbol{q}, \tag{6.1}$$

where ∇ is the gradient operator, and $\boldsymbol{q}$ is the gas mass flow per unit area. The flow $\boldsymbol{q}$ is given by:

$$\boldsymbol{q} = C\ \boldsymbol{U} - \begin{bmatrix} K_X & 0 & 0 \\ 0 & K_Y & 0 \\ 0 & 0 & K_Z \end{bmatrix} \otimes \nabla C, \tag{6.2}$$

where $C\ \boldsymbol{U}$ is the mean mass advection by the wind, $\otimes$ is the tensor product, and K_X, K_Y, K_Z are eddy diffusivities in the X, Y and Z directions respectively. Equation (6.2) can then be written as follows:

$$\boldsymbol{q} = \left(C\ u_X - K_X \frac{\partial C}{\partial X}, C\ u_Y - K_Y \frac{\partial C}{\partial Y}, -K_Z \frac{\partial C}{\partial Y} \right). \tag{6.3}$$

By substituting (6.3) into (6.1), we get an equation that can be solved subject to two boundary conditions. The first condition results from the fact that the concentration is zero at infinity in all spatial directions, and the second condition is that the gas is not absorbed by the ground. The velocity of the wind $\boldsymbol{U}$, as well as the eddy diffusivities K_X, K_Y, K_Z, are assumed to be constant. Following these assumptions, we get the following solution:

$$C(x,y,z,t) = \frac{Q}{8\,\pi^{\frac{3}{2}}\,(K_X K_Y K_Z)^{\frac{1}{2}}\,\Delta t^{\frac{3}{2}}} \times \exp\left(-\frac{(\Delta x - u_X\Delta t)^2}{4K_X\Delta t} - \frac{(\Delta y - u_Y\Delta t)^2}{4K_Y\Delta t}\right) \times \left(\exp\left(-\frac{\Delta z^2}{4K_Z\Delta t}\right) + \exp\left(-\frac{\Delta z'^2}{4K_Z\Delta t}\right)\right), \tag{6.4}$$

where $\Delta x = x - x_0$, $\Delta y = y - y_0$, $\Delta z = z - z_0$, $\Delta z' = z + z_0$ and $\Delta t = t - t_0$.

For the sake of simplicity, we assume in the following that all measurements are taken at ground level. We also suppose that the gas release occurs at ground level, which means that the gas source is at location $(x_0, y_0, z_0) = (x_0, y_0, 0)$. Therefore, using (6.4), the concentration measured at a location $(x, y, 0)$ is given by the following:

$$C(x,y,0,t) = \frac{Q}{4\,\pi^{\frac{3}{2}}\,(K_X K_Y K_Z)^{\frac{1}{2}}\,\Delta t^{\frac{3}{2}}} \times \exp\left(-\frac{(\Delta x - u_X\Delta t)^2}{4K_X\Delta t} - \frac{(\Delta y - u_Y\Delta t)^2}{4K_Y\Delta t}\right). \tag{6.5}$$

Note that the wind velocity $\boldsymbol{U}$ could be provided by an anemometer, and is therefore treated as a known constant [Christopoulos and Roumeliotis, 2005]. The other parameters of the model are unknown and need to be estimated when a gas diffusion occurs over the area under scrutiny. Hence, having gas concentrations measured over a certain area using a WSN, we aim at detecting the gas diffusion when it occurs, as we will show in Section 6.3. Then, in Section 6.4, the objective is to estimate the location $(x_0, y_0, 0)$ of the source, the released gas mass Q, and the eddy diffusivities K_X, K_Y and K_Z, with K_X and K_Y assumed to be equal [Kathirgamanathan et al., 2002].

6.3 The Detection Phase

In this section, we first give a brief description of the network setup and of the detection phase. Then, we define a classifier that is capable of optimally separating the data into normal and abnormal data, where the abnormality denotes the occurrence of a gas diffusion. Several classification methods can be used to define the classifier, most of them are parametric, thus requiring the physical knowledge of the underlying system. In this chapter, we only focus on non-parametric methods, with the support vector data description.

6.3.1 Description of the detection phase

Now let $(C(x_1, y_1, z_1, t) \dots C(x_N, y_N, z_N, t))^\top$ be the $N \times 1$ vector of the gas concentrations measured at time t by all N sensors. These concentrations are assumed to follow the advection-diffusion model, detailed in Subsection 6.2.3. As we already explained, in order to thoroughly monitor the area of interest, the concentrations are measured at regular and short sampling intervals by the deployed sensors. The network is configured as in Figure 6.1, that is Z distinct clusters, each one managed by a cluster head. At every time t, each cluster head receives the measured concentrations from the sensors in its cluster. Let $N^{(z)}$ denote the number of sensors in cluster z, $z \in \{1, ..., Z\}$, and let $\boldsymbol{C}^{(z)}(t)$ denote the concentration vector of size $N^{(z)} \times 1$ collected by the cluster head of cluster z at time t. The received data are then processed instantly to detect whether a gas diffusion has occurred or not.

Assume now that an instantaneous gas release of a mass Q occurs at time t_0 at location $(x_0, y_0, z_0) = (x_0, y_0, 0)$, all these parameters being unknown. The gas particles are then spread by a wind with a mean known velocity $\boldsymbol{U} = (u_X, u_Y, 0)$. Since a gas is diffusing in the area, an alarm should be triggered based on the vector of measured concentrations. To this end, we need to develop a detector, capable of determining whether the concentrations are normal or not; in other words, it can detect whether or not there is one or more gas diffusions. Therefore, the objective is to define a data-driven detector, given samples that are originated from a single class, i.e., the normal data, but are possibly contaminated with a small number of outliers, i.e., the abnormal data. To this end, we investigate the one-class classification framework. Indeed, the one-class support vector machines (one-class SVM) [Schölkopf et al., 2001b] and the support vector data description (SVDD) are widely used to address the problem of anomaly detection.

Let ϕ denote a function that maps the data from the input space into a feature space $\mathcal{H}$. By input space we mean the space of concentration vectors ($\mathbb{R}^{N^{(z)}}$) that are collected from the clusters. Consider a kernel $\kappa : \mathbb{R}^{N^{(z)}} \times \mathbb{R}^{N^{(z)}} \to \mathbb{R}$, with $\mathcal{H}$ its reproducing kernel Hilbert space (RKHS) with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. We then have $\kappa(\boldsymbol{C}_i, \boldsymbol{C}_j) = \langle \phi(\boldsymbol{C}_i), \phi(\boldsymbol{C}_j) \rangle_{\mathcal{H}}$. Now consider a training set of concentrations $\boldsymbol{C}_i$, $i \in \{1, ..., N_{\text{det}}\}$, N_{det} being the size of the training set for the detection phase. The time t is dropped because the concentrations are collected at any time, since normal and abnormal concentrations are considered in the training set. Indeed, the presence of outliers is authorized in the training set to allow a better design of the detector. In the following, we define the detector using the SVDD, which consists in estimating the hypersphere with minimum radius that encloses most of the data $\phi(\boldsymbol{C}_i)$ in the feature space $\mathcal{H}$. Note that the

optimization problem in the case of the SVDD is essentially similar to the one-class SVM. Moreover, they are equivalent when the Gaussian kernel is used.

6.3.2 Definition of the detector using support vector data description

Support vector data description (SVDD) essentially fits the smallest possible sphere around the given data of the training set, allowing a small fraction of the samples to be excluded as outliers. Therefore, a spherically shaped decision boundary with minimum radius is computed to enclose most of the training data. Data lying outside this decision boundary are considered as abnormal, i.e., outliers.

Now let $\boldsymbol{a}$ be the center of the hypersphere, and $R > 0$ its radius. The slack variables, $\xi_i \geq 0$, are introduced here to allow the presence of outliers in the training data, namely samples not inside the hypersphere. All these variables are obtained by minimizing a cost function that balances the volume of the hypersphere against the penalty associated with outliers. Note that minimizing the hypersphere volume is equivalent to minimizing R^2. Therefore, we get the following constrained optimization problem:

$$\min_{\boldsymbol{a},R,\xi_i} R^2 + \frac{1}{\nu N_{\text{det}}} \sum_{i=1}^{N_{\text{det}}} \xi_i \tag{6.6}$$

subject to

$$\|\phi(\boldsymbol{C}_i) - \boldsymbol{a}\|_{\mathcal{H}}^2 \leq R^2 + \xi_i \quad \text{and} \quad \xi_i \geq 0 \;\; \forall i = 1, ..., N_{\text{det}}.$$

The quantity ν is a predefined parameter that regulates the tradeoff between the volume of the hypersphere and the number of outliers.

The Lagrangian L of the above constrained optimization problem is then given as follows:

$$L(R, \boldsymbol{a}, \alpha_i, \upsilon_i, \xi_i) = R^2 + \frac{1}{\nu N_{\text{det}}} \sum_{i=1}^{N_{\text{det}}} \xi_i - \sum_{i=1}^{N_{\text{det}}} \alpha_i \left(R^2 + \xi_i - \|\phi(\boldsymbol{C}_i) - \boldsymbol{a}\|_{\mathcal{H}}^2\right) - \sum_{i=1}^{N_{\text{det}}} \upsilon_i \xi_i, \tag{6.7}$$

where the α_i and the υ_i are the Lagrangian multipliers. Their values depend on whether the constraint $\|\phi(\boldsymbol{C}_i) - \boldsymbol{a}\|_{\mathcal{H}}^2 \leq R^2 + \xi_i$ is satisfied by the corresponding sample $\boldsymbol{C}_i$ or not. Therefore, three cases can be encountered:

$$\begin{aligned}
\|\phi(\boldsymbol{C}_i) - \boldsymbol{a}\|_{\mathcal{H}}^2 < R^2 \quad &\Longleftrightarrow \quad \alpha_i = 0, \quad &\upsilon_i = 0, \\
\|\phi(\boldsymbol{C}_i) - \boldsymbol{a}\|_{\mathcal{H}}^2 = R^2 \quad &\Longleftrightarrow \quad 0 < \alpha_i < \frac{1}{\nu N_{\text{det}}}, \quad &\upsilon_i = 0, \\
\|\phi(\boldsymbol{C}_i) - \boldsymbol{a}\|_{\mathcal{H}}^2 > R^2 \quad &\Longleftrightarrow \quad \alpha_i = \frac{1}{\nu N_{\text{det}}}, \quad &\upsilon_i > 0.
\end{aligned}$$

Now by taking the partial derivatives of L with respect to R, $\boldsymbol{a}$ and ξ_i, we get the following relations:

$$\begin{aligned}
\frac{\partial L}{\partial R} = 0 \quad &\longrightarrow \quad \sum_{i=1}^{N_{\text{det}}} \alpha_i = 1, \\
\frac{\partial L}{\partial \boldsymbol{a}} = 0 \quad &\longrightarrow \quad \boldsymbol{a} = \sum_{i=1}^{N_{\text{det}}} \alpha_i \phi(\boldsymbol{C}_i), \\
\frac{\partial L}{\partial \xi_i} = 0 \quad &\longrightarrow \quad 0 \leq \alpha_i \leq \frac{1}{\nu N_{\text{det}}}.
\end{aligned}$$

Incorporating these relations into the Lagrangian L gives us the following objective functional to be maximized with respect to α_i:

$$\sum_{i=1}^{N_{\text{det}}} \alpha_i \kappa(\boldsymbol{C}_i, \boldsymbol{C}_i) - \sum_{i=1}^{N_{\text{det}}} \sum_{j=1}^{N_{\text{det}}} \alpha_i \alpha_j \kappa(\boldsymbol{C}_i, \boldsymbol{C}_j), \tag{6.8}$$

subject to

$$\sum_{i=1}^{N_{\text{det}}} \alpha_i = 1 \quad \text{and} \quad 0 \leq \alpha_i \leq \frac{1}{\nu N_{\text{det}}}.$$

This is a quadratic programming problem, that can be solved using any off-the-shelf optimization technique, as in Chapter 3.

As for the radius of the optimal hypersphere, it is, by definition, the distance from the center $\boldsymbol{a}$ to any sample $\phi(\boldsymbol{C}_k)$ on the boundary. Therefore, the radius is given by the following:

$$\begin{aligned}
R^2 &= \|\phi(\boldsymbol{C}_k) - \boldsymbol{a}\|_{\mathcal{H}}^2 \\
&= \langle \phi(\boldsymbol{C}_k) - \sum_{i=1}^{N_{\text{det}}} \alpha_i \phi(\boldsymbol{C}_i), \phi(\boldsymbol{C}_k) - \sum_{i=1}^{N_{\text{det}}} \alpha_i \phi(\boldsymbol{C}_i) \rangle \\
&= \kappa(\boldsymbol{C}_k, \boldsymbol{C}_k) - 2 \sum_{i=1}^{N_{\text{det}}} \alpha_i \kappa(\boldsymbol{C}_k, \boldsymbol{C}_i) + \sum_{i=1}^{N_{\text{det}}} \sum_{j=1}^{N_{\text{det}}} \alpha_i \alpha_j \kappa(\boldsymbol{C}_i, \boldsymbol{C}_j).
\end{aligned} \tag{6.9}$$

Now in order to evaluate a new concentration vector $\boldsymbol{C}^{(z)}(t_d)$ at time t_d, the decision rule is obtained by evaluating the squared distance between the center $\boldsymbol{a}$ and $\phi(\boldsymbol{C}^{(z)}(t_d))$, namely:

$$\|\phi(\boldsymbol{C}^{(z)}(t_d)) - \boldsymbol{a}\|_{\mathcal{H}}^2.$$

Note that to compute this distance, one does not need the exact expressions of ϕ and $\boldsymbol{a}$. Instead, by developing the distance expression as in (6.9), one gets inner products of ϕ functions, which are evaluated by kernels. The new concentration $\boldsymbol{C}^{(z)}(t_d)$ is considered

as normal if the distance calculated is smaller than the radius, i.e., $\|\phi(\boldsymbol{C}^{(z)}(t_d)) - \boldsymbol{a}\|_{\mathcal{H}} \leq R$. Otherwise, an alert is triggered in the considered cluster.

Finally, note that the proposed algorithm can detect any abnormal concentration, which can be generated by one or multiple sources. It is worth noting here that by dividing the whole area into several clusters, the computations are processed in parallel at all the cluster heads. Moreover, the dimension of the considered data in the classifier for training and detection is limited to $N^{(z)}$, instead of taking all the sensors concentrations at once. This way, the network could be as large as needed without increasing the computation complexity.

6.4 The Estimation Phase

Now that our detector is defined, we are able to detect a gas diffusion in a specified cluster. Once an alert is triggered in some cluster z, we proceed to the processing of the concentration vector $\boldsymbol{C}^{(z)}(t)$ acquired by the sensors in the specified cluster. In this section, we describe the estimation phase and emphasize on the selection of the concentrations to be used in the source parameter estimation. Then, we define both estimation and enhancement models as illustrated in Figure 6.2.

6.4.1 Description of the estimation phase

The main difficulty resides in merging the clusters in alert in a way to discern all the sources. One can take all the clusters in alert together; however, this approach does not scale up to large networks with many sources. To overcome this drawback, we propose to merge together adjacent clusters that are in alert; by adjacent, we mean the clusters that share a common boundary. Each cluster in alert will form a group of its own if it is not adjacent to others that are also in alert. To illustrate this, see Figure 6.3, where six sources are diffusing gas in the region of interest.

At the end of this fusion, several groups with different sizes are obtained, each covering one or multiple sources. The concentration vectors recorded by the sensors of a group are then communicated to a cluster head of the group, e.g., the one having the highest computation capabilities, and are concatenated in a single "group" vector. Now, in order to detect multiple gas diffusions per group, the proposed method uses the concentration group vectors to compute local maxima, leading ideally to the concentrations of the closest sensors to the gas sources. Let N_s be the total number of local maxima detected by all groups. This means that N_s diffusions have occurred at time t, or at different times,

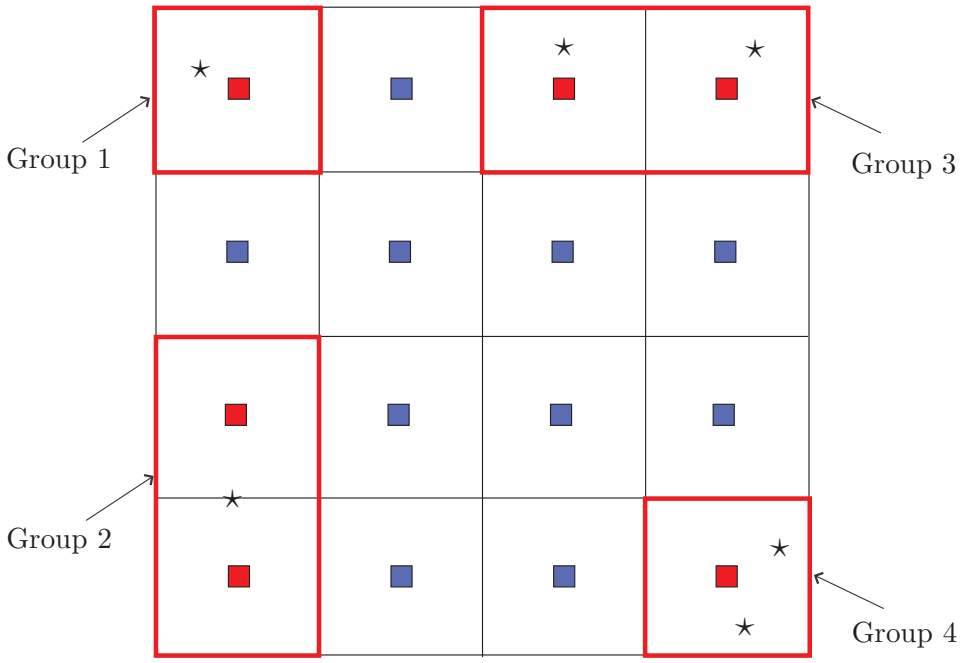


Figure 6.3: Region of interest in the case of multiple gas releases, where ■ represents the cluster heads when there is no alert, ■ when there is an alert, ⋆ represents the locations of the sources, and the red rectangle represents the defined groups. The merge of the clusters in alert according to their adjacencies leads here to four groups

but are being detected and estimated simultaneously, since the maximal concentrations would be at the release source. For illustration, consider the example of Figure 6.4, where the source is near the boundary of two clusters. The alert is then triggered in both clusters, the gas being spread all around the source. Since the clusters are adjacent, they are merged together in order to form one group. Then, the local maximum is estimated, leading to only one sensor, instead of two if the clusters were considered separately.

Having found the local concentration maxima, we now need to process the information in each group in order to estimate all N_s sources' parameters. We propose to treat the information collected around the local maxima instead of processing all the available information. This solution relies on the fact that the highest concentrations are collected by the sensors around the source, and these concentration measures are the most relevant for parameter estimation. Back to Figure 6.4, only the concentration information from the sensors around the red triangle denoting the maximum will be considered. Note that the size of the clusters and the number of sensors in the figure are only chosen for illustrative purposes.

We now propose to select the concentrations to be treated based on the advection-diffusion model of Subsection 6.2.3. In other words, we need to estimate the size of the small local zone around the local maximum as illustrated in Figure 6.4. Let C_{thres} be

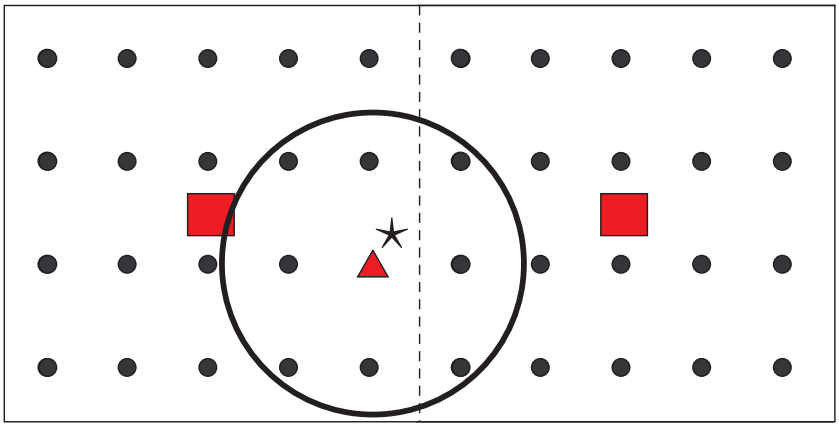


Figure 6.4: Group topology, where ■ represents the cluster heads, • represents the deployed sensors, ⋆ represents the location of the source, and ▲ represents a local maximum

a threshold under which the concentrations are considered really small, which means that they are measured by sensors far from the source. Depending on the considered application and the type of monitoring needed, one can determine an approximation of the maximum diffusion mass $Q_{\max}$. Using C_{thres} and $Q_{\max}$, we find the maximal distance between a source and a sensor. Then, this distance is used to determine the boundaries of the local zone, whose center is the local maximum and the estimated maximal distance is the distance separating the local maximum from the boundary of the zone. By taking the logarithm of (6.5), we get the following:

$$\log C_{\text{thres}} = \log\left(\frac{Q_{\max}}{4\ \pi^{\frac{3}{2}}\ (K_X K_Y K_Z)^{\frac{1}{2}}}\right) - \frac{3}{2}\log\Delta t - \frac{\Delta x^2 + u_X^2\Delta t^2 - 2u_X\Delta t\Delta x}{4K_X\Delta t} - \frac{\Delta y^2 + u_Y^2\Delta t^2 - 2u_Y\Delta t\Delta y}{4K_Y\Delta t}$$

The local zone is considered circularly shaped, which leads to $\Delta x = \Delta y$. In addition, having $K_X = K_Y$ yields the following:

$$\log C_{\text{thres}} = \log\left(\frac{Q_{\max}}{4\ \pi^{\frac{3}{2}}\ (K_X^2 K_Z)^{\frac{1}{2}}}\right) - \frac{3}{2}\log\Delta t - \frac{2\Delta x^2}{4K_X\Delta t} - \frac{(u_X^2\Delta t + u_Y^2\Delta t)}{4K_X} + \frac{\Delta x(2u_X + 2u_Y)}{4K_X}.$$

Finally, a second degree polynomial in terms of Δx is obtained as follows:

$$\mathfrak{a}\Delta x^2 + \mathfrak{b}\Delta x + \mathfrak{c} = 0, \tag{6.10}$$

where

$$\begin{cases} \mathfrak{a} & = -\frac{2}{4K_X\Delta t}, \\ \mathfrak{b} & = \frac{2u_X+2u_Y}{4K_X}, \\ \mathfrak{c} & = \log\left(\frac{Q_{\max}}{4\ \pi^{\frac{3}{2}}\ (K_X^2K_Z)^{\frac{1}{2}}}\right) - \frac{3}{2}\log\Delta t - \frac{(u_X^2.\Delta t+u_Y^2.\Delta t)}{4K_X} - \log C_{\text{thres}}. \end{cases}$$

The local zone dimensions $\Delta x = \Delta y$ are then obtained by resolving the quadratic equation (6.10), using $Q_{\max}$, C_{thres}, and initial approximative values of K_X and K_Z. Then, only the concentrations measured by the sensors in that zone will be considered for the source parameter estimation; in other words, the considered concentrations are the ones measured by the sensors at a maximal distance Δx from the local maximum. For simplicity, we assume that we have the same $Q_{\max}$ and C_{thres} for all the groups, thus for all N_s local maxima. Therefore, the same local zone size will be considered for all N_s sources. Nevertheless, depending on the type of application, one can define different sizes for the local zones around the N_s maxima.

Now let $\boldsymbol{c}^{(s)}$, $s \in \{1, ..., N_s\}$, denote the vector of useful concentrations collected at detection time around each local maximum s, i.e., around each source s. The first objective is to define an estimation model $\boldsymbol{\psi}_A$ that takes as input the concentration vector $\boldsymbol{c}^{(s)}$ and yields as output the vector of parameters $\boldsymbol{\theta}^{(s)}$ of source s, which includes the gas release mass $Q^{(s)}$, the source location $(x_0^{(s)}, y_0^{(s)}, z_0^{(s)}) = (x_0^{(s)}, y_0^{(s)}, 0)$ and the diffusivity constants. Then, the next objective is to define a second model $\boldsymbol{\psi}_B$, i.e., an enhancement model, that provides an enhanced estimation of the source location, using a part of the estimated source's parameters and the measured concentration vector $\boldsymbol{c}^{(s)}$. Kernel methods [Hofmann et al., 2008] provide an elegant framework to define both models, as it will be shown in the following subsection.

6.4.2 Definition of the models $\boldsymbol{\psi}_A$ and $\boldsymbol{\psi}_B$

Let V be the number of sensors in the local zone under investigation. As we mentioned earlier, only the concentrations around the local maxima will be considered in the estimation phase. Therefore, the vector $\boldsymbol{c}^{(s)}$ of size $V \times 1$ is the vector that will be processed to find the parameters of source s. In the following, we first define the estimation model $\boldsymbol{\psi}_A$, then we define the enhancement model $\boldsymbol{\psi}_B$.

6.4.2.1 First estimation

Here, we aim at defining a model $\boldsymbol{\psi}_A$ that associates to each concentration vector the corresponding source's parameters, namely

$$\begin{aligned}\boldsymbol{\psi}_A \colon \mathbb{R}^V &\rightarrow \mathbb{R}^5 \\ \boldsymbol{c}^{(s)} &\mapsto \boldsymbol{\theta}^{(s)} = \big(Q^{(s)} \;\; K_X \;\; K_Z \;\; x_0^{(s)} \;\; y_0^{(s)}\big).\end{aligned}$$

We propose to find the model $\boldsymbol{\psi}_A$ by solving a non-linear regression problem. The main benefit of such an approach is that no prior knowledge of the system is needed. As already shown in the previous chapters, kernel methods have been remarkably successful for solving non-linear regression problems. More specifically, we consider the kernel ridge regression to determine $\boldsymbol{\psi}_A$, by combining five separate optimization problems, one for each component of the output vector.

Consider the following training set $(\boldsymbol{c}_\ell, \boldsymbol{\theta}_\ell),\ \ell \in \{1, \ldots, N_{\text{reg}}\}$, where N_{reg} is the size of the training set used for the regression phase and $\boldsymbol{\theta}_\ell = (Q_\ell \;\; K_X \;\; K_Z \;\; x_{0\ell} \;\; y_{0\ell})$. The vector $\boldsymbol{c}_\ell$ is the vector of concentrations recorded by V sensors deployed in a zone of the same size as the small local zone introduced previously, from a gas diffusion of parameters $\boldsymbol{\theta}_\ell$, with $(x_{0\ell}, y_{0\ell}, 0)$ being the source location and Q_ℓ the gas release mass. As already mentioned, the eddy diffusivities K_X and K_Y, with $K_X = K_Y$, are assumed constant, since they depend on the atmospheric conditions and the type of gas, considered constant as well. Note here that the advantage of using a local small zone to select the relevant concentrations allows us to save computations and reduce the training complexity, since V sensors are now considered instead of N. Let $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1^\top \ldots \boldsymbol{\theta}_{N_{\text{reg}}}^\top)^\top$ be the matrix of size $N_{\text{reg}} \times 5$ having $\Theta_{\ell,i}$ for the (ℓ, i)-th entry, with $\ell \in \{1, \ldots, N_{\text{reg}}\}$ and $i \in \{1, \ldots, 5\}$. We also denote $\boldsymbol{\theta}_\ell$ by $\boldsymbol{\Theta}_{\ell,*}$, and the i-th column of $\boldsymbol{\Theta}$ by $\boldsymbol{\Theta}_{*,i}$.

Let $\boldsymbol{\psi}_A = (\psi_{A1} \ldots \psi_{A5})$ be the vector of functions, where each function $\psi_{Ai},\ i \in \{1, \ldots, 5\}$, estimates $\Theta_{\ell,i}$, the i-th component of the vector $\boldsymbol{\Theta}_{\ell,*}$, for an input $\boldsymbol{c}_\ell$. Each function ψ_{Ai} is determined by minimizing the mean quadratic error between the model's outputs $\psi_{Ai}(\boldsymbol{c}_\ell)$ and the desired outputs $\Theta_{\ell,i}$:

$$\psi_{Ai} = \arg \min_{\underline{\psi}_{Ai} \in \mathcal{H}_\mathcal{A}} \frac{1}{N_{\text{reg}}} \sum_{\ell=1}^{N_{\text{reg}}} (\Theta_{\ell,i} - \underline{\psi}_{Ai}(\boldsymbol{c}_\ell))^2 + \eta_A \|\underline{\psi}_{Ai}\|^2_{\mathcal{H}_\mathcal{A}}, \tag{6.11}$$

for some RKHS $\mathcal{H}_\mathcal{A}$, where η_A is the regularization parameter. According to the representer theorem, the optimal function ψ_{Ai} can be written as follows:

$$\psi_{Ai}(\cdot) = \sum_{\ell=1}^{N_{\text{reg}}} \beta_{\ell,i}\, \kappa_A(\boldsymbol{c}_\ell, \cdot), \tag{6.12}$$

where $\kappa_A : \mathbb{R}^V \times \mathbb{R}^V \to \mathbb{R}$ is a kernel, and $\beta_{\ell,i}, \ell \in \{1, \ldots, N_{\text{reg}}\}$, are parameters to be estimated. We now denote by $\boldsymbol{\beta}$ the $N_{\text{reg}} \times 5$ matrix whose (ℓ, i)-th entry is $\beta_{\ell,i}$. The vector $\boldsymbol{\beta}_{*,i}$ denotes then its i-th column, and $\boldsymbol{\beta}_{\ell,*}$ its ℓ-th row. By substituting (6.12) into (6.11), a dual optimization problem in terms of $\boldsymbol{\beta}_{*,i}$ is obtained, whose solution is given by taking the derivative of the corresponding cost function with respect to $\boldsymbol{\beta}_{*,i}$ and setting it to zero. We obtain the following result:

$$\boldsymbol{\beta}_{*,i} = (\boldsymbol{K}_A + \eta_A N_{\text{reg}} \mathbf{I})^{-1} \boldsymbol{\Theta}_{*,i},$$

where $\mathbf{I}$ is the $N_{\text{reg}} \times N_{\text{reg}}$ identity matrix, and $\boldsymbol{K}_A$ is the $N_{\text{reg}} \times N_{\text{reg}}$ Gram matrix whose (v, w)-th entry is $\kappa_A(\boldsymbol{c}_v, \boldsymbol{c}_w)$, for $v, w \in \{1, \ldots, N_{\text{reg}}\}$.

Since the same matrix $(\boldsymbol{K}_A + \eta_A N_{\text{reg}} \mathbf{I})$ needs to be inverted in order to estimate each source parameter, all five estimations can be collected into a single matrix inversion problem, thus reducing the computational complexity, as follows:

$$\boldsymbol{\beta} = (\boldsymbol{K}_A + \eta_A N_{\text{reg}} \mathbf{I})^{-1} \boldsymbol{\Theta}. \tag{6.13}$$

Finally, using equation (6.12) and the definition of the vector of functions $\boldsymbol{\psi}_A(\cdot)$, we can write $\boldsymbol{\psi}_A$ as follows:

$$\boldsymbol{\psi}_A(\cdot) = \sum_{\ell=1}^{N_{\text{reg}}} \boldsymbol{\beta}_{\ell,*} \, \kappa_A(\boldsymbol{c}_\ell, \cdot). \tag{6.14}$$

Once the model has been defined, it is applied throughout the network to get a first estimation of any source's parameters. Indeed, having detected a local maximum in the region, with its relevant concentration vector $\boldsymbol{c}^{(s)}$, one obtains a first estimate of the source s parameters as follows:

$$(\hat{Q} \quad \hat{K}_X \quad \hat{K}_Z \quad \hat{x}_0 \quad \hat{y}_0) = \boldsymbol{\psi}_A(\boldsymbol{c}^{(s)}). \tag{6.15}$$

6.4.2.2 Enhancement of the estimates

Next, we introduce the second model $\boldsymbol{\psi}_B$ that takes as input the measured concentrations, the estimated mass release and the estimated eddy diffusivities, and gives as output an enhanced estimate of the location of the source as follows:

$$\begin{array}{rccc} \boldsymbol{\psi}_B\colon & \mathbb{R}^{V+3} & \to & \mathbb{R}^2 \\ & \left(\boldsymbol{c}^{(s)\top} \quad Q^{(s)} \quad K_X \quad K_Z\right) & \mapsto & \left(x_0^{(s)} \quad y_0^{(s)}\right). \end{array}$$

Using the already estimated information as internal feedback is expected to provide an improved accuracy of the first location estimates of the sources.

The definition of the model $\boldsymbol{\psi}_B$ is also done using the kernel ridge regression. However, the training set is defined differently, such that the training input is given by $\boldsymbol{W}_\ell = (\boldsymbol{c}_\ell^\top \quad Q_\ell \quad K_X \quad K_Z)^\top$ and the training output is the source's location $(x_{0\ell} \quad y_{0\ell})$, $\ell \in \{1, \ldots, N_{\text{reg}}\}$. Let $\boldsymbol{\psi}_B = (\psi_{B1} \quad \psi_{B2})$, where ψ_{B1} and ψ_{B2} estimate $x_{0\ell}$ and $y_{0\ell}$ respectively. The models ψ_{B1} and ψ_{B2} are obtained by minimizing the mean quadratic errors between the estimated outputs and the desired ones, as follows:

$$\begin{cases} \psi_{B1} = \arg \min\limits_{\underline{\psi}_{B1} \in \mathcal{H}_\mathcal{B}} \dfrac{1}{N_{\text{reg}}} \displaystyle\sum_{\ell=1}^{N_{\text{reg}}} (x_{0\ell} - \underline{\psi}_{B1}(\boldsymbol{W}_\ell))^2 + \eta_B \|\underline{\psi}_{B1}\|^2_{\mathcal{H}_\mathcal{B}} \\ \psi_{B2} = \arg \min\limits_{\underline{\psi}_{B2} \in \mathcal{H}_\mathcal{B}} \dfrac{1}{N_{\text{reg}}} \displaystyle\sum_{\ell=1}^{N_{\text{reg}}} (y_{0\ell} - \underline{\psi}_{B2}(\boldsymbol{W}_\ell))^2 + \eta_B \|\underline{\psi}_{B2}\|^2_{\mathcal{H}_\mathcal{B}} \end{cases}$$

For some RKHS $\mathcal{H}_\mathcal{B}$, the quantity η_B is also a regularization parameter. In analogy with the previous paragraph, the optimal solutions can be written as follows:

$$\psi_{Bj}(\cdot) = \sum_{\ell=1}^{N_{\text{reg}}} \gamma_{\ell,j}\, \kappa_B(\boldsymbol{W}_\ell, \cdot),$$

where $\kappa_B : \mathbb{R}^{V+3} \times \mathbb{R}^{V+3} \to \mathbb{R}$ is a kernel, and $\gamma_{\ell,j}, \ell \in \{1, \ldots, N_{\text{reg}}\}$ and $j \in \{1, 2\}$, are the unknown parameters to be determined. Let $\boldsymbol{\gamma}$ denote the $N_{\text{reg}} \times 2$ matrix whose (ℓ, j)-th entry is $\gamma_{\ell,j}$, and $\boldsymbol{\gamma}_{\ell,*}$ its ℓ-th row. Following the same line of reasoning as in the previous paragraph, one can collect the two estimations into a single one. Then, the unknown parameters $\gamma_{\ell,j}$ are estimated at once, as follows:

$$\boldsymbol{\gamma} = (\boldsymbol{K}_B + \eta_B N_{\text{reg}} \mathbf{I})^{-1} (\boldsymbol{x}_0 \quad \boldsymbol{y}_0), \tag{6.16}$$

where $\boldsymbol{K}_B$ is the $N_{\text{reg}} \times N_{\text{reg}}$ matrix whose (v, w)-th entry is $\kappa_B(\boldsymbol{W}_v, \boldsymbol{W}_w)$, for $v, w \in \{1, \ldots, N_{\text{reg}}\}$. As for $\boldsymbol{x}_0$, it denotes the $N_{\text{reg}} \times 1$ vector whose ℓ-th entry is given by $x_{0\ell}$, and $\boldsymbol{y}_0$ denotes the $N_{\text{reg}} \times 1$ vector whose ℓ-th entry is given by $y_{0\ell}$. Finally, one is able to write $\boldsymbol{\psi}_B$ as follows:

$$\boldsymbol{\psi}_B(\cdot) = \sum_{\ell=1}^{N_{\text{reg}}} \boldsymbol{\gamma}_{\ell,*}\, \kappa_B(\boldsymbol{W}_\ell, \cdot). \tag{6.17}$$

After determining the training parameters of the model $\boldsymbol{\psi}_B$, the latter is also applied throughout the network to obtain an enhanced position of a given source. Indeed, back to the first estimates of the parameters of the source s given by equation (6.15), one can set the $(V+3) \times 1$ vector $\boldsymbol{W} = (\boldsymbol{c}^{(s)\top} \quad \hat{Q} \quad \hat{K}_x \quad \hat{K}_z)^\top$. Using the model $\boldsymbol{\psi}_B$, a new

estimate of the source location is obtained as follows:

$$(\hat{x}_{0\text{enh}} \quad \hat{y}_{0\text{enh}}) = \boldsymbol{\psi}_B(\boldsymbol{W}), \tag{6.18}$$

where $(\hat{x}_{0\text{enh}} \quad \hat{y}_{0\text{enh}})$ is the new enhanced location estimate of the source.

It is important to note that we are only using the first position estimate to compare its accuracy to the enhanced one. When implementing the algorithm in a practical setup, one does not need to find the first position estimate, since it is not used in the enhancement process, which then reduces the complexity of the algorithm in terms of storage and time.

6.5 Simulations

In this section, we evaluate the performance of the proposed detection and estimation method for different scenarios. We consider a region of study of dimensions 5000 m $\times$ 5000 m. Sensors are deployed over a uniform grid in the area at a rate of one sensor each five meters, which is equivalent to a density of 0.04 sensor/m^2. The region is divided into $Z = 25$ clusters with same size, thus having the same density of sensors. Figure 6.5 shows the region topology. The concentrations measured by the sensors are generated using the advection-diffusion model of (6.5). The maximum gas mass Q_{max} is taken equal to 1000 kg. The eddy diffusivity in the Z direction, K_Z is taken equal to 0.211 m^2/s, while the eddy diffusivities K_X and K_Y, in the X and Y directions respectively, are taken equal to 12 m^2/s. As for the mean wind velocity $\boldsymbol{U}$, it is taken equal to (1.8, 0, 0) m/s for illustrative purposes as given in [Kathirgamanathan et al., 2002]. Finally, the sampling time T is taken equal to 1 s. The rest of this section is organized as follows. The training parameters are defined in the first subsection. In the second subsection, an evaluation of the method is proposed in the case of a single source with different noise values. We also study the influence of the number of clusters Z and the density of sensors on the performance of the proposed method. Then, in the third subsection, an evaluation in the case of multiple gas sources is provided. Finally, in the fourth subsection, a comparison of the proposed framework to the state-of-the-art is provided.

6.5.1 Training parameters

First, we start by describing the training phase for the detection. As we already mentioned, we assume that the atmospheric conditions are the same in all the region of interest. Therefore, in order to reduce computations, we only define one detector; then,

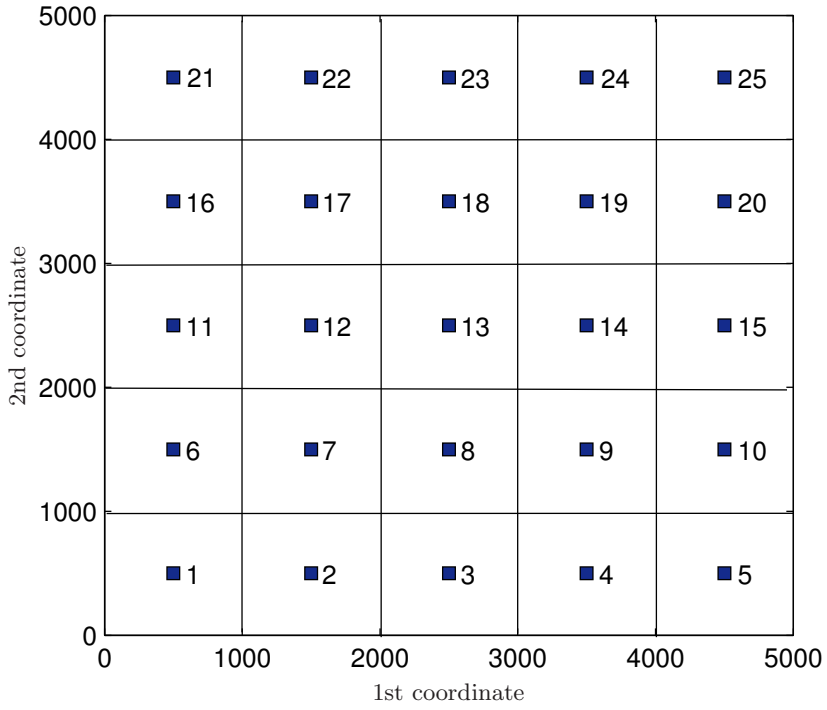


Figure 6.5: Simulated region topology, where ■ represents the cluster heads

this detector is communicated to all cluster heads in order to be used in the clusters. The advantage is that only one training phase is needed. The size of the training set for the detection N_{det} is taken equal to 330, where 300 are normal data and 30 are outliers. By outliers, we mean concentrations measured by sensors in the case of a gas diffusion. In order to generate the outlier data, 30 different gas diffusion scenarios are considered, where the source could be anywhere in the cluster, and the gas diffusion mass Q is also randomly varying between 5 kg and $Q_{\max} = 1000$ kg. The Gaussian kernel is considered in our simulations. Consequently, the detection kernel $\kappa : \mathbb{R}^{N(z)} \times \mathbb{R}^{N(z)} \to \mathbb{R}$ is given by the following:

$$\kappa(\boldsymbol{C}_i, \boldsymbol{C}_j) = \exp\left(-\frac{\|\boldsymbol{C}_i - \boldsymbol{C}_j\|^2}{2\sigma^2}\right),$$

where $i, j \in \{1, ..., N_{\text{det}}\}$, and σ is the kernel bandwidth.

Next, let us give a brief description of the training phase for the estimation phase of the proposed framework. According to Subsection 6.4.2, after finding the local maxima, only the concentrations around these maxima are used with the models $\boldsymbol{\psi}_A$ and $\boldsymbol{\psi}_B$. By solving (6.10), one can find the size of the zone encapsulating the useful concentrations. For $Q_{\max} = 1000$ kg and $C_{\text{thres}} = 10^{-5}$ kg/m^3, the solution is then $\Delta x = 26$ m. Therefore, we only need to use a zone of 52 m $\times$ 52 m for the training. Let the number of considered scenarios N_{reg} in the training phase be equal to 200, where $x_0 \in [0; 52]m$

and $y_0 \in [0; 52]m$. Here, Q is again randomly varying between 5 kg and 1000 kg. The Gaussian kernel is used here as well. The first kernel $\kappa_A : \mathbb{R}^V \times \mathbb{R}^V \to \mathbb{R}$ is then given by the following:

$$\kappa_A(\boldsymbol{c}_v, \boldsymbol{c}_w) = \exp\left(-\frac{\|\boldsymbol{c}_v - \boldsymbol{c}_w\|^2}{2\sigma_A^2}\right),$$

where $v, w \in \{1, ..., N_{\text{reg}}\}$, and σ_A is the kernel bandwidth that controls, together with the regularization parameter η_A, the degree of smoothness, noise tolerance, and generalization of the solution.

Due to the difference in the order of magnitude between the elements of the input $\boldsymbol{W}_v = (\boldsymbol{c}_v^\top \quad Q_v \quad K_X \quad K_Z)^\top$, $v \in \{1, ..., N_{\text{reg}}\}$, we consider four different distances and bandwidths for the second kernel $\kappa_B : \mathbb{R}^{V+3} \times \mathbb{R}^{V+3} \to \mathbb{R}$. Then, κ_B is given by the following:

$$\begin{aligned}\kappa_B(\boldsymbol{W}_v, \boldsymbol{W}_w) = &\exp\left(-\frac{\|\boldsymbol{c}_v - \boldsymbol{c}_w\|^2}{2\sigma_{B1}^2}\right) \times \exp\left(-\frac{(Q_v - Q_w)^2}{2\sigma_{B2}^2}\right) \\ &\times \exp\left(-\frac{(K_{Xv} - K_{Xw})^2}{2\sigma_{B3}^2}\right) \times \exp\left(-\frac{(K_{Zv} - K_{Zw})^2}{2\sigma_{B4}^2}\right),\end{aligned}$$

where $v, w \in \{1, ..., N_{\text{reg}}\}$, and $\sigma_{B1}, \sigma_{B2}, \sigma_{B3}, \sigma_{B4}$ are also the kernel bandwidths, with $\sigma_{B3} = \sigma_{B4}$, since K_X and K_Z have the same physical properties. The regularization parameters and the kernels' bandwidths are chosen in a way to minimize the error on the training set, using the 10-fold cross-validation technique [Stone, 1974]. Simulation results show that taking $\sigma_{B1} = \sigma_{B2} = \sigma_{B3} = \sigma_{B4}$ yields very close results to the ones obtained by taking different values for the bandwidths.

6.5.2 Evaluation of the performance for a single source

Now that we have defined all the parameters for the training of the detector and of the two regression models, we can evaluate the performance of the proposed method in the case of a single source. The source is randomly positioned in the region of interest, with $x_0 \in [0; 5000]m$ and $y_0 \in [0; 5000]m$, and $Q \in [5; 1000]$ kg. All other parameters are set according to the beginning of this section. The testing phase takes place during an interval of 10 s; the concentrations are measured each $T = 1$ s during this interval. The gas diffusion occurs at any time in this interval. Note that random relative noises varying from 1% to 5% are added to the simulated concentration vectors, in order to account for transmission impairments and sensing errors. Using the proposed method, the gas release is detected in the affected clusters, and the source parameters are estimated based on the measured concentrations from the clusters. Table 6.2 shows the mean percentages of errors on the estimated source parameters averaged over 50 Monte-Carlo

simulations. The diffusions are well detected in the affected zones, and one can see that the parameters of the source are accurately estimated. Also, notice the improvement in the source location after the enhancement phase. Moreover, one can see from the small increase in the percentages of errors that the method is robust to noise.

Table 6.2: Percentages of errors on the estimated source parameters for different random relative noises

Noise	$\hat{Q}$	$\hat{K}_x$	$\hat{K}_z$	$\hat{x}_0$	$\hat{y}_0$	$\hat{x}_{0\text{enh}}$	$\hat{y}_{0\text{enh}}$
1%	0.69	0.006	0.006	0.02	0.03	0.01	0.01
2%	0.75	0.007	0.007	0.03	0.04	0.01	0.02
3%	0.89	0.01	0.01	0.03	0.04	0.02	0.02
4%	0.96	0.02	0.02	0.04	0.04	0.02	0.02
5%	1.15	0.02	0.02	0.04	0.04	0.02	0.02

6.5.2.1 Influence of the number of clusters

Previously, the number of clusters is taken equal to $Z = 25$. In this paragraph, we keep the density of sensors to one sensor each five meters (0.04 sensor/m^2) as in the previous paragraphs, and we vary the number of clusters, such that $Z = 3^2, 4^2, \ldots, 7^2$. We also consider a random relative noise of 5%. We start by evaluating the time complexity of our method for the different considered values of Z. To this end, we measure the elapsed time for the cross-validation in the training phase and for the test phase, for both detection and estimation phases. Note that for the training phase, we consider the same parameters as in Subsection 6.5.1, and simulations are run on version 7.10.0.499 of Matlab on a Dell laptop with Windows 7 and Intel Core i7 CPU. Table 6.3 shows the measured elapsed times in seconds. One can see that the estimation phase is not affected by the number of clusters, since it depends on the density of sensors. As for the detection phase, one can see that the smaller the number of clusters, the slower the cross-validation and the test phases. Indeed, when we decrease the number of clusters, while keeping the same density of sensors in the network, the number of sensors in each cluster $N^{(z)}$ increases. This increase induces an increase in the time complexity of the method. Now let us consider the detection phase in the case of $Z = 9$, which is the most computational demanding one. The cross-validation time is the highest here; however, since the training phase is performed only once before the monitoring process, this does not really affect the performance later on. Nevertheless, the test time is of 1.05 s, which is really high considering that concentrations are measured each 1 s. Therefore, the more clusters are available, the more performant the proposed framework.

Table 6.3: Elapsed cross-validation and test times for both detection and estimation phases, when varying Z

		$Z = 9$	$Z = 16$	$Z = 25$	$Z = 36$	$Z = 49$
Cross-validation time	- detection	290	165	107	76	56
Test time	- detection	1.05	0.52	0.31	0.23	0.17
Cross-validation time	- estimation	171	172	171	171	172
Test time	- estimation	0.005	0.005	0.005	0.005	0.005

Finally, it is important to note here that the variation of the number of clusters does not have any effect on the performance of the detector, and we still have a correct detection for all considered values of Z. As for the accuracy in the estimation of the parameters, it is also not affected by the changes in Z, since it only depends on the density of the sensors, that is fixed in this paragraph.

6.5.2.2 Influence of the sensor density

We now study the influence of the number of deployed sensors N, that is the density of the sensors. Previously, we considered the case of one sensor for each five meters; now, we will consider the cases of one sensor each ten, fifteen and twenty meters. We consider, here as well, a random relative noise of 5%. Table 6.4 shows, for different sensor densities, the elapsed times in seconds with the cross-validation in the training phase and with the test phase, for both detection and estimation phases. One can see that when the number of sensors is decreased, the computational time for both detection and estimation phases is decreased. Indeed, when decreasing N, we are also decreasing the number of sensors $N^{(z)}$ in each cluster, which explains the decrease in the computational time in the detection phase. As for the decrease in the computational time in the estimation phase, it is explained by the fact that decreasing the density of sensors induces a decrease in the number of relevant concentrations, i.e., a decrease in V.

Now we examine the accuracy of the proposed framework. Table 6.5 shows the mean percentages of errors on the estimated source parameters averaged over 50 Monte-Carlo simulations, for different sensor densities. We still have a 100% detection, and one can see that with the decrease of the number of sensors, we have a decrease in the accuracy of the estimation of the parameters. Indeed, when decreasing the number of sensors, we do not have a good coverage of the zone of interest, which explains the deterioration in the results. Nevertheless, one can see that we still have accurate results. It is important to highlight here that for the case of a density of one sensor each twenty meters, the proposed estimation method is not steady anymore. In fact, in cases where the gas

diffusion occurs close to the deployed sensors, its parameters are accurately estimated; however, in cases where the diffusion occurs far from the sensors, due to the bad coverage, the parameters are not well estimated. We can then conclude that it is essential to have a good coverage of the area in order to obtain accurate results and stability. In such a scenario, it is better not to have less than one sensor each fifteen meters.

Table 6.4: Elapsed cross-validation and test times for both detection and estimation phases, for different densities

		1 sensor each 5 m	1 sensor each 10 m	1 sensor each 15 m	1 sensor each 20 m
Cross-validation time	- detection	107	31	15	11
Test time	- detection	0.31	0.23	0.05	0.04
Cross-validation time	- estimation	171	105	91	84
Test time	- estimation	0.005	0.004	0.004	0.003

Table 6.5: Percentages of errors on the estimated parameters, for different densities

	1 sensor each 5 m	1 sensor each 10 m	1 sensor each 15 m	1 sensor each 20 m
$\hat{Q}$	1.15	2.56	15.31	45.71
$\hat{x}_{0\text{enh}}$	0.02	0.08	0.25	3.67
$\hat{y}_{0\text{enh}}$	0.02	0.08	0.23	3.56
$\hat{K}_x$	0.02	0.04	2.35	3.84
$\hat{K}_z$	0.02	0.04	2.34	3.85

6.5.3 Evaluation of the performance for multiple sources

In this section, we evaluate the accuracy of the method in the case of multiple gas sources, having different parameters and occurring at different times. To this end, consider four sources, whose parameters are given in Table 6.6. The concentrations are generated using (6.5), and a random relative noise of 5% is considered. The test phase is run here as well during an interval of 10 seconds. The subplots of Figure 6.6 show the concentrations measured by the sensors around Source 2, whose parameters are given in Table 6.6. Notice how the maximal concentration at each time t is moving along with the X-axis, and its value is decreasing with time because of the diffusion phenomenon. The estimated sources parameters and time of detection of each source are given in Table 6.7. One can see that the estimations are accurate, showing that the proposed method can also be used for the case of multiple sources.

Table 6.6: Parameters of the four sources

	Source 1	Source 2	Source 3	Source 4
t_0 (s)	2	3	5	5
Q (kg)	500	1000	20	200
x_0 (m)	100	100	2500	4500
y_0 (m)	100	4000	3000	1500
K_X (m^2/s)	12	12	12	12
K_Z (m^2/s)	0.211	0.211	0.211	0.211

Table 6.7: Estimation of the parameters of the four sources using the proposed framework

	Source 1	Source 2	Source 3	Source 4
detection (s)	3	4	6	6
$\hat{Q}$ (kg)	503.5	993	20.3	201.1
$\hat{x}_{0\text{enh}}$ (m)	100.0	100.2	2499.9	4500.3
$\hat{y}_{0\text{enh}}$ (m)	100.1	3999.9	2999.2	1499.8
$\hat{K}_x$ (m^2/s)	12.01	11.99	11.98	11.99
$\hat{K}_z$ (m^2/s)	0.211	0.211	0.211	0.211

6.5.4 Comparison to the state-of-the-art

In this subsection, we compare the percentages of errors to the ones obtained with the method in [Kathirgamanathan et al., 2002], for different values of the random relative noise varying from 1% to 5%. In [Kathirgamanathan et al., 2002], the source parameters are estimated in an area of around 5500 m $\times$ 5500 m, using the measured concentrations and an inverse model obtained after solving a non-linear least squares estimation problem. The source is placed at a fixed location with $x_0 = 5000$ and $y_0 = 100$, and the release rate Q is taken equal to 1000 kg. Table 6.8 shows the percentages of errors for both methods, where PM denotes our method and KM the one proposed by Kathirgamanathan et al. [2002]. Note that in this table, for the proposed method, only the enhanced estimates of the source's location are taken into account. Also, results over K_Z are not shown, since it is assumed to be known in [Kathirgamanathan et al., 2002]. When comparing the errors, one can see that the proposed method greatly outperforms the KM method in terms of accuracy. Moreover, it is important to see here that the proposed method is much more robust to the increase of the noise compared to the KM.

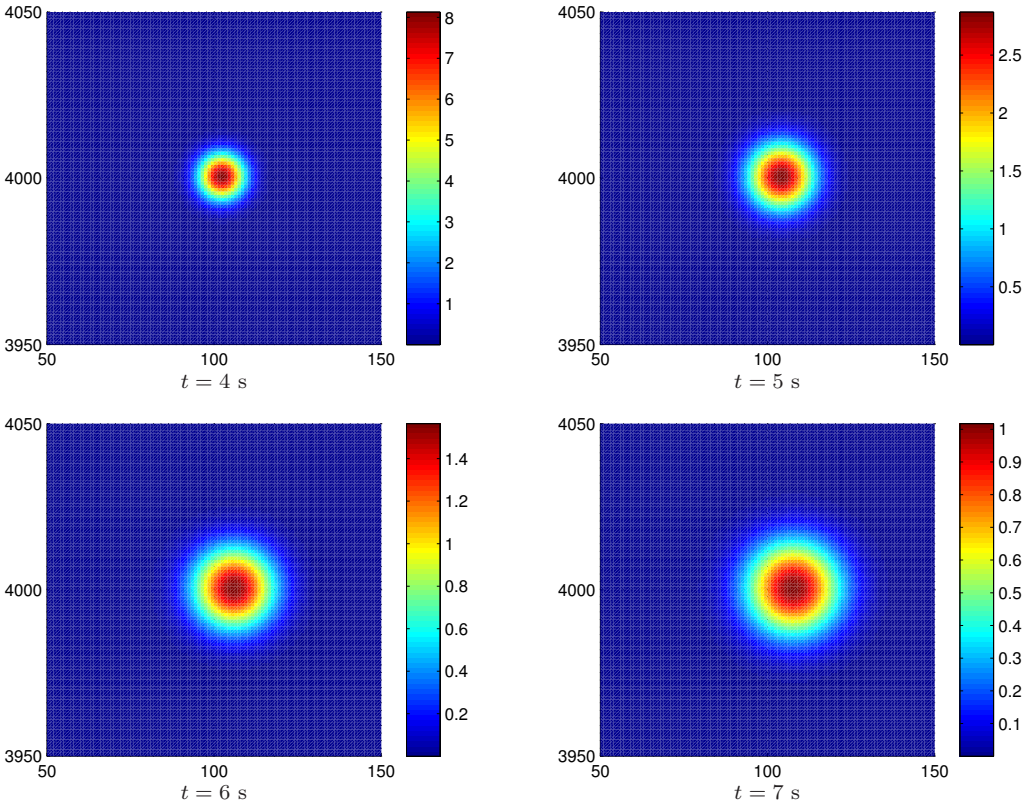


Figure 6.6: Concentration distribution in kg/m^3 at different times t around Source 2 activated at $t_0 = 3$ s

6.6 Conclusion

In this chapter, we introduced a new clusterized method for the detection and estimation of the parameters of multiple gas sources in wireless sensor networks. Both phases of the method are defined within the framework of kernel methods, proved to be very efficient for nonlinear detection and estimation. Indeed, the evaluation of the proposed method on simulated data showed that the method yields accurate estimates, and the accuracy is maintained even with the increase of the noise level and in the case of multiple sources. Future work will handle further improvements of this method, such as to include cases where the eddy diffusivities or the wind's velocity and direction are not constant.

Table 6.8: Comparison of the percentages of errors obtained by the proposed method (PM) and the method in [Kathirgamanathan et al., 2002] (KM), for different random relative noises

Noise	Method	$\hat{Q}$	$\hat{K}_x$	$\hat{x}_0$	$\hat{y}_0$
1%	PM	0.32	0.01	0.01	0.02
	KM	2.7	0.7	0.8	0.4
2%	PM	0.41	0.01	0.01	0.01
	KM	5.8	1.4	1.5	0.7
3%	PM	0.52	0.01	0.01	0.02
	KM	8.0	2.2	2.4	1.1
4%	PM	0.82	0.02	0.01	0.02
	KM	11.0	2.7	2.9	1.6
5%	PM	1.05	0.02	0.01	0.03
	KM	12.3	3.7	4.0	1.9

Chapter 7

Concluding Remarks

Contents

This manuscript addressed several problems in wireless sensor networks. First, an original centralized localization method was proposed within the framework of kernel methods. This method was later extended to a clusterized scheme, allowing us to consider the needs of large-middle-scale networks. Next, using a Kalman filter, the inertial information of mobile nodes were considered in order to obtain more accurate position estimates. We also proposed two new original models that characterize the relationship between the power of signals exchanged between sensors and the distance separating these sensors. Using these models and the inertial information, we proposed two other tracking methods employing either a Kalman filter or a particle filter. Finally, in the last part of this thesis, we were able to monitor a region of interest and detect any gas diffusion. Moreover, the proposed method allowed the estimation of the gas diffusion parameters. A brief recalling over all previously discussed methods is given in this chapter, along with some future research perspectives.

7.1 Summary of Contributions

The work presented in the preceding chapters has concentrated on two major applications in wireless sensor networks. The first one is the localization and tracking of mobile nodes in the network, while the second one consists in monitoring gas diffusions in a given region. The main contribution of this thesis can be summarized as follows.

In Chapter 3, we introduced an original kernel-based model for localization in WSNs using radio-location fingerprinting. This model allowed us to estimate the position of a given node using the received signal strength indicators (RSSI) of the signals exchanged between this node and the fixed sensors in the network. Several kernel-based machine learning techniques were investigated for the definition of this model. The proposed method was developed in both centralized and clusterized schemes. The results showed that both schemes can be used for accurate estimations in the case of moderate intensities of the additive noise over the RSSIs. To improve the performance of the method when the noise over the RSSIs is high, we proposed in Chapter 4 to take advantage (when available) of the inertial information of the mobile node, i.e., of the target. This information was combined with the estimated positions obtained using the kernel-based model by means of a Kalman filter. The results showed that this combination gave more accurate position estimates. The main constraint of this proposed method was that the sensors should remain fixed at the same locations during the tracking process, which can be binding for some applications.

In order to overcome the stationarity problem of the sensors, we proposed to consider the distances separating the target from these sensors for the tracking process. This way, we only needed to known the positions of the sensors, but they were free to move in the network. Note that the estimation of the distances between sensors was and still is really challenging. Nevertheless, in Chapter 5, we proposed two new distance models, such that, given the RSSIs of the signals exchanged between sensors, the models accurately estimated the distances separating these sensors. The first proposed distance model is non-parametric, not needing any prior knowledge of the physical system, while the second one is a semi-parametric regression model, combining the well-known log-distance theoretical propagation model with a non-linear fluctuation term, estimated within the framework of kernel-based machines. We then proposed two new tracking methods that combine the estimated distances and the inertial information of the target, using either the Kalman filter or the particle filter.

Finally, in Chapter 6, an original clusterized method for monitoring gas diffusions in WSNs was introduced. This method consisted in detecting and estimating the the parameters of multiple gas sources in WSNs, using collected concentration measures

from the studied region. The detection and estimation phase were both defined within the framework of kernel methods. Indeed, in order to detect any anomaly in the area, we defined a kernel-based detector using one-class classification techniques. Then, we defined two kernel-based estimation models that estimate the parameters of a source diffusion. The method proved to be accurate in the case of a single source, as well as for multiple sources.

7.2 Future Research Directions

This thesis provided several important solutions for localization and tracking, as well as monitoring gas diffusion in wireless sensor networks. As part of future research, we would like to investigate the following aspects concerning improvements of our proposed methods.

- *Choosing relevant RSSI information*
  The proposed localization and tracking methods were all based on the assumption that all sensors are able to communicate and exchange signals with the target. However, in real scenarios, some sensors may not be in the communication range of the target, leading to missing RSSI information. It is then important to adapt the proposed methods to such cases, by replacing for instance the non available information by a predefined value. On the other hand, we do not necessarily need all available RSSI information. It is then wiser to choose the relevant RSSIs instead of using all of them. For instance, this can be done by dynamically selecting sensors to be included in the estimation process using a predefined threshold on the power level of the signals received by the target from the sensors.

- *Improving the distance models*
  In the definition of the proposed distance models and the definition of the well-known models, such as log-distance theoretical propagation model and the polynomial models, the sensors' antennas are assumed to be isotropic. They are assumed to radiate uniformly in all directions; however, a uniform communication over all directions may not always be possible. A possible solution would be to add orientation information to the model, leading then to a more representative map of the available network's communications.

- *Improving the monitoring method*
  Now for the approach proposed for the detection and estimation of gas diffusion sources, future work will handle further improvements, such as to include cases where the eddy diffusivities or the wind's velocity and direction are not constant.

We will also study the possibility of using transfer learning, that proved to be helpful in cases where not enough training data are available.

- *Clustering solutions*
  In the clusterized localization method, as well as the clusterized monitoring approach, clusters are rectangular of equal sizes. Future works will include studies concerning the definition of the clusters, and an algorithm that automatically divides the region into clusters will be developed.

Appendix A

Résumé en français

Contents

Les réseaux de capteurs sans fil constituent une technologie en plein développement depuis les deux dernières décennies et jouent un rôle de plus en plus important dans un grand nombre d'applications. L'une des problématiques principales des réseaux de capteurs sans fil est la géolocalisation des capteurs qui s'avère primordiale pour la plupart des applications. Dans ce manuscrit, nous abordons d'une part le problème de la localisation des capteurs et du suivi de cibles dans les réseaux de capteurs sans fil. Les solutions proposées, basées sur des données de fingerprinting et de mobilité, se situent dans le cadre de l'apprentissage par méthodes à noyaux. D'un autre côté, nous traitons le problème de la surveillance de la diffusion d'un gaz dans une zone d'intérêt, basée aussi sur l'apprentissage par noyaux. Cette partie résume les travaux réalisés autour de ces deux problèmes.

A.1 Introduction

Les réseaux de capteurs sans fil (RCSF) sont des réseaux composés d'un grand nombre de capteurs intelligents [Akyildiz et al., 2002; Vieira et al., 2003; Sohraby et al., 2007]. Les capteurs, collaborant entre eux pour l'analyse des données recueillies, révolutionnent l'accès et le traitement de l'information dans plusieurs domaines d'applications tels que le domaine militaire, médical, environnemental, de sécurité, etc. Dans ce qui suit, nous décrivons brièvement quelques caractéristiques des RCSF. Nous exposons ensuite le problème de localisation des capteurs et de suivi de cibles. Finalement, nous exposons le problème de la surveillance de la diffusion d'un gaz à base de réseaux de capteurs.

A.1.1 Présentation générale des RCSF

Les capteurs constituant un RCSF sont dits capteurs intelligents. Ils sont munis d'un capteur physique (capteurs de chaleur, d'humidité, de vibrations, de gaz comme par exemple CO, NO_2, NH_3, SO_2, CH_4, H_2S ...), d'unités de traitement et de stockage de données et d'un module de communication sans fil. Ces capteurs sont autonomes mais à durée de vie limitée. En effet, leurs principales sources d'énergie sont des batteries non renouvelables. Ainsi, pour maximiser la durée de vie du réseau, il conviendra de veiller à réduire au maximum la consommation d'énergie de ces capteurs. La figure 1.1 illustre le schéma fonctionnel d'un capteur intelligent.

Un point essentiel dans le fonctionnement des réseaux de capteurs est le choix d'une architecture du réseau. Ce choix influe fortement sur les performances du réseau. Nous distinguons principalement trois types de topologies dans les RCSF définissant la façon dont les capteurs communiquent ensemble : centralisée, distribuée, et hybride [Akyildiz

et al., 2002; Bandyopadhyay and Coyle, 2003; Jardosh and Ranjan, 2008; Mourad, 2010; Mamun, 2012]. La figure 1.2 illustre ces trois principales topologies. Dans le cas d'une topologie centralisée, les capteurs acheminent toutes les données mesurées vers la station de fusion centrale où elles seront stockées et traitées. Ce genre de réseaux fournit une haute qualité de traitement mais tout en consommant énormément d'énergie. Dans une topologie distribuée, les capteurs ont la capacité de traiter localement les données mesurées ; ils n'échangent ainsi avec leurs voisins que des données traitées, d'habitude de taille réduite. Ce type de réseaux est beaucoup plus robuste à la défaillance que le mode centralisé ; en effet, la panne d'un capteur en mode distribué n'affectera pas tout le réseau contrairement à une panne fatale du centre de fusion en mode centralisé. Toutefois, la topologie distribuée nécessite des algorithmes distribués qui sont souvent plus difficile à développer (décomposition en sous-problèmes, propriétés de convergence, ...) et parfois moins précis que leurs versions centralisées. La topologie hybride combine les idées des deux autres topologies. Elle consiste à partitionner le réseau en des groupes de capteurs, appelés *clusters*, chacun ayant une tête de cluster ou *cluster head*, chargé du traitement et du routage des données. Cette topologie permet d'assurer un équilibre optimal entre la consommation d'énergie et la capacité de traitement, ce qui convient donc à des réseaux à grandes échelles.

La diminution de la taille et du coût des capteurs, ainsi que la diversité des types de capteurs disponibles et l'évolution des communications sans fil, ont élargi le champ d'application des réseaux de capteurs sur plusieurs domaines. Nous citons en particulier le domaine militaire pour la localisation de l'ennemi et la surveillance des zones hostiles [Lee et al., 2009; Watthanawisuth et al., 2011], le domaine environnemental pour la détection des feux de forêt [Yu et al., 2005], pour le contrôle de la qualité de l'eau ou de l'air [Chaamwe, 2010; S. De Vito, 2012], le domaine médical pour surveiller les activités des personnes âgées et des personnes atteintes de la maladie d'Alzheimer [Suryadevara and Mukhopadhyay, 2012], etc.

A.1.2 Problème de localisation et de suivi de cibles

Dans la plupart des applications, la localisation des capteurs s'avère primordiale afin de pouvoir lier les données mesurées à une position géographique. Une solution intuitive au problème de localisation est d'équiper tous les capteurs de GPS [B. Hofmann-Wellenhof and Collins, 2004]. Cependant, cette solution n'est pas optimale pour plusieurs raisons [Benbadis et al., 2005; Liu et al., 2010]. Entre autres, le prix et la taille des récepteurs GPS peuvent être contraignants pour plusieurs applications, surtout pour celles qui nécessitent un grand nombre de capteurs (de l'ordre de milliers par exemple). Une autre contrainte est le fait que l'utilisation de récepteurs GPS consomme beaucoup d'énergie,

réduisant ainsi la durée de vie du réseau. Finalement, le service GPS est indisponible ou de qualité médiocre dans des milieux indoor. Ainsi, cette technologie est, jusqu'à maintenant, limitée à une utilisation dans des environnements extérieurs.

Une solution alternative est de considérer deux types de capteurs : les *ancres* et les *nœuds non-ancres*. Les ancres ont des positions connues, alors que les nœuds non-ancres, ou tout simplement *nœuds*, ont des positions inconnues. Les positions des ancres peuvent être obtenues à l'aide de dispositifs GPS si possible, ou alors elles sont fixes, les ancres étant dans ce cas installées à des positions connues. Le problème de localisation se résume ainsi à estimer les positions des nœuds en se basant sur les mesures échangées avec les ancres. Il existe de nombreuses méthodes de localisation à base d'ancres répondant aux exigences des réseaux de capteurs sans fil. Entre autres, nous citons les méthodes basées sur la mesure de la puissance des signaux reçus ou *received signal strength indicator (RSSI)* [Medeisis and Kajackas, 2000; Gholami et al., 2013], la mesure du temps de propagation des signaux [Pal, 2010; Okello et al., 2011], l'angle d'arrivée des signaux [Niculescu and Nath, 2003; Rong and Sichitiu, 2006], etc. Les méthodes mesurant le temps de propagation des signaux nécessitent une synchronisation entre les capteurs, ce qui n'est pas le cas des techniques mesurant les angles. Toutefois, ces dernières nécessitent un matériel volumineux et plus coûtant que les autres techniques. Comparées aux méthodes basées sur les temps de propagation et les angles, les méthodes basées sur les RSSIs sont largement utilisées en raison de leur faible coût et de leur consommation faible en termes d'énergie, puisqu'aucun dispositif supplémentaire n'est nécessaire pour les capteurs. Cependant, ces méthodes sont sensibles à la présence du bruit et des interférences.

Une des applications les plus prometteuses des RCSF est le suivi de cibles ou *target tracking* [Lau and Chung, 2007; Zhang et al., 2013; Xu et al., 2013]. Le suivi de cibles peut être considéré comme un problème de localisation séquentielle, nécessitant donc un algorithme précis de localisation. La base du suivi de cibles est d'estimer de manière récursive les positions et les vitesses de la cible, qui sont souvent désignées comme des *états* [Svensson, 2010]. Une première étape est de prédire la valeur de l'état futur en utilisant un modèle qui décrit le mouvement de la cible, appelé *le modèle d'état*. Ensuite, l'état prédit est mis à jour en utilisant des informations supplémentaires qui pourraient couvrir toute mesure pertinente collectée du réseau. Cette étape nécessite une relation entre les états et les mesures collectées, ou un *modèle d'observation*. Afin de combiner les informations des deux modèles définis, nous utilisons différents types de filtres, le filtre de Kalman [Kalman, 1960; Welch and Bishop, 2001] ou le filtrage particulaire [Branko Ristic, 2004].

A.1.3 Problème de la surveillance de la diffusion d'un gaz

Dans un autre contexte, les RCSF sont largement utilisés pour la surveillance de la diffusion d'un gaz dans l'environnement [Christopoulos and Roumeliotis, 2005; Ickowicz et al., 2012]. Compte tenu des conséquences potentiellement catastrophiques des diffusions de gaz nocifs, il est important de détecter le plus tôt possible les diffusions et d'estimer les paramètres des sources de gaz comme leurs positions et les quantités des gaz libérés.

Les émissions de gaz peuvent être divisées en deux catégories : délibérées ou accidentelles. Les émissions délibérées se produisent habituellement pendant des périodes de guerres ou sont causées par des attaques terroristes. Une des premières attaques majeures a eu lieu en 1995 à Tokyo, au Japon, à l'intérieur d'un train métro bondé [Tu, 1999]. Son objectif était d'infliger la terreur parmi la population en libérant un agent neurotoxique, conçu pour paralyser et tuer les personnes en quelques minutes. Quant aux émissions de gaz accidentelles, elles se produisent habituellement dans des industries, des mines, des volcans, etc. Un des pires exemples est la catastrophe de Bhopal, en Inde, en 1984 [Kalelkar, 1988]. Cette catastrophe est la conséquence de l'explosion d'une usine de pesticides dégageant quarante tonnes d'isocyanate de méthyle dans l'atmosphère.

A.1.4 Contributions

Dans le cadre ce cette thèse, nous nous intéressons en premier lieu à la localisation des capteurs. Nous proposons ainsi une approche basée sur les puissances des signaux échangés entre les capteurs. En utilisant la technique de fingerprinting et les méthodes à noyaux pour la régression, nous obtenons une très bonne estimation de la position des capteurs. Nous revisitons ensuite ce travail pour l'adapter au suivi de cibles. Cette méthode permet d'améliorer l'estimée de la position de la cible en tenant compte du mouvement de la cible, et cela à l'aide du filtre de Kalman.

Nous proposons également un modèle semi-paramétrique et un non-paramétrique qui permettent d'estimer les distances séparant les capteurs du réseau d'une cible en mouvement à l'aide des RSSIs. La position de la cible est ensuite obtenue en combinant les informations d'accélération et les distances estimées, en s'appuyant sur un filtrage de Kalman ou sur un filtrage particulaire.

Finalement, nous proposons une méthode pour la surveillance de la diffusion d'un gaz dans une zone d'intérêt, en utilisant l'apprentissage par noyaux. Cette méthode permet de détecter la diffusion d'un gaz en utilisant des concentrations relevées régulièrement par des capteurs déployés dans la zone d'intérêt. Les concentrations mesurées sont

ensuite traitées pour estimer, à l'aide des modèles proposés, les paramètres de la source du gaz comme sa position et la quantité du gaz libéré.

A.2 Localisation par méthodes à noyaux

Comme nous venons d'expliquer, il est important de localiser les capteurs dans un RCSF pour un grand nombre d'applications. Les méthodes de localisation basées sur les RSSIs sont considérées les plus populaires puisqu'elles présentent de nombreux avantages du point de vue de leur simplicité ainsi que du coût de matériel. Certaines de ces techniques exploitent l'atténuation de l'intensité du signal avec la distance parcourue pour estimer les distances séparant les capteurs. Cependant, ces techniques ne sont pas toujours efficaces, puisque les signaux sont sensibles à la présence du bruit et des interférences. Ainsi, au lieu de convertir les RSSIs en distances, des techniques alternatives comparent la puissance du signal reçu à un seuil prédéfini, conduisant ainsi à des données de connectivités et donc des contraintes sur les distances [Y. Shang and Fromherz, 2003]. Les distances seront ensuite combinées à l'aide du filtrage particulaire [Ozdemir et al., 2009], de l'analyse par intervalles [Mourad et al., 2009, 2011], du filtre variationnel [Teng et al., 2010], etc. D'autres méthodes s'appuient sur la technique de fingerprinting [Lin and Lin, 2005] qui consiste à collecter des informations du réseau pour constituer une base de données à utiliser pour la localisation. Comme ces méthodes ne font aucune hypothèse sur les communications dans le réseau, elles sont plus fiables que celles utilisant des mesures de connectivités.

Dans cette section, nous proposons une approche originale de localisation basée sur la technique de fingerprinting et l'apprentissage par noyaux. Pour cela, nous définissions dans le cadre des méthodes à noyaux un modèle qui prend en entrée la puissance du signal échangé entre une ancre et un nœud et donne en sortie une estimation de la position du nœud en question. Pour définir ce modèle, nous investiguons plusieurs techniques d'apprentissage [Aronszajn, 1950], comme la technique de kernel ridge regression [Saunders et al., 1998], de support vector regression (SVR) [Vapnik, 1995] et de vector-output regularized least squares (vo-RLS) [Honeine et al., 2013].

A.2.1 Problématique

Nous considérons un environnement à δ dimensions, avec par exemple $\delta = 2$ pour un environnement à deux dimensions, et N_a ancres ayant des positions connues, notées $\boldsymbol{a}_i,\ i \in \{1, \ldots, N_a\}$. L'approche proposée est centralisée (voir figure 1.2(a)) ; ainsi, toutes les données recueillies sont acheminées vers la station centrale de fusion, où tous les

traitements et calculs seront effectués. Nous considérons ensuite une topologie hybride pour notre approche, avec plusieurs centres de fusion locaux au lieu d'un seul. Comme il sera montré plus tard dans l'étude, cette topologie permet la distribution du traitement de l'information entre plusieurs têtes de cluster et donc la réduction de la probabilité de défaillance du réseau. Basée sur la technique de fingerprinting, notre méthode consiste à construire une base de données en offline, à utiliser par la suite en temps réel, pour estimer la position d'un nœud, notée $\boldsymbol{x}$. Notons que plusieurs nœuds peuvent être localisés à la fois, puisqu'ils sont localisés chacun indépendamment des autres.

Pour construire la base de données, nous considérons N_p positions de référence, notées $\boldsymbol{p}_\ell = (p_{\ell,1} \dots p_{\ell,\delta})^\top, \ell \in \{1, \dots, N_p\}$, et générées de façon uniforme ou aléatoire dans la région, comme le montre la figure 3.1. Les ancres diffusent régulièrement des signaux dans la région étudiée. En posant un capteur consécutivement à chacune des positions de référence, il serait possible de mesurer les RSSIs des signaux qu'il reçoit. Soit $\boldsymbol{\rho}_\ell$ le vecteur des RSSIs des signaux envoyés par les N_a capteurs et mesurés à la position $\boldsymbol{p}_\ell$. Un ensemble d'apprentissage de N_p couples $(\boldsymbol{\rho}_\ell, \boldsymbol{p}_\ell)$ est ainsi obtenu, avec $\ell \in \{1, \dots, N_p\}$. La phase d'apprentissage consiste par la suite à trouver un modèle $\boldsymbol{\psi}(\cdot)\colon \mathbb{R}^{N_a} \to \mathbb{R}^{\delta}$ qui prend en entrée un vecteur RSSI $\boldsymbol{\rho}_\ell$, et donne en sortie la position correspondante $\boldsymbol{p}_\ell$ (figure 3.2). Le nœud à localiser mesure en temps réel le vecteur des RSSIs des signaux reçus des N_a ancres, noté $\boldsymbol{\rho}$. L'estimée de la position de ce nœud est alors obtenue en utilisant le modèle ψ comme suit : $\hat{\boldsymbol{x}} = \boldsymbol{\psi}(\boldsymbol{\rho})$.

A.2.2 Définition du modèle $\boldsymbol{\psi}$

La régression est l'une des plus grandes préoccupations des chercheurs dans le domaine de la modélisation des systèmes. Elle est utilisée pour prédire une variable continue à partir d'une certaine entrée. L'objectif de cette sous-section est de définir, en utilisant les méthodes à noyaux, le modèle $\boldsymbol{\psi}(\cdot)$ qui associe à une entrée $\boldsymbol{\rho}_\ell$ la sortie correspondante $\boldsymbol{p}_\ell$ et qui permet donc d'estimer à partir d'un nouveau vecteur RSSI la position qui y est associée. Pour ce faire, nous considérons l'ensemble d'apprentissage suivant : $\{(\boldsymbol{\rho}_\ell, p_{\ell,d})\}_{\ell=1}^{N_p}$, $d \in \{1, \dots, \delta\}$, où $p_{\ell,d}$ est le d-ème élément de $\boldsymbol{p}_\ell = (p_{\ell,1} \dots p_{\ell,\delta})$. Soit $\boldsymbol{\psi}(\cdot) = (\psi_1(\cdot) \dots \psi_\delta(\cdot))$, où $\psi_d(\cdot)\colon \mathbb{R}^{N_a} \to \mathbb{R}$ estime la d-ème coordonnée de $\boldsymbol{p}_\ell$ pour une entrée $\boldsymbol{\rho}_\ell$, comme le montre la figure 3.4.

Considérons un noyau reproduisant $\kappa : \mathbb{R}^{N_a} \times \mathbb{R}^{N_a} \to \mathbb{R}$ et dénotons par $\mathcal{H}$ son espace de Hilbert à noyau reproduisant (EHNR). Le problème revient à trouver la fonction $\psi_d(\cdot)$ qui minimise l'erreur entre la sortie estimée $\psi_d(\boldsymbol{\rho}_\ell)$ et la sortie désirée $p_{\ell,d}$, comme suit :

$$\mathcal{L}\big((p_{1,d}, \psi_d(\boldsymbol{\rho}_1)), \dots, (p_{N_p,d}, \psi_d(\boldsymbol{\rho}_{N_p}))\big) + \eta\, \Omega(\|\psi_d\|_{\mathcal{H}}^2), \tag{A.1}$$

où $\mathcal{L}$ et Ω sont respectivement une fonction coût et une fonction croissante strictement monotone sur $[0, \infty[$ à définir dans la suite, et $\|\cdot\|_{\mathcal{H}}$ représente la norme dans l'EHNR. Le second terme de (A.1) est un terme de régularisation, où $\eta > 0$ est un paramètre de régularisation qui permet d'atteindre un compromis entre la justesse de la solution et sa complexité. Dans ce qui suit, pour définir ψ_d et éventuellement $\boldsymbol{\psi}$, nous considérons la technique de kernel ridge regression (méthode des moindres carrés à noyaux), où la fonction coût est une fonction quadratique. Ensuite, nous considérons la technique de SVR (support vector machines pour la régression), où la fonction coût est la fonction charnière ("hinge loss"). Finalement, nous définissons $\boldsymbol{\psi}$ en utilisant la technique de vo-RLS (moindre carrés régularisée à sortie vectorielle), où une approche multi-tâches est considérée. Nous notons $\boldsymbol{P} = (\boldsymbol{p}_1^\top \dots \boldsymbol{p}_{N_p}^\top)^\top$ la matrice de taille $N_p \times \delta$, ayant $p_{\ell,d}$ pour son (ℓ, d)-ème élément et $\boldsymbol{p}_\ell$ pour son ℓ-ème ligne. Dans la suite, le vecteur $\boldsymbol{p}_\ell$ est noté par $\boldsymbol{P}_{\ell,*}$ et la d-ème colonne de $\boldsymbol{P}$ par $\boldsymbol{P}_{*,d}$.

A.2.2.1 Kernel ridge regression

Dans cette partie, nous définissons le modèle en utilisant la technique de kernel ridge regression [Saunders et al., 1998]. Dans ce cas, la fonction coût est la fonction quadratique et le terme de régularisation $\Omega(\|\psi_d\|_{\mathcal{H}}^2)$ est égal à $\|\psi_d\|_{\mathcal{H}}^2$. Nous obtenons ainsi le problème d'optimisation suivant :

$$\psi_d = \arg\min_{\underline{\psi}_d \in \mathcal{H}} \frac{1}{N_p} \sum_{\ell=1}^{N_p} (p_{\ell,d} - \underline{\psi}_d(\boldsymbol{\rho}_\ell))^2 + \eta \|\underline{\psi}_d\|_{\mathcal{H}}^2. \tag{A.2}$$

D'après le théorème de représentation [Kimeldorf and Wahba, 1971; Schölkopf et al., 2001a] (Section 2.3, page 30), la fonction $\psi_d(\cdot)$ admet une représentation de la forme :

$$\psi_d(\cdot) = \sum_{\ell=1}^{N_p} \alpha_{\ell,d}\, \kappa(\boldsymbol{\rho}_\ell, \cdot). \tag{A.3}$$

En injectant (A.3) dans (A.2), nous obtenons un problème dual en termes de $\alpha_{\ell,d}$. La solution de ce problème est donnée par $\boldsymbol{\alpha}_{*,d} = (\boldsymbol{K} + \eta N_p \mathbf{I})^{-1} \boldsymbol{P}_{*,d}$, où $\boldsymbol{\alpha}_{*,d}$ est le vecteur colonne dont le ℓ-ème élément est $\alpha_{\ell,d}$, $\mathbf{I}$ est la matrice identité de taille $N_p \times N_p$ et $\boldsymbol{K}$ est la matrice de Gram de taille $N_p \times N_p$ dont le (i,j)-ème élément est $\kappa(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j)$, pour $i, j \in \{1, ..., N_p\}$. L'expression de $\boldsymbol{\alpha}_{*,d}$ montre que la même matrice $(\boldsymbol{K} + \eta N_p \mathbf{I})$ doit être inversée pour estimer chaque coordonnée. Néanmoins, nous pouvons grouper les δ estimations, pour réduire la complexité du calcul, comme suit :

$$\boldsymbol{\alpha} = (\boldsymbol{K} + \eta N_p \mathbf{I})^{-1} \boldsymbol{P}, \tag{A.4}$$

où $\boldsymbol{\alpha}$ désigne la matrice dont le (ℓ, d)-ème élément est $\alpha_{\ell,d}$ et sa ℓ-ème ligne est notée $\boldsymbol{\alpha}_{\ell,*}$. Ensuite, en utilisant (A.3) et la définition de la fonction $\boldsymbol{\psi}(\cdot)$, nous définissons un modèle qui estime simultanément les δ coordonnées :

$$\boldsymbol{\psi}(\cdot) = \sum_{\ell=1}^{N_p} \boldsymbol{\alpha}_{\ell,*}\, \kappa(\boldsymbol{\rho}_\ell, \cdot). \tag{A.5}$$

Nous pouvons ajouter un biais $\boldsymbol{b}$ au modèle que nous venons de définir comme dans [Suykens and Vandewalle, 1999], tel que : $\boldsymbol{\psi}(\cdot) = \sum_{\ell=1}^{N_p} \boldsymbol{\alpha}_{\ell,*}\, \kappa(\boldsymbol{\rho}_\ell, \cdot) + \boldsymbol{b}$. Nous obtenons alors un système linéaire, où $\boldsymbol{\alpha}$ et $\boldsymbol{b}$ sont les inconnues à déterminer (voir la Sous-section 3.3.1, page 43 pour plus de détails).

Nous venons de considérer dans cette partie un terme de régularisation égal à $\|\psi_d\|^2_{\mathcal{H}}$. Toutefois, nous avons également considéré à la Sous-section 3.3.2 (page 45) le cas où le terme de régularisation est égal à $\|\boldsymbol{\alpha}_{*,d}\|^2$ au lieu de $\|\psi_d\|^2_{\mathcal{H}}$. Nous avons montré que ces deux régularisations sont directement liées ; ainsi, les solutions obtenues sont similaires. Se référer au Chapitre 3 pour plus de détails.

A.2.2.2 Support vector regression

Dans cette partie, nous considérons la technique ϵ-support vector regression (ϵ-SVR) [Vapnik, 1995] pour définir les modèles $\psi_d(\cdot)$. L'objectif de cette technique est de trouver $\psi_d(\boldsymbol{\rho}_\ell)$ ayant au maximum une déviation égale à ϵ de la sortie désirée $p_{\ell,d}$ ($|p_{\ell,d} - \psi_d(\boldsymbol{\rho}_\ell)| \leq \epsilon$), et cela pour tout l'ensemble d'apprentissage. La fonction coût est alors donnée par

$$\sum_{\ell=1}^{N_p} \max(0, |p_{\ell,d} - \psi_d(\boldsymbol{\rho}_\ell)| - \epsilon),$$

et le terme de régularisation par $\|\psi_d\|^2_{\mathcal{H}}$. Le problème d'optimisation est résolu plus facilement dans sa forme duale [Smola and Schölkopf, 2004], donnée en termes des multiplicateurs de Lagrange $\boldsymbol{\alpha}_{*,d}$ et $\widetilde{\boldsymbol{\alpha}}_{*,d}$ comme suit :

$$\max_{\boldsymbol{\alpha}_{*,d}, \widetilde{\boldsymbol{\alpha}}_{*,d}} \sum_{\ell=1}^{N_p} \widetilde{\alpha}_{\ell,d}\,(p_{\ell,d} - \epsilon) - \alpha_{\ell,d}\,(p_{\ell,d} + \epsilon) - \frac{1}{2}\sum_{\ell=1}^{N_p}\sum_{j=1}^{N_p}(\widetilde{\alpha}_{\ell,d} - \alpha_{\ell,d})(\widetilde{\alpha}_{j,d} - \alpha_{j,d})\kappa(\boldsymbol{\rho}_\ell, \boldsymbol{\rho}_j),$$

avec les contraintes suivantes :

$$0 \leq \alpha_{\ell,d}, \widetilde{\alpha}_{\ell,d} \leq \frac{1}{2\eta N_p}, \ell \in 1, \ldots, N_p \quad \text{and} \quad \sum_{\ell=1}^{N_p}(\alpha_{\ell,d} - \widetilde{\alpha}_{\ell,d}) = 0. \tag{A.6}$$

En résolvant ce problème sous les contraintes (A.6), nous obtenons les multiplicateurs de Lagrange, et la fonction de régression est donnée par :

$$\psi_d(\cdot) = \sum_{\ell=1}^{N_p} (\alpha_{\ell,d} - \widetilde{\alpha}_{\ell,d})\, \kappa(\boldsymbol{\rho}_\ell, \cdot) + b. \tag{A.7}$$

Se référer à la Sous-section 3.3.3 (page 47) pour le calcul du terme b.

A.2.2.3 Vector-output regularized least squares

Dans les algorithmes précédents, nous posons, séparément, δ problèmes d'optimisation pour définir δ modèles $\psi_d(\cdot)$, un modèle par coordonnée. Dans cette partie, un problème d'optimisation unique est utilisé pour estimer simultanément toutes les δ coordonnées. Ainsi, nous déterminons le modèle $\boldsymbol{\psi}(\cdot)$ en explorant l'apprentissage statistique multi-tâches [Micchelli and Pontil, 2005; Evgeniou et al., 2005]. Pour cela, nous utilisons l'algorithme vo-RLS, qui est un algorithme d'apprentissage multi-tâches décrit dans [Honeine et al., 2013]. Par conséquent, $\boldsymbol{\psi}(\cdot)$ sera représentée comme suit :

$$\boldsymbol{\psi}(\cdot) = \sum_{\ell=1}^{N_p} \beta_\ell\, \boldsymbol{P}_{\ell,*}\, \kappa(\boldsymbol{\rho}_\ell, \cdot),$$

où β_ℓ, $\ell \in \{1, \ldots, N_p\}$, sont des paramètres à définir, et le problème d'optimisation est donné par :

$$\min_{\underline{\boldsymbol{\psi}}, \underline{\boldsymbol{\beta}}} \frac{1}{N_p} \sum_{\ell=1}^{N_p} \|\boldsymbol{P}_{\ell,*} - \underline{\boldsymbol{\psi}}(\boldsymbol{\rho}_\ell)\|^2 + \eta \|\underline{\boldsymbol{\beta}}\|^2, \tag{A.8}$$

où $\boldsymbol{\beta} = \begin{pmatrix}\beta_1 \ldots \beta_{N_p}\end{pmatrix}^\top$. En remplaçant l'expression de $\boldsymbol{\psi}(\cdot)$ dans (A.8), nous obtenons un problème d'optimisation en termes de $\boldsymbol{\beta}$ dont la solution est donnée par :

$$\boldsymbol{\beta} = (\boldsymbol{G} + \eta N_p \mathbf{I})^{-1} \boldsymbol{v},$$

où $\boldsymbol{G}$ est la $N_p \times N_p$ matrice, ayant $\boldsymbol{P}_{j,*}\, \boldsymbol{P}_{k,*}^\top \sum_{i=1}^{N_p} \kappa(\boldsymbol{\rho}_j, \boldsymbol{\rho}_i)\, \kappa(\boldsymbol{\rho}_k, \boldsymbol{\rho}_i)$ pour (j,k)-ème élément et $\boldsymbol{v}$ est le vecteur de taille $N_p \times 1$, ayant $\sum_{k=1}^{N_p} \boldsymbol{P}_{j,*}\, \boldsymbol{P}_{k,*}^\top \kappa(\boldsymbol{\rho}_j, \boldsymbol{\rho}_k)$ pour j-ème élément. Il est intéressant de noter que N_p inconnues sont à estimer dans cet algorithme, tandis que $N_p \times \delta$ inconnues sont à estimer pour les algorithmes précédents. Cependant, pour localiser un nœud en utilisant le modèle vo-RLS en temps réel, nous avons besoin des mesures de référence RSSIs, ainsi que des positions de référence, alors que pour les

autres algorithmes, seules les mesures de référence RSSIs sont exigées.

A.2.3 Extension au cas d'une topologie hybride

Dans cette sous-section, nous proposons un algorithme clusterisé de localisation qui permet d'obtenir des estimations locales en parallélisme correspondant à chaque cluster, combinées par la suite pour l'obtention d'une estimation globale. Par rapport à une stratégie centralisée, l'approche clusterisée est plus robuste aux défaillances, puisque plusieurs têtes de cluster sont engagées. Ainsi, pour faire le passage vers une approche clusterisée, nous partitionnons la région d'intérêt en Z clusters distincts. Chaque cluster a son propre centre de fusion ou tête de cluster capable de traiter les données et d'échanger des informations avec les capteurs (voir figure 3.5). En conséquence, notre objectif est de définir pour chaque cluster z, $z \in \{1, ..., Z\}$, un modèle local $\boldsymbol{\psi}^{(z)}(\cdot)$ en utilisant l'un des algorithmes d'apprentissage susmentionnés. Les estimations locales seront ensuite combinées pour donner une estimation globale de la position d'un nœud. Pour plus de détails, se référer à la Sous-section 3.4 (page 50).

A.2.4 Analyse et simulations

Dans cette sous-section, nous évaluons les performances de la méthode proposée pour les différents algorithmes d'apprentissage. Ainsi, considérons des RSSIs générées à l'aide du modèle de propagation logarithmique [Medeisis and Kajackas, 2000] :

$$\rho_{\boldsymbol{a}_i,\boldsymbol{p}_\ell} = \rho_0 - 10\, n_P \, \log_{10} \|\boldsymbol{a}_i - \boldsymbol{p}_\ell\| + \varepsilon_{i,\ell}, \tag{A.9}$$

où $\rho_{\boldsymbol{a}_i,\boldsymbol{p}_\ell}$ est la puissance du signal envoyé par l'ancre à la position $\boldsymbol{a}_i$ et reçu par un capteur à la position $\boldsymbol{p}_\ell$, ρ_0 la puissance mesurée à une distance de référence $d_0 = 1$ m, n_{P} le paramètre de perte qui dépend du type de canal de transmission et qui est pris égal à 4, $\|\boldsymbol{a}_i - \boldsymbol{p}_\ell\|$ est la distance euclidienne séparant $\boldsymbol{p}_\ell$ de $\boldsymbol{a}_i$. Finalement, $\varepsilon_{i,\ell}$ est un bruit additif gaussien ayant un écart-type de $\sigma_{\boldsymbol{\rho}}$. Notons que nous avons également analysé les performances de la méthode sur des données réelles ; voir la Sous-section 3.5 (page 53) pour plus de détails.

A.2.4.1 Illustration générale

Nous considérons une zone de surveillance de 100 m × 100 m, où nous générons de manière uniforme 16 ancres fixes et 100 positions de référence. Ensuite, nous générons une trajectoire comme le montre la figure 3.6. Afin d'estimer la position d'un nœud qui se déplace le long de la trajectoire, nous appliquons la méthode proposée pour les

différents algorithmes d'apprentissage (en considérant le noyau gaussien). Les tableaux 3.2 et 3.3 montrent respectivement les erreurs d'estimation (en mètres) pour des données non bruitées et dans le cas où $\sigma_{\boldsymbol{\rho}}$ =1 dB. Nous pouvons voir que la plus faible erreur est obtenue avec la technique de kernel ridge regression pour les données bruitées et non bruitées. De plus, cette technique est la plus performante en termes de complexité de calcul et en termes de mémoire comme le montre les tableaux 3.4 à 3.7. Pour toutes ces raisons, nous considérons seulement la technique de kernel ridge regression pour les études qui suivent.

A.2.4.2 Impact des paramètres N_a et N_p

Dans cette partie, nous étudions l'impact du nombre d'ancres et du nombre de positions de référence sur les performances de la méthode. Pour cela, nous considérons le scénario de la figure 3.6 avec des données non bruitées. Nous fixons N_p à 100 et varions N_a tel que $N_a = 1^2, 2^2, \ldots, 20^2$. La figure 3.7 montre l'évolution de l'erreur d'estimation en fonction de N_a. Ensuite, pour la figure 3.8, nous fixons N_a à 16 et varions le nombre de positions de référence tel que $N_p = 5^2, 6^2, \ldots, 25^2$. En comparant les résultats obtenus, nous pouvons remarquer que l'erreur d'estimation est plus affectée par la variation de N_p que par la variation de N_a. En effet, l'augmentation du nombre de positions de référence permet une meilleure couverture et une meilleure connaissance de l'environnement, ce qui explique l'amélioration des résultats. Par exemple, pour $N_p = 25^2 = 625$ et $N_a = 16$, nous obtenons une faible erreur d'estimation de 0,0045 m, mais avec une augmentation significative de la complexité de l'algorithme, par rapport au cas où $N_p = 25$ par exemple. Par conséquent, en fonction des contraintes du système, un compromis doit être trouvé entre la précision de l'algorithme et la complexité du calcul.

Nous avons également analysé les performances de la méthode pour une distribution aléatoire des ancres et des positions de référence. Nous avons observé une légère augmentation de l'erreur (tableaux 3.8 et 3.9). Ceci peut être expliqué par le fait qu'une distribution uniforme permet une meilleure couverture de la région d'intérêt, tandis qu'une distribution aléatoire ne garantit pas toujours une bonne couverture. Néanmoins, les résultats sont toujours satisfaisants, et les distributions aléatoires permettent d'avoir une localisation assez exacte quand les distributions uniformes ne sont pas applicables.

A.2.4.3 Comparaison à d'autres méthodes de localisation

L'objectif de ce paragraphe est de fournir une comparaison de l'approche proposée à l'égard de deux techniques de localisation bien connues : la localisation basée sur les connectivités [Y. Shang and Fromherz, 2003] et la localisation utilisant l'algorithme des K

plus proches voisins (weighted K-nearest neighbor algorithms, WKNN) pour différentes fonctions de pondération [Kushki et al., 2007; Koyuncu and Yang, 2011]. Les différentes fonctions de pondération sont données au tableau 3.12. Le tableau 3.11 montre les erreurs d'estimation pour différents nombres d'ancres pour la localisation par connectivités et la méthode proposée. Les résultats montrent que la méthode proposée est plus précise que la méthode par connectivités. De plus, elle donne de meilleurs résultats en termes de complexité de calcul. Les tableaux 3.13 et 3.14 montrent respectivement les erreurs d'estimation dans le cas de données non bruitées et pour $\sigma_{\rho} = 1$ dB, pour la méthode proposée et les méthodes dans [Kushki et al., 2007] et [Koyuncu and Yang, 2011] avec différents poids. Là aussi, la méthode proposée donne les résultats les plus précis.

A.2.5 Conclusion

Cette section propose une nouvelle technique de localisation utilisant la technique de fingerprinting et les méthodes à noyaux, avec différents algorithmes d'apprentissage. Une version clusterisée de cette approche de localisation a également été proposée. Les simulations montrent que nos approches permettent une localisation précise dans les deux cas de données réelles et simulées. En outre, elles sont plus performantes que d'autres techniques de localisation (WKNN et la localisation par connectivités). Cependant, nous montrerons dans ce qui suit qu'elles sont vulnérables face à la présence de bruit additif élevé sur les RSSIs. Par conséquent, dans la section suivante, nous introduisons de nouvelles améliorations à notre algorithme, et cela en y intégrant des informations supplémentaires de mobilité pour corriger la trajectoire estimée.

A.3 Suivi de cibles basé sur les positions

Comme nous avons expliqué à la Section A.1, le suivi de cibles est une application très populaire des RCSF [Li et al., 2002; Dallil et al., 2013]. Le suivi de cibles consiste à estimer de manière récursive les états de la cible, à savoir, ses positions et ses vitesses [Svensson, 2010]. En utilisant un modèle d'état qui décrit le mouvement de la cible, nous pouvons d'abord prédire son état futur à partir de son état actuel. Ensuite, l'état prédit est mis à jour en utilisant les observations du réseau, à savoir, un modèle d'observation. Plusieurs filtres peuvent être utilisés pour combiner les informations d'inertie et les observations ; à noter que, normalement, le choix du filtre se fait en fonction du type des observations. Par exemple, le filtre de Kalman (KF) [Kalman, 1960; Welch and Bishop, 2001] peut être utilisé dans le cas d'un modèle d'observation linéaire. Dans le cas de la non-linéarité, le filtre de Kalman étendu (EKF) [Julier and Uhlmann, 1997] et le filtre de

Kalman sans parfum (UKF) [Julier and Uhlmann, 2004] peuvent être utilisés. Cependant, de telles approches effectuent des linéarisations et des approximations conduisant à une performance sous-optimale et parfois à la divergence du filtre [UmaMageswari et al., 2012]. Le filtre particulaire (PF) [Branko Ristic, 2004] est également utilisé pour le suivi de cibles. Un tel filtre a plus de potentiel que le filtre de Kalman dans le cas de bruits non gaussiens et de modèles non linéaires [Gustafsson et al., 2002] ; cependant, la génération d'échantillons et l'étape de ré-échantillonnage rendent les algorithmes employant ce filtre plus complexes en termes de calculs que le filtre de Kalman. Plusieurs techniques de suivi utilisant des informations d'inertie ont été proposé dans la littérature. En plus des mesures de RSSI, ces techniques emploient un modèle d'état pour affiner l'estimation de la position grâce à sa position antécédente. Par exemple, un modèle d'état du premier ordre est utilisé avec un filtre de Kalman dans [Li and Li, 2012], et avec un filtre particulaire dans [Dias and Bruno, 2013]. Une autre technique de suivi est présentée dans [Chan et al., 2009], où les auteurs estiment les positions d'une cible à l'aide de l'algorithme WKNN. Ensuite, ces estimations sont combinées aux données inertielles au moyen d'un filtre de Kalman. Toutefois, ces modèles ne sont fiables que pour des cibles ayant des vitesses ou accélérations qui varient légèrement.

Dans cette section, nous effectuons le suivi de cible en utilisant la technique de localisation introduite à la section précédente. Ainsi, en utilisant la technique de fingerprinting et les méthodes à noyaux, nous définissons un modèle qui prend en entrée les puissances RSSIs des signaux échangés entre la cible et des capteurs fixes dans le réseau, et donne en sortie une première estimation de la position de la cible. Ensuite, nous combinons cette première estimation aux données inertielles de la cible par l'intermédiaire d'un filtre de Kalman pour obtenir une estimation plus affinée de la position de la cible.

A.3.1 Problématique

L'approche proposée est centralisée ; toutefois, le passage à une approche hybride peut se faire facilement en se basant sur l'explication fournie à la section précédente. Nous considérons un RCSF composé de N_s capteurs fixes ayant des positions connues, notées $\boldsymbol{s}_i = (s_{i,1} \dots s_{i,\delta})^\top$, $i \in \{1, \dots, N_s\}$. L'objectif est de suivre une cible ayant une position inconnue notée par $\boldsymbol{x}(k) = (x_1(k) \dots x_\delta(k))^\top$ à un pas de temps k. La cible est supposée être équipée d'un accéléromètre qui mesure ses δ accélérations instantanées. Elle est supposée être fixe au début du suivi à une position connue $\boldsymbol{x}(0)$ et elle a donc une vitesse initiale $\boldsymbol{\nu}(0)$ nulle. Dans la suite, $\boldsymbol{\nu}(k)$ désigne le vecteur de vitesse de la cible à l'instant k et $\boldsymbol{\gamma}(k)$ désigne son vecteur d'accélération mesuré à l'instant k.

Tout d'abord, nous devons définir un modèle d'état caractérisant la relation entre la position actuelle $\boldsymbol{x}(k)$ de la cible et sa position précédente $\boldsymbol{x}(k-1)$. Nous décrivons à la Sous-section 4.2.2 (page 73) quatre modèles d'état. Le modèle du premier ordre suppose d'abord que les vitesses sont constantes tout au long du suivi de cibles, ainsi les accélérations sont-elles considérées comme nulles. Nous proposons ensuite une version hybride de ce modèle qui suppose d'abord que les accélérations sont constantes entre deux pas de temps consécutifs, $k-1$ et k, pour le calcul des vitesses à l'instant k. Ensuite, une approximation supplémentaire est effectuée en supposant que les vitesses sont constantes entre $k-1$ et k. Il est évident que ces deux modèles sont seulement adaptés aux cibles ayant des vitesses légèrement variables ($\boldsymbol{\gamma} \simeq \mathbf{0}$). En raison de ces deux hypothèses, les performances de ces modèles se dégradent très rapidement pour des cibles plus actives. Le modèle du second ordre suppose que les accélérations sont constantes entre deux instants consécutifs. Si les accélérations varient légèrement, ce modèle conduit à des estimations précises. Cependant, avec des changements brusques d'accélérations, les estimations de la position de la cible pourraient être sensiblement déviées des vraies positions en raison des erreurs cumulatives du modèle. Pour plus de détails sur ces deux modèles, se référer au Chapitre 4. Finalement, ayant les accélérations aux instants $k-1$ et k, nous considérons un modèle d'état du troisième ordre qui est plus adapté que les autres modèles au cas où les accélérations de la cible changent brusquement. Ce modèle suppose que les accélérations varient linéairement entre deux pas de temps consécutifs $k-1$ et k, avec une pente égale à $\frac{\boldsymbol{\gamma}(k)-\boldsymbol{\gamma}(k-1)}{\Delta t}$, où Δt est l'intervalle de temps séparant deux pas de temps consécutifs. Par conséquent, nous pouvons estimer de manière récursive le vecteur vitesse $\boldsymbol{\nu}(k)$ puis le vecteur position $\boldsymbol{x}(k)$ comme suit :

$$\begin{cases} \boldsymbol{\nu}(k) = \boldsymbol{\nu}(k-1) + \boldsymbol{\gamma}(k-1)\,\Delta t + \frac{\boldsymbol{\gamma}(k)-\boldsymbol{\gamma}(k-1)}{\Delta t}\,\frac{\Delta t^2}{2}, \\ \boldsymbol{x}(k) = \boldsymbol{x}(k-1) + \boldsymbol{\nu}(k-1)\,\Delta t + \boldsymbol{\gamma}(k-1)\,\frac{\Delta t^2}{2} + \frac{\boldsymbol{\gamma}(k)-\boldsymbol{\gamma}(k-1)}{\Delta t}\,\frac{\Delta t^3}{6}. \end{cases}$$

Nous notons aussi que dans cette section, nous considérons que la cible ne peut pas tourner autour d'elle. Cependant, pendant son mouvement, la cible est susceptible de tourner et le système de coordonnées où les accélérations sont mesurées peut devenir différent du système de coordonnées global du réseau. Nous proposons d'équiper la cible d'un gyroscope qui mesure ses orientations ; une solution détaillée est proposée à la Sous-section 4.2.2 (page 73) pour surmonter ce problème.

D'une autre part, pendant son mouvement, la cible mesure en temps réel le vecteur des RSSIs des signaux reçus des N_s capteurs, noté par $\boldsymbol{\rho}(k)$, et l'envoie au centre de fusion. Comme à la section précédente, nous proposons d'utiliser un modèle $\boldsymbol{\psi}\colon \mathbb{R}^{N_s} \to \mathbb{R}^{\delta}$ pour estimer la position de la cible. La définition de ce modèle se fait aussi en utilisant la technique de fingerprinting. Pour cela, nous générons de façon uniforme ou aléatoire N_p

positions de références, notées $\boldsymbol{p}_\ell$, $\ell \in \{1, \ldots, N_p\}$. En plaçant un capteur pour mesurer les RSSIs à chaque position de référence, nous obtenons N_p couples $(\boldsymbol{\rho}_\ell, \boldsymbol{p}_\ell)$ à utiliser pour l'apprentissage du modèle $\boldsymbol{\psi}$. Notons que n'importe quel algorithme susmentionné peut être utilisé pour l'apprentissage. Une fois le modèle défini, nous pouvons estimer la position de la cible à un instant k comme suit :

$$\boldsymbol{z}(k) = \boldsymbol{\psi}(\boldsymbol{\rho}(k)). \tag{A.10}$$

La quantité estimée $\boldsymbol{z}(k)$ est alors considérée comme une observation à chaque instant k. Le modèle d'observation obtenu est clairement linéaire ; ainsi, nous utiliserons dans la suite un filtre de Kalman pour résoudre le problème de suivi.

A.3.2 Solution à l'aide du filtre de Kalman

Dans cette sous-section, nous combinons les données inertielles aux observations par l'intermédiaire d'un filtre de Kalman. L'idée de ce filtre est de prédire l'état inconnu en utilisant les équations de mobilité, employant les accélérations, puis de le corriger en utilisant les observations, définies ici par l'estimation de la position obtenue en utilisant les mesures de RSSIs et le modèle $\boldsymbol{\psi}$. Nous désignons par $\boldsymbol{X}(k)$ l'état inconnu de la cible à un instant k. Or comme les positions et les vitesses de la cible sont prédites à la fois dans les équations de mobilité données ci-dessus, l'état $\boldsymbol{X}(k)$ sera un vecteur de taille $2\delta \times 1$, tel que $\boldsymbol{X}(k) = (\boldsymbol{x}(k)\ \boldsymbol{\nu}(k))^\top$. Les équations du filtre seront données par :

$$\begin{cases} \boldsymbol{X}(k) = \boldsymbol{A}\ \boldsymbol{X}(k-1) + \boldsymbol{B}(k) + \boldsymbol{\epsilon}(k), \\ \boldsymbol{z}(k)^\top = \boldsymbol{C}\ \boldsymbol{X}(k) + \boldsymbol{n}(k), \end{cases}$$

où la quantité $\boldsymbol{\epsilon}(k) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{V})$ est l'erreur du modèle d'état ayant une distribution normale à moyenne nulle et une matrice de covariance $\boldsymbol{V}$ de taille $2\delta \times 2\delta$. La quantité $\boldsymbol{n}(k) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R})$ est l'erreur du modèle d'observation ayant une distribution normale à moyenne nulle et une matrice de covariance $\boldsymbol{R}$ de taille $\delta \times \delta$. La matrice $\boldsymbol{C}$ est la matrice d'observation qui lie $\boldsymbol{X}(k)$ à la position estimée $\boldsymbol{z}(k)$. La matrice de transition $\boldsymbol{A}$ et le vecteur de contrôle $\boldsymbol{B}(k)$ sont définis selon le choix de l'ordre du modèle de mobilité. Par exemple, pour le modèle du troisième ordre, nous avons :

$$\boldsymbol{A} = \begin{pmatrix} \mathbf{I}_\delta & \Delta t \mathbf{I}_\delta \\ \mathbf{0}_{\delta\times\delta} & \mathbf{I}_\delta \end{pmatrix} \quad \text{et} \quad \boldsymbol{B}(k) = \begin{pmatrix} \boldsymbol{\gamma}(k-1)^\top \frac{\Delta t^2}{3} \\ \boldsymbol{\gamma}(k-1)^\top \frac{\Delta t}{2} \end{pmatrix} + \begin{pmatrix} \boldsymbol{\gamma}(k)^\top \frac{\Delta t^2}{6} \\ \boldsymbol{\gamma}(k)^\top \frac{\Delta t}{2} \end{pmatrix},$$

où $\mathbf{I}_\delta$ est la matrice identité de taille $\delta \times \delta$ et $\mathbf{0}_{\delta\times\delta}$ est la matrice de zéros de taille $\delta \times \delta$.

La résolution du problème à l'aide du filtre de Kalman est composée de deux phases. À la première phase, le filtre prédit l'état inconnu à partir de l'équation d'état comme suit :

$$\widehat{\boldsymbol{X}}^{-}(k) = \boldsymbol{A}\,\widehat{\boldsymbol{X}}(k-1) + \boldsymbol{B}(k), \tag{A.11}$$

avec $\widehat{\boldsymbol{X}}(k-1)$ l'état prédit par Kalman à l'instant $k-1$ et $\widehat{\boldsymbol{X}}(0)$ supposé connu. Ensuite, la prédiction de la matrice de covariance de l'estimation est mise à jour telle que $\boldsymbol{T}^{-}(k) = \boldsymbol{A}\,\boldsymbol{T}(k-1)\boldsymbol{A}^{\top} + \boldsymbol{Q}(k)$, où $\boldsymbol{T}(k-1)$ est la matrice de covariance à $k-1$ et $\boldsymbol{T}(0)$ est nulle puisque l'état initial est connu. La matrice $\boldsymbol{Q}(k)$ est la matrice de covariance de $\boldsymbol{X}(k)$ sachant $\boldsymbol{X}(k-1)$; pour plus de détails, se référer à la Section 4.3, page 79. Finalement, les quantités prédites $\widehat{\boldsymbol{X}}^{-}(k)$ et $\boldsymbol{T}^{-}(k)$ sont corrigées en utilisant le modèle d'observation, notamment :

$$\widehat{\boldsymbol{X}}(k) = \widehat{\boldsymbol{X}}^{-}(k) + \boldsymbol{G}_{\mathrm{KF}}(k)\,(\boldsymbol{z}(k)^{\top} - \boldsymbol{C}\widehat{\boldsymbol{X}}^{-}(k)) \tag{A.12}$$

$$\boldsymbol{T}(k) = (\mathbf{I}_{2\delta} - \boldsymbol{G}_{\mathrm{KF}}(k)\,\boldsymbol{C})\,\boldsymbol{T}^{-}(k), \tag{A.13}$$

où $\boldsymbol{G}_{\mathrm{KF}}(k)$ est le gain de Kalman donné par $\boldsymbol{G}_{\mathrm{KF}}(k) = \boldsymbol{T}^{-}(k)\,\boldsymbol{C}^{\top}\,(\boldsymbol{C}\,\boldsymbol{T}^{-}(k)\,\boldsymbol{C}^{\top} + \boldsymbol{R})^{-1}$.

A.3.3 Analyse et simulations

Cette sous-section évalue les performances de la méthode de suivi de cibles que nous venons de proposer. Pour cela, nous considérons 16 capteurs fixes et 100 positions de référence distribués dans une région de 100 m × 100 m. Nous nous intéressons au suivi d'une cible qui se déplace dans cette région. Les mesures de RSSIs des signaux échangés entre la cible et les capteurs fixes sont générées en utilisant le modèle de propagation logarithmique donné par (A.9), avec les mêmes paramètres que la section précédente. Pour la définition du modèle $\boldsymbol{\psi}$, nous utilisons deux des techniques d'apprentissage décrites à la Section A.2 : la technique de kernel ridge regression (KRR) et la technique de vo-RLS (avec le noyau gaussien).

A.3.3.1 Illustration générale

Afin d'évaluer la précision de la méthode de suivi proposée pour différents types de cibles (cibles en mouvement monotone, puis cibles plus actives), nous considérons trois trajectoires différentes de 100 points avec $\Delta t = 1$s. Pour la trajectoire illustrée à la figure 4.3, les accélérations sont nulles donc les vitesses sont constantes. Quant à la deuxième et troisième trajectoires représentées respectivement aux figures 4.4 et 4.5, leurs accélérations sont représentées à la figure 4.6. Nous pouvons voir que les accélérations de la troisième trajectoire ont plus de variations que les accélérations de

la deuxième trajectoire. Ensuite, les coordonnées de la cible sont obtenues en intégrant deux fois les accélérations. L'écart-type du bruit sur les puissances RSSIs $\sigma_{\boldsymbol{\rho}}$ est égal à 1 dB et les deux composantes de l'écart-type sur les accélérations $\boldsymbol{\sigma_\gamma}$ sont égales à 0,01 m/s^2. Le tableau 4.2 montre les erreurs d'estimations en mètres pour les trois trajectoires et les quatre modèles d'état, en utilisant les techniques de KRR et de vo-RLS dans l'apprentissage. Les quatre modèles donnent presque les mêmes résultats pour la première trajectoire. Cependant, pour la deuxième et la troisième trajectoire, le premier modèle donne la plus grande erreur d'estimation. Ce résultat peut être expliqué par le fait que ce modèle suppose que les vitesses sont constantes, alors qu'elles ne le sont pas dans le cas de ces deux trajectoires. Pour la troisième trajectoire, l'erreur la plus faible est obtenue en utilisant le modèle d'état du troisième ordre. Ce résultat est prévu puisque les accélérations dans cette trajectoire présentent de fortes variations, et comme nous avons expliqué, le modèle d'état du troisième ordre est le mieux adapté pour de telles cibles. Par conséquent, dans le reste de cette étude, nous considérons uniquement ce modèle.

A.3.3.2 Impact des différents paramètres

Dans cette partie, nous étudions l'impact de différents paramètres sur notre méthode de suivi, notamment l'impact de $\sigma_{\boldsymbol{\rho}}$, $\boldsymbol{\sigma_\gamma}$, des capteurs fixes et des positions de références. Commençons par fixer $\sigma_{\boldsymbol{\rho}}$ à 5% de l'écart-type des mesures de RSSIs ($\sigma_{\boldsymbol{\rho}}$ =0,54 dB) et varions $\boldsymbol{\sigma_\gamma}$ de 1% à 10% de l'écart-type des accélérations. La figure 4.7 montre l'impact de la variation de $\boldsymbol{\sigma_\gamma}$ sur l'erreur d'estimation. Nous pouvons voir que les résultats obtenus en utilisant la localisation avec les techniques de ridge régression et de vo-RLS de la section précédente sont indépendants du bruit d'accélération, alors que les estimations en utilisant uniquement les données inertielles sont fortement affectées par les variations de $\boldsymbol{\sigma_\gamma}$. Nous remarquons aussi que la technique KRR combinée avec le filtre de Kalman donne les meilleurs résultats. En effet, le filtre affine les résultats, et l'erreur obtenue est toujours plus petite que l'erreur dans le cas de la technique KRR seule. Nous fixons maintenant $\boldsymbol{\sigma_\gamma}$ à 1% de l'écart type des accélérations et nous considérons plusieurs pourcentages de l'écart-type des mesures de RSSIs, allant de 0% à 50% (σ_ρ varie de 0 à 5,40 dBm). La figure 4.8 montre l'impact de la variation de $\sigma_{\boldsymbol{\rho}}$ sur l'erreur d'estimation. Nous remarquons la localisation avec les techniques de KRR et de vo-RLS sont fortement affectées par les variations de bruit, car elles utilisent ces mesures de RSSIs pour l'estimation, alors que le suivi à partir des données inertielles seules n'est pas affecté par le bruit sur les RSSIs. Toutefois, l'approche de suivi proposée surpasse la technique de suivi utilisant seulement les accélérations. Il est intéressant aussi de voir

que le filtre de Kalman permet d'avoir au final une erreur constante face à l'augmentation du bruit.

Quant à l'impact des capteurs fixes et des positions de référence, nous remarquons tout d'abord qu'en considérant une distribution aléatoire dans le réseau, à la place d'une distribution uniforme, les erreurs d'estimation augmentent mais les résultats restent satisfaisants (voir tableau 4.5). Quant à l'impact du nombre des capteurs N_s et du nombre des positons de référence N_p, nous remarquons que l'augmentation de l'un ou de l'autre permet d'obtenir une meilleure erreur d'estimation, comme le montrent les figures 4.9 et 4.10. En effet, comme nous avons expliqué précédemment, une distribution uniforme ou un nombre plus élevé de capteurs ou de position de référence permettent de mieux couvrir la région d'intérêt et donc permettent une meilleure connaissance de l'environnement. Cependant, l'augmentation du nombre de capteurs fixes augmente le coût total en matériel, alors que l'augmentation du nombre de positions de référence induit une augmentation significative de la complexité du calcul.

A.3.3.3 Comparaison à d'autres méthodes de suivi de cibles

La dernière étape de cette analyse consiste à comparer l'approche proposée à d'autres méthodes de suivi de cibles. La première méthode considérée pour la comparaison est proposée par Chan et al. [2009] qui utilisent un filtre de Kalman pour corriger des estimations obtenues avec la méthode de WKNN. La seconde méthode est proposée par les auteurs de [Dias and Bruno, 2012], où un filtre particulaire est utilisé pour le suivi. Notons que, pour les comparaisons, nous choisissons les mêmes paramètres (N_s, N_p, K) que ceux proposés par les auteurs. Les résultats obtenus sont montrés au tableau 4.6. Il est clair que notre méthode proposée donne les meilleurs résultats, même dans le cas où un modèle d'état du second ordre est utilisé.

A.3.4 Conclusion

Nous avons proposé dans cette section une nouvelle méthode pour le suivi de cibles dans les RCSF en combinant l'apprentissage par noyaux et le filtrage de Kalman. Les résultats des simulations ont montré que la méthode proposée permet une estimation précise et exacte de la position de la cible tout en étant robuste face aux bruits affectant les accélérations et les RSSIs. De plus, elle surpasse deux méthodes récentes de suivi de cibles qui utilisent un filtre de Kalman et un filtre particulaire. Cependant, la principale contrainte de la méthode proposée est que les capteurs doivent rester fixes au cours du processus de suivi, ce qui peut être contraignant pour certaines applications. La section suivante présente une solution au problème de stationnarité des capteurs en introduisant une nouvelle méthode de suivi indépendante des positions de ces capteurs.

A.4 Suivi de cibles basé sur les distances

Dans les Sections A.2 et A.3, nous avons profité des méthodes à noyaux pour définir un modèle qui prend en entrée les RSSIs des signaux échangés entre une cible en mouvement et des capteurs fixes dans le réseau, et qui donne en sortie la position de la cible. Dans cette section, au lieu d'utiliser ce modèle de position, nous proposons deux nouveaux modèles, définis aussi dans le cadre des méthodes à noyaux, qui permettent d'estimer les distances séparant la cible des capteurs. En raison de la relation logarithmique entre les RSSIs et les distances, l'utilisation de ces modèles de distance est susceptible de donner de meilleurs résultats que les modèles de position. De plus, nous montrerons dans la suite que l'utilisation des modèles de distance permet aux capteurs de se déplacer dans le réseau, sans affecter le processus de suivi. Néanmoins, l'estimation de la distance à l'aide des mesures de RSSIs présente un défi majeur, puisque celles-ci sont sensibles à la présence du bruit et des interférences. Un modèle populaire pour caractériser la relation RSSI/distance est le modèle de propagation logarithmique susmentionné [Medeisis and Kajackas, 2000; Zanella and Bardella, 2014]. Ce modèle est largement utilisé en raison de sa simplicité, malgré ses limitations et le fait qu'il ne prend pas en compte les caractéristiques du milieu, tels que les murs et les planchers. Différents modèles ont été proposés pour remédier à ce problème, comme des versions modifiées de ce modèle [Seidel and Rappaport, 1992; Sandeep et al., 2008], dans lesquels les atténuations dues aux planchers et les murs sont explicitement incluses. D'autres modèles proposés par [Wang et al., 2003; Yang and Chen, 2009] déterminent une relation mathématique entre les RSSIs et les distances, sans prendre en considération les propriétés physiques. À cette fin, un modèle empirique est estimé par la régression polynomiale. Selon les applications et l'environnement considéré, plusieurs autres modèles peuvent également être trouvés dans la littérature [Neskovic et al., 2000; Andrade and Hoefel, 2010].

Une fois que les distances sont estimées à partir des RSSIs, elles sont combinées en utilisant par exemple la trilatération [Manolakis, 1996] pour localiser la cible (voir figure 1.4, page 11). Nous pouvons également utiliser des méthodes basées sur le filtrage qui aident à lisser la trajectoire de la cible et donc à réduire l'erreur d'estimation. Par exemple, une approche de suivi utilisant le filtre de Kalman étendu avec les distances et les accélérations est proposée dans [Shareef and Zhu, 2009; Correa et al., 2014]. Cependant, comme nous l'avons expliqué précédemment, la linéarisation et les approximations induites par ce filtre pourraient conduire à une estimation inexacte des positions [Uma-Mageswari et al., 2012]. Le filtre particulaire peut également être utilisé pour le suivi. Les méthodes de suivi décrites dans [Wang et al., 2007; Farmani et al., 2012] profitent d'un tel filtre, en utilisant le mouvement de la cible et les distances.

Dans cette section, nous introduisons deux modèles originaux qui permettent d'estimer les distances séparant les capteurs à l'aide des RSSIs. Le premier modèle est non-paramétrique, et donc ne nécessite aucune connaissance préalable des propriétés physiques des signaux. Le second modèle est semi-paramétrique, combinant le modèle de propagation logarithmique avec une composante non linéaire additionnelle, estimée par les méthodes à noyaux. Les deux modèles prennent en entrée la puissance du signal échangé entre deux capteurs et donnent en sortie la distance qui les sépare. Ensuite, en se basant sur les distances estimées séparant la cible des capteurs à positions connues, nous proposons deux nouvelles méthodes centralisées pour localiser la cible. La première combine les distances estimées avec l'information sur les accélérations de la cible à l'aide d'un filtre de Kalman. La seconde permet de suivre la cible à l'aide d'un filtre particulaire, en utilisant des accélérations et des distances estimées.

A.4.1 Problématique

Nous considérons N_s capteurs ayant des positions connues, notées $\boldsymbol{s}_i = (s_{i,1} \dots s_{i,\delta})^\top$, $i \in \{1, \dots, N_s\}$. L'objectif est de définir les N_s modèles χ_i, tels que $\chi_i \colon \mathbb{R} \to \mathbb{R}$, $i \in \{1, \dots, N_s\}$, à savoir un modèle par capteur. Pour cela, nous construisons une base de données à utiliser pour l'apprentissage. Nous considérons ainsi N_p positions de référence, notées $\boldsymbol{p}_\ell = (p_{\ell,1} \dots p_{\ell,\delta})^\top$, $\ell \in \{1, \dots, N_p\}$. Soit $\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}$, $i \in \{1, \dots, N_s\}$, la puissance RSSI du signal envoyé par le capteur à la position $\boldsymbol{s}_i$ et mesuré à la position $\boldsymbol{p}_\ell$ et soit $d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}$, $i \in \{1, \dots, N_s\}$ la distance séparant $\boldsymbol{p}_\ell$ de $\boldsymbol{s}_i$. De cette façon, N_s ensembles d'apprentissage de N_p couples $(\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}, d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell})$ sont ainsi obtenus. La phase d'apprentissage consiste alors à trouver l'ensemble des modèles χ_i, $i \in \{1, \dots, N_s\}$, tels que chaque modèle χ_i prenne en entrée $\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}$, $\ell \in \{1, \dots, N_p\}$, et donne en sortie la distance $d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}$ séparant $\boldsymbol{s}_i$ de $\boldsymbol{p}_\ell$. La définition des modèles χ_i sera détaillée à la sous-section suivante.

Soit $\boldsymbol{x}(k) = (x_1(k) \dots x_\delta(k))^\top$ la position de la cible à un pas de temps k. La cible mesure en temps réel les RSSIs des signaux reçus des N_s capteurs, tel que $\rho_{\boldsymbol{s}_i,\boldsymbol{x}(k)}$ désigne la mesure de RSSI du signal envoyé par le capteur à la position $\boldsymbol{s}_i$ et reçu par la cible à la position $\boldsymbol{x}(k)$. L'estimé de la distance séparant la cible du capteur à la position $\boldsymbol{s}_i$, notée $\widehat{d}_{\boldsymbol{s}_i,\boldsymbol{x}(k)}, i \in \{1, \dots, N_s\}$, est alors obtenue en utilisant le modèle χ_i comme suit: $\widehat{d}_{\boldsymbol{s}_i,\boldsymbol{x}(k)} = \chi_i(\rho_{\boldsymbol{s}_i,\boldsymbol{x}(k)})$. D'un autre côté, la cible est supposée être équipée d'un gyroscope et d'un accéléromètre qui mesurent respectivement son orientation et ses accélérations. Le problème de suivi de cibles est résolu de deux façons différentes, en utilisant à la fois les distances estimées et les données inertielles de la cible soit avec un filtre de Kalman soit avec un filtre particulaire.

A.4.2 Définition des modèles χ_i

L'objectif maintenant est de définir l'ensemble des modèles $\chi_i,\ i \in \{1,\dots,N_s\}$, en utilisant l'ensemble d'apprentissage déjà défini. Chaque modèle χ_i doit associer à toute mesure de RSSI $\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell},\ \ell \in \{1,\dots,N\}$, la distance correspondante $d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}$. Dans le premier paragraphe, nous définissons l'ensemble des modèles χ_i comme des modèles non-paramétriques ; ils sont alors pris comme des combinaisons linéaires de noyaux centrés sur les données d'apprentissage. Dans le deuxième paragraphe, les modèles χ_i sont des modèles de régression semi-paramétriques qui combinent le modèle de propagation logarithmique [Medeisis and Kajackas, 2000; Patwari et al., 2005] avec un terme de fluctuation non linéaire, défini dans un EHNR. Ce terme non linéaire compense les facteurs manquants dans le modèle logarithmique, permettant ainsi une meilleure modélisation de la relation RSSI/distance.

A.4.2.1 Modèles non-paramétriques

Dans ce paragraphe, les modèles de distances sont obtenus en se basant seulement sur l'ensemble d'apprentissage et donc sans faire aucune hypothèse sur le modèle. Par conséquent, nous définissons χ_i en utilisant une technique de régression non linéaire, à savoir, la technique de kernel ridge regression. Ainsi, chaque fonction χ_i est obtenue en minimisant l'erreur quadratique entre les sorties estimées $\chi_i(\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell})$ et les sorties désirées $d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}$, comme suit :

$$\chi_i = \arg\min_{\underline{\chi}_i \in \mathcal{H}} \frac{1}{N_p} \sum_{\ell=1}^{N_p} (d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell} - \underline{\chi}_i(\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}))^2 + \eta_i \|\underline{\chi}_i\|_{\mathcal{H}}^2,$$

où η_i sont des paramètres de régularisation. D'après le théorème de représentation, la fonction optimale a la forme suivante :

$$\chi_i(\cdot) = \sum_{\ell=1}^{N_p} \alpha_{\ell,i}\, \kappa(\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}, \cdot), \tag{A.14}$$

où $\alpha_{\ell,i}, \ell \in \{1,\dots,N_p\}$, sont des paramètres à déterminer similairement à la Section A.2.

A.4.2.2 Modèles semi-paramétriques

Les modèles proposés dans ce paragraphe sont semi-paramétriques, chaque modèle étant une combinaison du modèle de propagation logarithmique avec une composante non

linéaire, définie à l'aide des méthodes à noyaux. Ce terme non linéaire permet de compenser les facteurs qui ont été négligés par le modèle physique, tels que les pertes dues aux obstacles physiques et autres interférences, permettant ainsi une meilleure modélisation de la relation RSSI/distance. Soit $\psi_i(\cdot)$ la fonction qui associe à toute mesure de RSSI $\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}$ le logarithme de la distance, c'est-à-dire $\log_{10}(d_{\boldsymbol{s}_i,\boldsymbol{p}_\ell})$. Nous définissons cette fonction comme une somme d'un modèle linéaire déduit du modèle de propagation logarithmique (A.9) et d'un modèle non linéaire comme suit :

$$\psi_i(\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}) = a\,\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell} + b + \psi_{i,\text{nlin}}(\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell}),$$

où a et b sont des inconnues à estimer. Les inconnues sont déterminées en minimisant l'erreur quadratique sur l'ensemble d'apprentissage, comme suit:

$$\frac{1}{N_p}\sum_{\ell=1}^{N_p}\Big(\log_{10} d_\ell - a\,\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell} - b - \psi_{i,\text{nlin}}(\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell})\Big)^2 + \eta_i\|\psi_{i,\text{nlin}}\|^2_{\mathcal{H}}, \tag{A.15}$$

où les η_i sont aussi des paramètres de régularisation permettant d'atteindre un compromis entre la justesse de la solution et sa complexité. En se basant sur le théorème de représentation semi-paramétrique [Schölkopf et al., 2001a], nous proposons d'écrire le terme non linéaire sous forme d'une combinaison de noyaux, tel que $\psi_{i,\text{nlin}}(\cdot) = \sum_{\ell=1}^{N_p}\beta_{\ell,i}\,\kappa(\rho_{\boldsymbol{s}_i,\boldsymbol{p}_\ell},\cdot)$, où $\beta_{\ell,i}$, $\ell \in \{1,\ldots,N_p\}$, sont des paramètres à déterminer. Une solution détaillée est proposée à la Sous-section 5.3.2, page 104. Après avoir trouvé les paramètres du modèle, il est possible d'estimer n'importe quelle distance séparant un capteur à la position $\boldsymbol{s}_i$ d'une cible à la position $\boldsymbol{x}(k)$ en utilisant la mesure de RSSI comme suit: $\chi_i\left(\rho_{\boldsymbol{s}_i,\boldsymbol{x}(k)}\right) = 10^{\log_{10} d_{\boldsymbol{s}_i,\boldsymbol{x}(k)}} = 10^{\psi_i\left(\rho_{\boldsymbol{s}_i,\boldsymbol{x}(k)}\right)}$.

A.4.3 Solution à l'aide du filtre de Kalman

Dans cette section, nous proposons d'estimer la position de la cible en combinant ses accélérations instantanées aux distances la séparant des capteurs du réseau à l'aide du filtre de Kalman (KF). Les distances sont estimées à l'aide d'un des modèles définis à la sous-section précédente. Nous utilisons un modèle d'état du troisième ordre pour décrire la trajectoire de la cible, comme ce modèle a donné des résultats bien meilleurs que les modèles du premier ordre ou du second. Soit $\boldsymbol{X}(k) = (\boldsymbol{x}(k)\ \boldsymbol{\nu}(k))^\top$ l'état inconnu de la cible à un instant k, $\boldsymbol{\nu}(k)$ étant son vecteur vitesse. L'équation d'état sera donnée comme suit :

$$\boldsymbol{X}(k) = \boldsymbol{A}\ \boldsymbol{X}(k-1) + \boldsymbol{B}(k) + \boldsymbol{\epsilon}_{\text{KF}}(k), \tag{A.16}$$

où la matrice de transition $\boldsymbol{A}$ et le vecteur de commande $\boldsymbol{B}(k)$ sont définis comme à la Section A.3. La quantité $\boldsymbol{\epsilon}_{\mathrm{KF}}(k) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{V})$ est l'erreur du modèle d'état ayant une distribution normale à moyenne nulle et une matrice de covariance $\boldsymbol{V}$ de taille $2\delta \times 2\delta$.

Ensuite, pour définir un modèle linéaire d'observation, nous utilisons les différences des carrés des distances $d^2_{\boldsymbol{s}_i,\boldsymbol{x}(k)}$; nous obtenons ainsi un modèle de la forme suivante :

$$\boldsymbol{z}_{\mathrm{KF}}(k) = \boldsymbol{C}\boldsymbol{X}(k) + \boldsymbol{n}(k), \tag{A.17}$$

où $\boldsymbol{n}(k)$ est l'erreur d'observation, suivant une distribution normale de moyenne nulle et de matrice de covariance $\boldsymbol{R}$. Le vecteur des observations $\boldsymbol{z}_{\mathrm{KF}}(k)$ est donné par :

$$\begin{pmatrix} d^2_{\boldsymbol{s}_1,\boldsymbol{x}(k)} - d^2_{\boldsymbol{s}_{N_s},\boldsymbol{x}(k)} + \sum_{v=1}^{\delta} s^2_{N_s,v} - \sum_{v=1}^{\delta} s^2_{1,v} \\ \vdots \\ d^2_{\boldsymbol{s}_{N_s-1},\boldsymbol{x}(k)} - d^2_{\boldsymbol{s}_{N_s},\boldsymbol{x}(k)} + \sum_{v=1}^{\delta} s^2_{N_s,v} - \sum_{v=1}^{\delta} s^2_{N_s-1,v} \end{pmatrix}$$

La matrice d'observation $\boldsymbol{C}$, de taille $(N_s - 1) \times 2\delta$ est alors donnée par :

$$\boldsymbol{C} = 2 \begin{pmatrix} s_{N_s,1} - s_{1,1} & \dots & s_{N_s,\delta} - s_{1,\delta} & \\ \vdots & & \vdots & \mathbf{0}_{(N_s-1)\times\delta} \\ s_{N_s,1} - s_{N_s-1,1} & \cdots & s_{N_s,\delta} - s_{N_s-1,\delta} & \end{pmatrix},$$

où $\mathbf{0}_{(N_s-1)\times\delta}$ désigne la matrice de zéros de taille $(N_s - 1) \times \delta$.

Maintenant que nous avons défini les deux équations linéaires d'état et d'observation, nous procédons à l'estimation de la position de la cible à chaque pas de temps k à l'aide du KF. Ce filtre prédit d'abord la position de la cible à l'aide de l'estimation de sa position précédente et de l'équation d'état, comme suit :

$$\widehat{\boldsymbol{X}}^{-}(k) = \boldsymbol{A}\,\widehat{\boldsymbol{X}}(k-1) + \boldsymbol{B}(k), \tag{A.18}$$

où $\widehat{\boldsymbol{X}}^{-}(k)$ est l'état estimé de la cible à $k-1$, avec $\widehat{\boldsymbol{X}}(0)$ supposé connu. Ensuite, la prédiction de la matrice de covariance de l'estimation est mise à jour telle que $\boldsymbol{T}^{-}(k) = \boldsymbol{A}\boldsymbol{T}(k-1)\boldsymbol{A}^{\top} + \boldsymbol{Q}(k)$, où $\boldsymbol{T}(k-1)$ est la matrice de covariance à $k-1$ et $\boldsymbol{T}(0)$ est nulle puisque l'état initial est connu. La matrice $\boldsymbol{Q}(k)$ est la matrice de covariance de $\boldsymbol{X}(k)$ sachant $\boldsymbol{X}(k-1)$, donnée à la Sous-section 5.4, 106. La dernière étape de filtrage consiste à corriger les quantités prédites en utilisant l'équation d'observation. Nous obtenons alors :

$$\begin{aligned} \widehat{\boldsymbol{X}}(k) &= \widehat{\boldsymbol{X}}^{-}(k) + \boldsymbol{G}_{\mathrm{KF}}(k)\,(\boldsymbol{z}_{\mathrm{KF}}(k)^{\top} - \boldsymbol{C}\widehat{\boldsymbol{X}}^{-}(k)), \\ \boldsymbol{T}(k) &= (\mathbf{I}_{2\delta} - \boldsymbol{G}_{\mathrm{KF}}(k)\,\boldsymbol{C})\,\boldsymbol{T}^{-}(k), \end{aligned}$$

où $\boldsymbol{G}_{\mathrm{KF}}(k) = \boldsymbol{T}^{-}(k)\,\boldsymbol{C}^{\top}\,(\boldsymbol{C}\,\boldsymbol{T}^{-}(k)\,\boldsymbol{C}^{\top} + \boldsymbol{R})^{-1}$ est le gain optimal de Kalman.

A.4.4 Solution à l'aide du filtre particulaire

L'objectif maintenant est de trouver la position de la cible en combinant, à l'aide du filtre particulaire (PF), les accélérations de la cible aux distances estimées par les modèles semi-paramétriques. A la différence du KF, le modèle d'observation du PF peut être non linéaire et par la suite, nous prenons comme observations les distances estimées séparant la cible des capteurs comme suit :

$$\boldsymbol{z}_{\mathrm{PF}}(k) = \boldsymbol{d}_{x(k)},$$

où $\boldsymbol{d}_{x(k)} = (d_{\boldsymbol{s}_1,\boldsymbol{x}(k)} \dots d_{\boldsymbol{s}_{N_s},\boldsymbol{x}(k)})^{\top}$. Pour résoudre le problème à l'aide du PF, nous générons N_{PF} particules autour de la position initiale $\boldsymbol{x}(0)$ considérée comme connue. Ces particules notées $\boldsymbol{x}^m(0)$ ont des poids initiaux égaux : $\omega^m(0) = \frac{1}{N_{\mathrm{PF}}}$, $m \in \{1, \dots, N_{\mathrm{PF}}\}$. En partant de la position initiale, les particules $\boldsymbol{x}^m(k)$ et leurs vitesses $\boldsymbol{\nu}^m(k)$ à un pas de temps k sont estimées en utilisant le modèle d'état du troisième ordre :

$$\boldsymbol{\nu}^m(k) = \boldsymbol{\nu}^{m'}(k-1) + \boldsymbol{\gamma}(k-1)\,\Delta t + \frac{\boldsymbol{\gamma}(k) - \boldsymbol{\gamma}(k-1)}{\Delta t}\,\frac{\Delta t^2}{2},$$
$$\boldsymbol{x}^m(k) = \boldsymbol{x}^{m'}(k-1) + \boldsymbol{\nu}^{m'}(k-1)\,\Delta t + \boldsymbol{\gamma}(k-1)\,\frac{\Delta t^2}{2} + \frac{\boldsymbol{\gamma}(k) - \boldsymbol{\gamma}(k-1)}{\Delta t}\,\frac{\Delta t^3}{6} + \boldsymbol{\epsilon}_{\mathrm{PF}}(k),$$

où $\boldsymbol{x}^{m'}(k-1)$ est une des particles au pas de temps $k-1$, choisie aléatoirement selon la distribution discrète de leurs poids, $\boldsymbol{\nu}^{m'}(k-1)$ est la vitesse associée à $\boldsymbol{x}^{m'}(k-1)$ et $\boldsymbol{\epsilon}_{\mathrm{PF}}(k) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{H})$ est l'erreur sur l'équation d'état, suivant une distribution normale de moyenne nulle et de matrice de covariance $\boldsymbol{H}$ de taille $\delta \times \delta$. Les poids des particules sont mis à jour suivant les distances observées Farmani et al. [2012] par $\omega^m(k) = \frac{1}{\|\boldsymbol{d}_{\boldsymbol{x}^m(k)} - \boldsymbol{z}_{\mathrm{PF}}(k)\|}\,\omega^{m'}(k-1)$, où $\|\boldsymbol{d}_{\boldsymbol{x}^m(k)} - \boldsymbol{z}_{\mathrm{PF}}(k)\|$ est la distance entre $\boldsymbol{z}_{\mathrm{PF}}(k)$ et $\boldsymbol{d}_{\boldsymbol{x}^m(k)}$. Les poids ainsi obtenus sont ensuite normalisés. La dernière étape est l'étape de rééchantillonnage, où les particules ayant des poids négligeables sont remplacées par de nouvelles particules créées autour des particules ayant des poids plus importants. Le passage à cette étape se fait lorsque le nombre effectif de particules N_{eff} devient inférieur à un seuil N_{th}, avec $N_{\mathrm{eff}} = \frac{1}{\sum_{m=1}^{N_{\mathrm{PF}}} (\omega^m(k))^2}$. Notons que N_{th} est habituellement égal à 10% du nombre de particules N_{PF}. Finalement, la position de la cible au pas de temps k est donnée par :

$$\widehat{\boldsymbol{x}}(k) = \sum_{m=1}^{N_{\mathrm{PF}}} \omega^m(k)\;\boldsymbol{x}^m(k).$$

Cet algorithme est plus complexe en termes de calculs que la méthode basée sur le KF, vu toutes les particules à définir et l'étape de rééchantillonnage ; cependant, il est plus robuste quand les erreurs sur les distances ne sont plus gaussiennes.

A.4.5 Analyse et simulations

Dans cette sous-section, nous utilisons pour l'estimation des distances les modèles non-paramétriques. En effet, notre choix est basé sur une étude présentée à la Section 5.6 (page 111), où nous comparons nos modèles non-paramétriques et semi-paramétriques aux différents modèles de la littérature pour des données simulées et réelles. Les modèles proposés sont plus précis que les modèles de la littérature que ce soit pour données simulées ou pour des données réelles. De plus, les modèles semi-paramétriques se sont avérés les plus précis dans le cas des données réelles, alors que les non-paramétriques sont les plus précis dans le cas des données simulées de RSSIs, comme celles que nous utiliserons dans cette sous-section. Notons que les données simulées sont générées selon le modèle de propagation logarithmique (A.9). Dans la suite, nous évaluons les performances de l'approche proposée et nous comparons nos résultats tout d'abord à la méthode de suivi de cibles combinant l'algorithme WKNN et le filtre de Kalman, ensuite, à la méthode de suivi proposée à la Section A.3.

A.4.5.1 Illustration générale

Considérons la trajectoire de la figure 5.5, où 100 positions de référence et 16 capteurs sont uniformément générés dans une région de 100 m×100 m. Nous ajoutons un bruit additif gaussien aux RSSIs, de moyenne nulle et d'écart-type égal à 10% de l'écart-type des RSSIs, c'est-à-dire $\sigma_{\boldsymbol{\rho}} = 1{,}08$ dB. Nous ajoutons également un bruit aux accélérations d'écart-type $\boldsymbol{\sigma_\gamma}$ égal à 5% de celui des accélérations. Les distances sont estimées à l'aide des modèles non-paramétriques, utilisés avec un noyau gaussien, et la trajectoire est ensuite estimée à l'aide du filtre de Kalman (KF) et du filtre particulaire (PF). Une erreur moyenne de 1,03m est obtenue en utilisant le KF et de 0,68m en utilisant le PF (avec $N_{\mathrm{PF}} = 50$). Les deux méthodes donnent donc une estimation très satisfaisante des positions de la cible.

A.4.5.2 Impact des paramètres $\sigma_{\boldsymbol{\rho}}$ et $\boldsymbol{\sigma_\gamma}$

Dans cette partie, nous étudions l'impact des paramètres $\sigma_{\boldsymbol{\rho}}$ et $\boldsymbol{\sigma_\gamma}$ sur notre méthode de suivi. Pour cela, nous fixons d'abord $\sigma_{\boldsymbol{\rho}}$ à 10% de l'écart-type des mesures de RSSIs. Ensuite, nous varions $\boldsymbol{\sigma_\gamma}$ de 1% à 10% de l'écart-type des accélérations. La figure 5.7

montre la variation de l'erreur d'estimation en fonction de la variation de $\boldsymbol{\sigma_\gamma}$. Nous remarquons que la technique utilisant le filtre de particulaire donne un meilleur résultat que celle utilisant le filtre de Kalman sous ces conditions. Nous fixons ensuite la valeur de $\boldsymbol{\sigma_\gamma}$ à 5% de l'écart-type des accélérations et nous faisons varier $\sigma_{\boldsymbol{\rho}}$ de 0% à 50%. La figure 5.8 montre l'impact de la variation de σ_ρ sur l'erreur d'estimation pour les deux techniques. Nous pouvons voir que le filtre particulaire donne de meilleurs résultats que le filtre de Kalman. Cela est dû à la distribution des erreurs sur les observations, supposée être gaussienne dans le filtre de Kalman. En effet, les erreurs d'observation restent faibles lorsque le bruit sur les RSSIs est faible ; dans ce cas, la distribution du bruit d'observation reste proche d'une distribution gaussienne, ce qui explique pourquoi le filtre de Kalman fonctionne bien. Cependant, pour des erreurs plus élevées, la distribution du bruit s'éloigne d'une gaussienne ; le filtre particulaire donne dans ce cas une meilleure estimation. Toutefois, les erreurs d'estimation sont faibles pour les deux méthodes et les résultats obtenus sont satisfaisants et précis.

A.4.5.3 Comparaison à d'autres méthodes de suivi de cibles

Dans cette sous-section, nous comparons nos deux méthodes à celle proposée par Chan et al. [2009], en adoptant une configuration similaire. Le tableau 5.7 montre les erreurs d'estimation obtenues avec nos deux méthodes, la méthode de WKNN et la méthode de WKNN avec Kalman. Nous pouvons voir que nos deux méthodes donnent les meilleurs résultats. Ensuite, nous faisons varier le bruit sur les RSSIs. La figure 5.9 montre l'erreur d'estimation en fonction de σ_ρ. Nous remarquons que nos méthodes surpassent les autres méthodes basées sur l'algorithme WKNN.

Nous comparons aussi ces méthodes pour différentes valeurs du nombre de capteurs N_s, supposés fixes, pour $N_p = 100$ positions de référence. $\sigma_{\boldsymbol{\rho}}$ est égal à 10% de l'écart-type des RSSIs et $\boldsymbol{\sigma_\gamma}$ égal à 5% de l'écart-type des accélérations. La figure 5.10 montre l'erreur en fonction de N_s. Nous pouvons voir que les méthodes proposées donnent la plus faible erreur pour n'importe quelle valeur de N_s. De plus, nos méthodes sont le moins affectées par les variations de N_s. Finalement, nous considérons la méthode de localisation que nous avons proposée à la Section A.2 et la méthode de suivi proposée à la Section A.3, en utilisant la technique de kernel ridge regression. Nous gardons les mêmes valeurs pour les bruits, et nous considérons 16 capteurs. Le tableau **??** montre que les trois méthodes de suivi que nous avons proposées donnent des résultats très proches. Ensuite, nous faisons varier la positions des capteurs dans un rayon allant de 0 à 30 m et un angle de 0 à 2π. L'estimation de l'erreur en fonction du rayon pour les différentes méthodes proposées est donnée à la figure 5.11. Nous pouvons voir que les méthodes de suivi basées sur les distances sont plus performantes que les méthodes de suivi utilisant

les positions. En effet, l'avantage des méthodes proposées dans cette section réside dans le fait que nous estimons la relation RSSI/distance, et donc nous trouvons directement les distances à partir des RSSIs. Lorsque les capteurs changent leurs positions, cette relation ne change pas, donc nous n'avons pas besoin de reconfigurer le modèle, ce qui n'est pas le cas pour les modèles utilisant les positions.

A.4.6 Conclusion

Cette section a proposé deux modèles originaux pour estimer les distances entre capteurs en utilisant les puissances des signaux échangés. Ces distances sont combinées aux accélérations instantanées de la cible en utilisant un filtrage de Kalman ou un filtrage particulaire. L'approche proposée s'est avérée robuste face au bruit et plus performante comparée à l'état-de-l'art. Les travaux futurs y apporteront des améliorations, notamment en termes de sélection des RSSIs à utiliser pour l'estimation.

A.5 Estimation des paramètres de sources de gaz

Dans un contexte différent de celui de la localisation et du suivi de cibles, les RCSF sont largement utilisés pour la détection et la localisation de la source d'une diffusion de gaz dans l'environnement. Comme nous l'avons expliqué à la Section A.1, les émissions de gaz pourraient se produire accidentellement ou délibérément. Dans les deux cas, leurs conséquences sont potentiellement catastrophiques sur l'environnement ou sur la vie humaine. Il est donc important de détecter la diffusion de gaz et d'estimer ses paramètres, y compris l'emplacement de la source et la quantité de gaz libéré. Une collecte de mesures de concentration de gaz à partir de la zone infectée est indispensable pour l'estimation des paramètres mais elle nécessite une intervention de personnes spécialement formées, utilisant des équipements de protection appropriés. Alternativement, les réseaux de capteurs sans fil se sont révélés très utiles dans de tels scénarios, où l'intervention humaine est risquée et coûteuse. Typiquement, les capteurs sont déployés dans la zone d'intérêt et collectent régulièrement des mesures de concentration. L'information recueillie est renvoyée à un centre de fusion, où elle sera traitée afin d'estimer les paramètres de la source.

Plusieurs méthodes ont été proposées dans la littérature pour l'estimation des paramètres de la source. Par exemple, Kathirgamanathan et al. [2002] ont développé un modèle inverse permettant de déduire les paramètres d'une source à partir des mesures de concentration de gaz. Dans [Christopoulos and Roumeliotis, 2005], les auteurs déterminent

les paramètres de la source à l'aide de robots mobiles qui collectent les mesures de concentration. L'étude porte aussi sur le choix de la séquence d'emplacements où chaque robot doit être déplacé afin d'obtenir des estimations précises en temps réel. Une autre méthode a été proposée dans [Delmaire and Roussel, 2012], où le problème de la localisation de la source est abordé dans un contexte unidimensionnel et l'estimation est conduite dans un cadre bayésien. Ickowicz et al. [2012] ont aussi proposé une méthode pour la prédiction et l'estimation des concentrations de polluants dans des environnement complexes. Ils se basent sur une formulation semi-paramétrique du problème et utilisent des processus gaussiens pour estimer la région dans laquelle se trouve la source et le temps de diffusion. Toutes les méthodes ci-dessus exigent une compréhension approfondie de la topologie du RCSF et nécessitent une connaissance a priori du processus de diffusion. Pour surmonter ces contraintes, nous pouvons clairement profiter de la flexibilité des méthodes à noyaux pour une estimation non-paramétrique, comme nous montrerons dans ce qui suit.

Dans cette section, nous proposons une nouvelle approche clusterisée pour la détection et l'estimation des paramètres de multiples sources de gaz dans les RCSF. Les capteurs sont déployés dans la région d'intérêt et collectent des mesures de concentration à intervalles de temps courts et réguliers. L'approche proposée est composée de deux phases: une phase de détection et une phase d'estimation. Dans la première, nous définissons un détecteur dans le cadre des méthodes à noyaux en utilisant la technique de *support vector data description* (SVDD) [Tax and Duin, 2004]. Le détecteur est utilisé par chaque tête de cluster afin d'identifier toute émission de gaz dans le cluster concerné. Lorsqu'une anomalie est repérée, le vecteur de concentration qui a déclenché l'alerte est d'abord traité pour estimer les paramètres de diffusion de gaz. Une première estimation des paramètres est obtenue à l'aide d'un modèle non linéaire, qui est également définie dans le cadre des méthodes à noyaux. Ensuite, une partie des paramètres estimés est traitée, en plus des concentrations mesurées, afin de fournir une estimation plus précise de la position de la source. Les simulations montrent que l'approche proposée donne des résultats précis, que ce soit pour une source unique ou pour plusieurs sources.

A.5.1 Problématique

Nous considérons N capteurs uniformément ou aléatoirement déployés dans une région Ω à surveiller à des positions fixes $(x_n, y_n, z_n) = (x_n, y_n, 0),\ n \in \{1, \ldots, N\}$. La région est partitionnée en Z clusters distincts afin de simplifier les calculs et rendre la méthode plus robuste face aux pannes du réseau. Chaque cluster est dirigé par une tête de cluster, qui est responsable du traitement des données (collecte et synchronisation) et des calculs et qui peut échanger des informations avec les autres têtes de cluster et les capteurs du

réseau. Notons que ces capteurs mesurent, à chaque instant t, les concentrations de gaz et les envoient aux têtes de cluster. Les clusters peuvent avoir n'importe quelle taille ou forme ; un exemple de topologie considérée dans cette section, où les clusters sont rectangulaires, est illustré par la figure 6.1.

Dans ce qui suit, nous proposons une version généralisée du modèle de diffusion donné dans [Kathirgamanathan et al., 2002]. Ainsi, nous considérons une émission de gaz de masse Q, se produisant à l'instant t_0 à une position (x_0, y_0, z_0). Les particules de gaz sont propagées par un vent de vitesse moyenne $\boldsymbol{U} = (u_x, u_y, 0)$. La concentration C du gaz, à une position arbitraire (x, y, z) et à un temps t, est donnée par :

$$C(x,y,z,t) = \frac{Q}{8\ \pi^{\frac{3}{2}}\ (K_xK_yK_z)^{\frac{1}{2}}\ \Delta t^{\frac{3}{2}}} \times \exp\left(-\frac{(\Delta x - u_x\Delta t)^2}{4K_x\Delta t} - \frac{(\Delta y - u_y\Delta t)^2}{4K_y\Delta t}\right) \times \left(\exp\left(-\frac{\Delta z^2}{4K_z\Delta t}\right) + \exp\left(-\frac{\Delta z'^2}{4K_z\Delta t}\right)\right),$$

où K_x, K_y et K_z sont les constantes de diffusion, avec $\Delta x = x - x_0$, $\Delta y = y - y_0$, $\Delta z = z - z_0$, $\Delta z' = z + z_0$ et $\Delta t = t - t_0$. Pour la simplicité, nous supposons que les mesures sont toutes collectées au niveau du sol et que l'émission de gaz se produit aussi à $z_0 = 0$. Ainsi, la concentration mesurée à $(x, y, 0)$ est-elle donnée par :

$$C(x,y,0,t) = \frac{Q}{4\ \pi^{\frac{3}{2}}\ (K_xK_yK_z)^{\frac{1}{2}}\ \Delta t^{\frac{3}{2}}} \times \exp\left(-\frac{(\Delta x - u_x\Delta t)^2}{4K_x\Delta t} - \frac{(\Delta y - u_y\Delta t)^2}{4K_y\Delta t}\right). \tag{A.19}$$

Notons que la vitesse du vent $\boldsymbol{U}$ pourrait être mesurée par un anémomètre ; elle est donc considérée comme une constante connue [Christopoulos and Roumeliotis, 2005]. Les autres paramètres du modèle sont inconnus et doivent alors être estimés. Ayant des concentrations de gaz mesurées sur une certaine zone en utilisant un RCSF, nous visons à détecter l'émission de gaz dès qu'elle se produit. Ensuite, l'objectif est d'estimer la position de la source $(x_0, y_0, 0)$, la masse de gaz libéré Q, et les constantes de diffusion K_x, K_y et K_z, avec K_x et K_y supposées égales [Kathirgamanathan et al., 2002].

A.5.2 Description de la méthode proposée

Comme nous l'avons déjà expliqué, la méthode proposée est constituée de deux phases : une phase de détection et une phase d'estimation. Nous décrivons brièvement ces deux phase dans cette sous-section. Plus d'explications sur la définition des modèles nécessaires pour la détection et l'estimation sont données ultérieurement.

Soit $N^{(z)}$ le nombre de capteurs dans le cluster z, $z \in \{1, ..., Z\}$, et soit $\boldsymbol{C}^{(z)}(t)$ le vecteur de taille $N^{(z)} \times 1$, composé des concentrations recueillies par la tête de cluster

z à un instant t. La première étape de la méthode proposée consiste à développer un détecteur capable de déterminer si les concentrations sont normales ou non, à savoir, si une émission de gaz a eu lieu ou non dans la région surveillée. Par conséquent, l'objectif est de définir la limite de la classe des concentrations normales, dite classe normale. Ainsi, toute concentration n'appartenant pas à cette classe sera considérée comme anormale. Un classifieur mono-classe est en mesure d'effectuer cette tâche. Ce classifieur peut être défini dans le cadre des méthodes à noyaux en utilisant par exemple la technique de one-class support vector machines (one-class SVM) [Schölkopf et al., 2001b] ou la technique de support vector data description (SVDD) [Tax and Duin, 2004]. Nous définissons à la sous-section suivante A.5.3 notre détecteur à l'aide de la technique de SVDD.

Dès qu'une alerte est déclenchée dans un cluster z, nous procédons au traitement du vecteur de concentrations $\boldsymbol{C}^{(z)}(t)$ relevées par les capteurs du cluster considéré. Nous groupons d'abord les clusters adjacents qui ont déclenché des alertes de façon à discerner toutes les sources d'émission de gaz. La figure 6.3 illustre le groupement dans le cas de six diffusions de gaz. Nous aurons ainsi plusieurs groupes de différentes tailles, chacun couvrant une ou plusieurs émissions de gaz. Ensuite, afin de détecter les multiples diffusions dans un même groupe, nous considérons les maximums locaux, conduisant idéalement aux concentrations relevées par les capteurs les plus proches des sources de gaz (pour illustration, voir la figure 6.4). Soit N_s le nombre de maximums locaux détectés ; cela signifie que N_s sources ont émis du gaz à l'instant t, ou à des instants différents, mais sont détectées et estimées simultanément. Seules les concentrations recueillies autour des maximums locaux seront traitées ; en effet, les concentrations les plus élevées sont recueillies par les capteurs autour de la source de la diffusion, et ces mesures de concentrations sont les plus pertinentes pour l'estimation de paramètres. Se référer à la Section 6.4 (page 136) pour les détails concernant le choix des concentrations pertinentes et la définition de la zone locale délimitant ces concentrations. Soit $\boldsymbol{c}^{(s)}$, $s \in \{1, ..., N_s\}$, le vecteur des concentrations collectées autour de chaque maximum local s (ainsi autour de la source s), aux alentours du temps de la détection. Notre premier objectif est de définir un modèle d'estimation $\boldsymbol{\psi}_A$ qui prend en entrée le vecteur $\boldsymbol{c}^{(s)}$ des concentrations et donne en sortie le vecteur $\boldsymbol{\theta}^{(s)}$ constitué des paramètres de la source s, notamment la masse de gaz libéré $Q^{(s)}$, la position de la source $(x_0^{(s)}, y_0^{(s)}, z_0^{(s)})$ et les constantes de diffusions. Ensuite, notre second objectif est de définir un second modèle $\boldsymbol{\psi}_B$ qui permet d'affiner l'estimation de la position de la source. Ces deux modèles sont définis dans le cadre des méthodes à noyaux à la Sous-section A.5.4. Finalement, la figure 6.2 illustre le schéma fonctionnel de la méthode proposée.

A.5.3 Définition du détecteur

Dans cette sous-section, nous considérons la technique de SVDD pour définir le détecteur qui permet d'identifier toute émission de gaz dans un cluster spécifique. Cette technique consiste essentiellement à trouver la plus petite hypersphère possible qui englobe les données normales de l'ensemble d'apprentissage, tout en permettant à certaines données d'être exclues comme valeurs aberrantes. Les données se trouvant en dehors de cette frontière de décision sphérique sont considérées comme anormales. Nous désignons par ϕ: $\mathbb{R}^{N^{(z)}} \rightarrow \mathcal{H}$ la transformation de l'espace (des concentrations) $\mathbb{R}^{N^{(z)}}$ vers un espace $\mathcal{H}$ et nous considérons un noyau reproduisant $\kappa : \mathbb{R}^{N^{(z)}} \times \mathbb{R}^{N^{(z)}} \rightarrow \mathbb{R}$, avec $\mathcal{H}$ son espace de Hilbert à noyau reproduisant.

Nous considérons les données d'apprentissage $\boldsymbol{C}_i$, $i \in \{1, ..., N_{\text{det}}\}$, N_{det} étant la taille de l'ensemble d'apprentissage correspondant à la phase de détection. La technique de SVDD consiste à estimer l'hypersphère de rayon minimal $R > 0$ et de centre $\boldsymbol{a}$ qui englobe toutes les données $\phi(\boldsymbol{C}_i)$ dans l'espace transformé $\mathcal{H}$. Les variables de relaxation ("slack variables¨), $\xi_i \geq 0$, permettent une meilleure description des données en autorisant quelques erreurs de classification lors de l'apprentissage. Le centre $\boldsymbol{a}$, le rayon R et les valeurs des variables de relaxation sont obtenues en minimisant une fonction coût qui permet d'atteindre un compromis entre le volume de l'hypersphère et la pénalité associée aux données aberrantes. Notons que minimiser le volume de l'hypersphère est équivalent à minimiser R^2. Par conséquent, nous obtenons le problème d'optimisation suivant :

$$\min_{\boldsymbol{a},R,\xi_i} R^2 + \frac{1}{\nu N_{\text{det}}} \sum_{i=1}^{N_{\text{det}}} \xi_i,$$

avec les contraintes suivantes :

$$\|\phi(\boldsymbol{C}_i) - \boldsymbol{a}\|_{\mathcal{H}}^2 \leq R^2 + \xi_i \quad \text{and} \quad \xi_i \geq 0 \;\; \forall i = 1, ..., N_{\text{det}}.$$

La quantité ν est un paramètre prédéfini qui assure un compromis entre le volume de l'hypersphère et le nombre des données aberrantes. La solution de ce problème est donnée à la Section 6.3, page 132.

Les capteurs mesurent les concentrations à des intervalles de temps réguliers notés T. Ainsi, chaque T secondes, un nouveau vecteur de concentrations $\boldsymbol{C}^{(z)}(t_d)$ est relevé de chaque cluster z. Afin de déterminer si ce vecteur est normal ou non, il suffit d'évaluer la distance du centre de l'hypersphère à $\phi(\boldsymbol{C}^{(z)}(t_d))$, telle que :

$$\|\phi(\boldsymbol{C}^{(z)}(t_d)) - \boldsymbol{a}\|_{\mathcal{H}}^2.$$

Le vecteur $\boldsymbol{C}^{(z)}(t_d)$ est considéré normal si la distance calculée est inférieure au rayon, telle que $\|\phi(\boldsymbol{C}^{(z)}(t_d)) - \boldsymbol{a}\|^2_{\mathcal{H}} \leq R^2$; sinon, le vecteur est anormal et une alerte est déclenchée dans le cluster considéré. Finalement, il est important de noter que le problème d'optimization dans le cas de la technique de SVDD est essentiellement similaire à celui de la technique de one-class SVM. Par ailleurs, ils sont équivalents quand le noyau gaussien est utilisé.

A.5.4 Définition des modèles d'estimation

Dans cette sous-section, nous définissons les deux modèles qui permettent d'estimer les paramètres de la source s à partir du vecteur de concentrations $\boldsymbol{c}^{(s)}$ relevées par les V capteurs autour de cette source. Ainsi, le vecteur $\boldsymbol{c}^{(s)}$ est de taille $V \times 1$. Tout d'abord, nous définissons un premier modèle noté $\boldsymbol{\psi}_A \colon \mathbb{R}^V \to \mathbb{R}^5$, qui associe au vecteur de concentration $\boldsymbol{c}^{(s)}$ les paramètres $\boldsymbol{\theta}^{(s)}$ de la source, notamment la quantité du gaz libéré $Q^{(s)}$, la position de la source $(x_0^{(s)}, y_0^{(s)}, 0)$ et les constantes de diffusion. Ensuite, nous définissons un second modèle $\boldsymbol{\psi}_B \colon \mathbb{R}^{V+3} \to \mathbb{R}^2$ qui permet d'affiner la position de la source, en utilisant les autres paramètres estimés et le vecteur de concentration $\boldsymbol{c}^{(s)}$. Nous utilisons pour les deux modèles la technique de kernel ridge regression.

Pour la définition du modèle $\boldsymbol{\psi}_A$, nous considérons un ensemble d'apprentissage $(\boldsymbol{c}_\ell, \boldsymbol{\theta}_\ell)$, $\ell \in \{1, \ldots, N_{\text{reg}}\}$, où N_{reg} est la taille de cet ensemble et $\boldsymbol{\theta}_\ell = (Q_\ell \quad K_x \quad K_z \quad x_{0\ell} \quad y_{0\ell})$. Le vecteur est constitué des concentrations mesurées par V capteurs uniformément distribués sur une grille de même taille que celle de la zone locale introduite à la sous-section précédente. Ces concentrations sont mesurées aux alentours d'une émission de gaz de paramètres $\boldsymbol{\theta}_\ell$. Le modèle $\boldsymbol{\psi}_A$ est ensuite donné par :

$$\boldsymbol{\psi}_A(\cdot) = \sum_{\ell=1}^{N_{\text{reg}}} \boldsymbol{\beta}_{\ell,*}\, \kappa_A(\boldsymbol{C}_\ell, \cdot),$$

où $\kappa_A \colon \mathbb{R}^V \times \mathbb{R}^V \to \mathbb{R}$ est un noyau reproduisant et $\boldsymbol{\beta}_{\ell,*}$ est la ℓ-ème ligne de la matrice $\boldsymbol{\beta}$, telle que :

$$\boldsymbol{\beta} = (\boldsymbol{K}_A + \eta_A N_{\text{reg}} \mathbf{I})^{-1} \boldsymbol{\Theta},$$

$\mathbf{I}$ étant la matrice identité de taille $N_{\text{reg}} \times N_{\text{reg}}$, $\boldsymbol{K}_A$ étant la matrice de Gram de taille $N_{\text{reg}} \times N_{\text{reg}}$ associée au noyau κ_A et $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1^\top \ldots \boldsymbol{\theta}_{N_{\text{reg}}}^\top)^\top$. Finalement, le paramètre η_A est un paramètre de régularisation.

Ensuite, pour définir le modèle $\boldsymbol{\psi}_B$, nous considérons un ensemble d'entrées d'apprentissage $\boldsymbol{W}_\ell = (\boldsymbol{c}_\ell^\top \quad Q_\ell \quad K_x \quad K_z)^\top$ et de sorties d'apprentissage $(x_{0\ell} \quad y_{0\ell})$, $\ell \in \{1, \ldots, N_{\text{reg}}\}$. En utilisant la technique de kernel ridge regression, le modèle $\boldsymbol{\psi}_B$ est

défini par :

$$\boldsymbol{\psi}_B(\cdot) = \sum_{\ell=1}^{N_{\text{reg}}} \boldsymbol{\gamma}_{\ell,*}\, \kappa_B(\boldsymbol{W}_\ell, \cdot),$$

où $\kappa_B \colon \mathbb{R}^{V+3} \times \mathbb{R}^{V+3} \to \mathbb{R}$ est un noyau reproduisant et $\boldsymbol{\gamma}_{\ell,*}$ est la ℓ-ème ligne de la matrice $\boldsymbol{\gamma}$, telle que :

$$\boldsymbol{\gamma} = (\boldsymbol{K}_B + \eta_B N_{\text{reg}} \mathbf{I})^{-1} (\boldsymbol{x}_0 \quad \boldsymbol{y}_0),$$

où $\boldsymbol{x}_0$ et $\boldsymbol{y}_0$ désignent les vecteurs de taille $N_{\text{reg}} \times 1$ dont les ℓ-ème éléments sont donnés respectivement par $x_{0\ell}$ et $y_{0\ell}$. La matrice $\boldsymbol{K}_B$ est la matrice de Gram de taille $N_{\text{reg}} \times N_{\text{reg}}$ associée au noyau κ_B et le paramètre η_B est un paramètre de régularisation.

Pour résumer, ayant détecté un maximum local et relevé le vecteur des concentrations associé $\boldsymbol{c}^{(s)}$, nous obtenons une première estimation des paramètres de la source s à l'aide du modèle $\boldsymbol{\psi}_A$, telle que $(\hat{Q} \quad \hat{K}_x \quad \hat{K}_z \quad \hat{x}_0 \quad \hat{y}_0) = \boldsymbol{\psi}_A(\boldsymbol{c}^{(s)})$. Ensuite, afin d'affiner l'estimation de la position de la source, nous utilisons le modèle $\boldsymbol{\psi}_B$, avec en entrée le vecteur $\boldsymbol{W} = (\boldsymbol{c}^{(s)\top} \quad \hat{Q} \quad \hat{K}_x \quad \hat{K}_z)^\top$. La nouvelle position améliorée de la source sera donnée par $(\hat{x}_{0\text{enh}} \quad \hat{y}_{0\text{enh}}) = \boldsymbol{\psi}_B(\boldsymbol{W})$.

A.5.5 Analyse et simulations

Dans cette sous-section, nous évaluons la performance de la méthode de détection et d'estimation proposée pour différents scénarios. Nous considérons une région de dimensions 5000 m × 5000 m. Les capteurs sont uniformément déployés sur une grille dans la zone à raison d'un capteur tous les cinq mètres. La région est divisée en Z =25 clusters de même taille, ayant ainsi la même densité de capteurs. La figure 6.5 illustre la topologie de la région. Les concentrations mesurées par les capteurs sont générées à l'aide du modèle de diffusion donné par (A.19). Les constantes de diffusion sont prises telles que K_z =0.211 m^2/s, tandis que $K_y = K_z$ =12 m^2/s. Quant à la vitesse moyenne du vent $\boldsymbol{U}$, elle est égale à (1,8; 0; 0) m/s. Enfin, le temps d'échantillonnage T est égal à 1 s. Notons que nous utilisons un noyau gaussien pour nos deux modèles de détection et d'estimation. La définition des paramètres est expliquée en détails à la page 143. Dans ce qui suit, nous évaluons d'abord la robustesse de la méthode proposée face au bruit. Nous étudions ensuite l'effet de la variation du nombre de clusters Z et de la densité des capteurs sur la performance de notre méthode. Finalement, nous comparons notre méthode à celle proposée par Kathirgamanathan et al. [2002].

Nous simulons d'abord la cas d'une source unique d'émission de gaz, telle que $x_0 \in [0; 5000]$ m, $y_0 \in [0; 5000]$ m et $Q \in [5; 1000]$ kg. La phase de test se déroule sur un intervalle 10 s où les concentrations sont mesurées chaque T = 1 s. L'émission de gaz a lieu à n'importe quel instant pendant cet intervalle. Un bruit relatif variant de

1% à 5% est ajouté aux vecteurs de concentrations simulées, afin de tenir compte des dégradations de la transmission ou des imprécisions des capteurs. Le tableau 6.2 montre les pourcentages d'erreurs sur les paramètres estimés de la source moyennés sur 50 simulations Monte-Carlo. Les résultats montrent que la méthode est précise et robuste face au bruit. De plus, nous pouvons voir l'amélioration de l'estimée de la position de la source après l'utilisation du second modèle. Ensuite, nous considérons le cas de quatre émissions de gaz, ayant différents paramètres donnés au tableau 6.6. Nous considérons un bruit relatif de 5% et la phase de test s'étend aussi sur 10 s. Le tableau 6.7 montre les pourcentages d'erreurs obtenus. Nous pouvons voir que les estimations sont bien précises ; ainsi, la méthode peut bien être utilisées dans le cas de sources multiples.

Dans ce paragraphe, nous étudions l'impact du nombre de clusters Z et de la densité des capteurs. Pour cela, nous faisons d'abord varier le nombre de clusters, tel que $Z = 3^2, 4^2, \ldots, 7^2$, et nous considérons un nombre fixe de capteurs (1 capteur tous les cinq mètres). Le tableau 6.3 montre que plus le nombre de clusters est élevé, plus le traitement des données pour la détection est rapide. Nous remarquons aussi que la variation de Z n'affecte que les temps d'apprentissage et de test pour la détection, mais n'a aucun effet sur l'estimation puisque celle-ci est indépendante de Z. Notons que nous avons un pourcentage de détection de 100% pour les différentes valeurs considérées de Z. Nous fixons maintenant le nombre de clusters et faisons varier le nombre de capteurs N (et donc la densité des capteurs). Le tableau 6.4 montre les temps d'apprentissage et de test pour les différentes densités considérées. Nous remarquons que la complexité de calcul diminue avec la diminution de la densité. Quant à la précision de la méthode, elle diminue aussi avec la diminution du nombre de capteurs, comme le montre le tableau 6.5. Malgré cela, les résultats sont toujours satisfaisants. Nous notons toutefois que pour le cas d'un capteur tous les vingt mètres, la méthode n'est plus stable puisque les capteurs ne sont plus en mesure de bien couvrir la région.

Finalement, nous terminons cette sous-section par une comparaison de notre méthode à celle proposée par Kathirgamanathan et al. [2002] pour différentes intensités du bruit allant de 1% à 5%. Nous considérons une zone de 5500 m $\times$ 5500 m environ et les mêmes paramètres utilisés dans [Kathirgamanathan et al., 2002]. Le tableau 6.8 montre les pourcentages d'erreurs pour les deux méthodes ; nous pouvons voir que notre méthode est la plus performante. De plus, notre méthode est plus robuste face à l'addition du bruit.

A.5.6 Conclusion

Nous avons introduit dans cette section une nouvelle méthode pour la détection et l'estimation des paramètres de sources multiples dans un RCSF. Les deux phases de la méthode sont définies dans le cadre des méthodes à noyaux. Les simulations ont montré que la méthode donne des estimations précises et qu'elle est robuste face à l'addition du bruit. Les travaux futurs porteront de nouvelles améliorations à cette méthode, de manière à inclure les cas où les constantes de diffusion ou la vitesse du vent ne seront plus fixes.

A.6 Conclusions et perspectives

Cette thèse a abordé plusieurs problèmes dans les réseaux de capteurs sans fil. Tout d'abord, nous avons proposé des méthodes de localisation et de suivi de cibles. Ensuite, nous avons proposé une méthode de surveillance pour la détection et l'estimation de sources multiples de diffusions de gaz dans un RCSF. Cette section résume les contributions majeures de cette thèse et propose quelques perspectives futures de recherche.

A.6.1 Contributions principales

Dans la Section A.2, nous avons introduit dans le cadre des méthodes à noyaux un modèle original pour la localisation dans les RCSF utilisant la technique de fingerprinting. Ce modèle a permis d'estimer la position d'un nœud mobile en utilisant les puissances des signaux échangés (RSSI) entre ce nœud et les capteurs fixes dans le réseau. Plusieurs techniques d'apprentissage par méthodes à noyaux ont été étudiées pour la définition de ce modèle. La méthode centralisée ainsi que son extension en version hybride ont donné des estimations précises de la position du nœud dans le cas de bruit additif modéré sur les RSSIs. Afin d'améliorer les performances de la méthode lorsque le bruit sur les RSSIs est élevé, nous avons proposé à la Section A.3 de combiner les positions estimées aux données d'inertie (si disponibles) du nœud mobile, à savoir, de la cible. La combinaison a été effectuée à l'aide d'un filtre de Kalman et les résultats ont montré qu'elle a permis l'obtention d'estimations plus précises. La principale contrainte de cette méthode proposée a été que les capteurs doivent rester fixes aux mêmes endroits au cours du processus de suivi, ce qui peut être contraignant pour certaines applications.

Afin de surmonter le problème de stationnarité des capteurs, nous avons proposé d'utiliser les distances séparant la cible de ces capteurs pour le processus de suivi. De cette façon, il était seulement nécessaire de connaître les positions des capteurs, et ils

étaient ainsi libres de se déplacer dans le réseau. Ainsi, dans la Section A.4, nous avons proposé deux nouveaux modèles de distance, de sorte que, compte tenu des mesures RSSIs des signaux échangés entre les capteurs, les modèles permettent d'estimer avec précision les distances séparant ces capteurs. Nous avons ensuite proposé deux nouvelles méthodes de suivi de cibles qui combinent les distances estimées aux données inertielles de la cible, en utilisant soit le filtre de Kalman soit le filtre particulaire.

Finalement, nous avons proposé à la Section A.5 une nouvelle méthode clusterisée pour surveiller les diffusions de gaz dans les RCSF. Cette méthode consiste à détecter et à estimer les paramètres des sources de gaz, en utilisant les mesures de concentration collectées à partir de la région étudiée. La phase de détection et d'estimation ont toutes les deux été définies dans le cadre des méthodes à noyaux. En effet, afin de détecter toute anomalie dans la région, nous avons défini un détecteur en utilisant les techniques de classification mono-classe. Ensuite, nous avons défini deux modèles qui estiment les paramètres d'une diffusion de source. Le procédé s'est révélé être précis dans le cas d'une source unique, ainsi que dans le cas de plusieurs sources.

A.6.2 Perspectives

Cette thèse a fourni plusieurs solutions importantes pour la localisation et le suivi de cibles, ainsi que pour la surveillance de la diffusion de gaz à base de réseaux de capteurs sans fil. Dans le cadre de futures recherches, nous tenons à étudier les aspects suivants concernant l'amélioration des méthodes proposées.

- *Choix des RSSIs*
  Les méthodes proposées de localisation et de suivi étaient toutes basées sur l'hypothèse que tous les capteurs sont capables de communiquer et d'échanger des signaux avec la cible. Cependant, en réalité, certains capteurs ne sont peut-être pas dans la portée de communication de la cible, conduisant à un manque de données de puissances. Il est alors important d'adapter les méthodes proposées à de tels cas, en remplaçant par exemple les informations non disponibles par une valeur prédéfinie. D'autre part, nous ne devons pas nécessairement traiter toutes les informations RSSI disponibles. Par exemple, nous pouvons sélectionner de façon dynamique les capteurs devant être inclus dans le processus d'estimation en se basant sur un seuil prédéfini de puissances.

- *Amélioration des modèles de distance*
  Dans la définition des modèles de distance proposés et des modèles connus, tels que le modèle de propagation logarithmique et les modèles polynomiaux, les antennes

des capteurs sont supposés être isotropes. Elles sont supposées émettre de façon uniforme dans toutes les directions ; cependant, une communication uniforme sur toutes les directions n'est pas toujours possible. Une solution à ce problème serait d'intégrer des informations d'orientation dans le modèle, conduisant alors à une carte plus représentative des communications du réseau.

- *Amélioration de la méthode de surveillance*
  Par rapport à l'approche proposée pour la détection et l'estimation des sources de diffusions de gaz, les travaux futurs porteront de nouvelles améliorations visant à inclure les cas où les constantes de diffusion et la vitesse du vent ne seront plus constantes. Nous allons également étudier la possibilité de l'utilisation de l'apprentissage par transfert, qui s'est révélé être utile dans les cas où les données d'apprentissage ne sont pas suffisantes.

- *Définition automatique des clusters*
  Dans la méthode de localisation clusterisée, ainsi que l'approche de surveillance clusterisée, les clusters ont été définis comme rectangulaires et de tailles égales. Les travaux futurs comprendront des études relatives à la définition des clusters, et un algorithme permettant une définition automatique des clusters sera développé.

Bibliography

I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102 – 114, 2002. Survey. 2, 4, 159

J. Al-Karaki and A. Kamal. Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications*, 11(6):6–28, Dec 2004. 4

C. Andrade and R. Hoefel. IEEE 802.11 WLANs: A comparison on indoor coverage models. In *23rd Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–6, May 2010. 99, 113, 177

A. Argyriou, C. A. Micchelli, and M. Pontil. When is there a representer theorem? vector versus matrix regularizers. *CoRR*, abs/0809.1590, 2008. 31, 48

N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68(3):337 – 404, 1950. 26, 28, 42, 163

H. L. B. Hofmann-Wellenhof and J. Collins. *Global Positioning System: Theory and Practice*. Springer, September 2004. 8, 160

S. Bandyopadhyay and E. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications*, volume 3, pages 1713–1723, March 2003. 5, 160

F. Benbadis, T. Friedman, M. Dias de Amorim, and S. Fdida. Gps-free-free positioning system for wireless sensor networks. In *Second IFIP International Conference on Wireless and Optical Communications Networks*, pages 541–545, March 2005. 8, 160

M. Bhuiyan, J. Cao, G. Wang, and X. Liu. Energy-efficient and fault-tolerant structural health monitoring in wireless sensor networks. In *IEEE 31st Symposium on Reliable Distributed Systems*, pages 301–310, Oct 2012. 7

N. G. Branko Ristic, Sanjeev Arulampalam. *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House Publishers, February 2004. 14, 70, 161, 171

S. Capkun and J.-P. Hubaux. Secure positioning of wireless devices with application to sensor networks. In *24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 3, pages 1917–1928, March 2005. 10

N. Chaamwe. Wireless sensor networks for water quality monitoring: A case of zambia. In *International Conference on Bioinformatics and Biomedical Engineering*, pages 1–6, June 2010. 6, 160

E. Chan, G. Baciu, and S. C. Mak. Using wi-fi signal strength to localize in wireless sensor networks. In *WRI Int. Conf. on Communications and Mobile Computing*, volume 1, pages 538–542, 2009. 71, 83, 93, 94, 117, 121, 171, 176, 184

D. Chen and M. Wang. A home security zigbee network for remote monitoring application. In *IET International Conference on Wireless, Mobile and Multimedia Networks*, pages 1–4, Nov 2006. 6

Z. Chen. Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond. Adaptive Syst. Lab., McMaster Univ., Hamilton, ON, Canada, 2003. 109

C.-Y. Chong and S. Kumar. Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, Aug 2003. 3

V. Christopoulos and S. Roumeliotis. Multi robot trajectory generation for single source explosion parameter estimation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2803–2809, April 2005. 15, 127, 132, 162, 185, 187

G. Colistra and L. Atzori. Estimation of physical layer performance in wsns exploiting the method of indirect observations. *Journal of Sensor and Actuator Networks*, 1(3): 272–298, 2012. 10

A. Correa, M. Barcelo, A. Morell, and J. Lopez Vicario. Distance-based tuning of the ekf for indoor positioning in wsns. In *Proceedings of the 22nd European Signal Processing Conference (EUSIPCO)*, pages 1512–1516, Sept 2014. 99, 177

G. Cote, R. Lec, and M. Pishko. Emerging biomedical sensing technologies and their applications. *IEEE Sensors Journal*, 3(3):251–266, 2003. 5

A. Dallil, M. Oussalah, and A. Ouldali. Sensor fusion and target tracking using evidential data association. *IEEE Sensors Journal*, 13(1):285–293, 2013. 12, 70, 170

G. Delmaire and G. Roussel. Joint estimation decision methods for source localization and restoration in parametric convolution processes. application to accidental pollutant release. *Digital Signal Processing*, 22(1):34 – 46, 2012. 127, 186

I. Demirkol, C. Ersoy, and F. Alagoz. MAC protocols for wireless sensor networks: a survey. *IEEE Communications Magazine*, 44(4):115–121, April 2006. 4

S. Dias and M. Bruno. Cooperative particle filtering for emitter tracking with unknown noise variance. In *IEEE International Conf. on Acoustics, Speech and Signal Processing*, pages 2629–2632, 2012. 14, 83, 94, 95, 176

S. Dias and M. Bruno. Cooperative target tracking using decentralized particle filtering and RSS sensors. *IEEE Transactions on Signal Processing*, 61(14):3632–3646, 2013. 71, 73, 171

A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, LCN '04, pages 455–462, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2260-2. 4

P. Dutta and A. Dunkels. Operating systems and network protocols for wireless sensor networks. *Philosophical Transactions of the Royal Society A*, 370, January 2012. 4

J. Esteves, A. Carvalho, and C. Couto. Generalized geometric triangulation algorithm for mobile robot absolute self-localization. In *IEEE International Symposium on Industrial Electronics*, volume 1, pages 346–351, 2003. 10, 36

J. Esteves, A. Carvalho, and C. Couto. Position and orientation errors in mobile robot absolute self-localization using an improved version of the generalized geometric triangulation algorithm. In *IEEE International Conference on Industrial Technology*, pages 830–835, Dec 2006. 10

T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005. 48, 167

M. Farmani, H. Moradi, and M. Asadpour. A hybrid localization approach in wireless sensor networks using a mobile beacon and inter-node communication. In *IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 269–274, May 2012. 99, 110, 177, 182

D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesc language: A holistic approach to networked embedded systems. *SIGPLAN Not.*, 38(5), May 2003. 4

S. Gezici, Z. Tian, G. Giannakis, H. Kobayashi, A. Molisch, H. Poor, and Z. Sahinoglu. Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks. *IEEE Signal Processing Magazine*, 22(4):70–84, July 2005. 9

M. Gholami, R. Vaghefi, and E. Strom. Rss-based sensor localization in the presence of unknown channel parameters. *IEEE Trans. on Signal Processing*, 61(15):3752–3759, 2013. 36, 161

C. Gomez and J. Paradells. Wireless home automation networks: A survey of architectures and technologies. *IEEE Communications Magazine*, 48(6):92–101, 2010. 5

S. Gunn. Support vector machines for classification and regression. Technical Report, School of Electronics and Computer Science, University of Southampton, Southampton, U.K., 1998. 22, 47, 48

F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund. Particle filters for positioning, navigation, and tracking. *IEEE Transactions on Signal Processing*, 50(2):425–437, Feb 2002. 14, 70, 99, 171

T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher. Range-free localization schemes for large scale sensor networks. In *9th Annual International Conference on Mobile Computing and Networking*, MobiCom '03, pages 81–95, New York, NY, USA, 2003. ACM. ISBN 1-58113-753-2. 9

T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. *Annals of Statistics*, 36(3):1171–1220, 2008. 22, 139

P. Honeine, F. Mourad-Chehade, M. Kallas, H. Snoussi, H. Amoud, and C. Francis. Wireless sensor networks in biomedical: body area networks. In *Proc. 7th International Workshop on Systems, Signal Processing and their Applications*, pages 388–391, Algeria, 09–11 May 2011. 5, 6

P. Honeine, Z. Noumir, and C. Richard. Multiclass classification machines with the complexity of a single binary classifier. *Signal Processing*, 93(5):1013–1026, May 2013. 31, 33, 38, 48, 163, 167

L. Hou and N. Bergmann. Novel industrial wireless sensor networks for machine condition monitoring and fault diagnosis. *IEEE Transactions on Instrumentation and Measurement*, 61(10):2787–2798, Oct 2012. 6

A. Ickowicz, F. Septier, and P. Armand. Estimating a cbrn atmospheric release in a complex environment using gaussian processes. In *15th International Conference on Information Fusion (FUSION)*, pages 1846–1853, July 2012. 127, 162, 186

S. Jardosh and P. Ranjan. A survey: Topology control for wireless sensor networks. In *International Conference on Signal Processing, Communications and Networking*, pages 422–427, Jan 2008. 5, 160

S. Jayamohan and M. Mathurakani. Noise tolerance analysis of marginalized particle filter for target tracking. In *Annual International Conference on Emerging Research Areas and International Conference on Microelectronics, Communications and Renewable Energy (AICERA/ICMiCR)*, pages 1–6, 2013. 71, 74

S. Julier and J. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, Mar 2004. 14, 70, 171

S. J. Julier and J. K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *AeroSense: 11th Int. Symp. Aerospace/Defense Sensing, Simulation and Controls*, pages 182–193, 1997. 14, 70, 170

K. Kaemarungsi and P. Krishnamurthy. Modeling of indoor positioning systems based on location fingerprinting. In *Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 2, pages 1012–1022, March 2004. 9

A. S. Kalelkar. Investigation of large-magnitude incidents: Bhopal as a case study. In *Conference On Preventing Major Chemical Accidents, Institute of chemical engineers*, May 1988. 15, 126, 162

R. E. Kalman. A new approach to linear filtering and prediction problems. *J. of Basic Engineering*, 82(1):35–45, 1960. 14, 70, 79, 161, 170

P. Kathirgamanathan, R. Mckibbin, and R. I. Mclachlan. Source term estimation of pollution from an instantaneous point source. *Research Letters in the Information and Mathematical Sciences*, 3:59–67, 2002. 126, 129, 132, 143, 149, 151, 185, 187, 191, 192

S. Keerthi, S. Shevade, C. Bhattacharyya, and K. Murthy. Improvements to platt's smo algorithm for svm classifier design. Technical report, National University of Singapore, 1999. 48

G. Kimeldorf and G. Wahba. Some results on tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33:82 – 95, 1971. 30, 43, 165

H. Koyuncu and S. H. Yang. A 2D positioning system using WSNs in indoor environment. *International Journal of Electrical and Computer Sciences IJECS-IJENS*, 11 (3), 2011. 11, 37, 64, 170

A. Kushki, K. Plataniotis, and A. Venetsanopoulos. Kernel-based positioning in wireless local area networks. *IEEE Transactions on Mobile Computing*, 6(6):689–705, 2007. 37, 53, 65, 66, 170

R. Lara-Cueva, D. Benitez, A. Caamano, M. Zennaro, and J. Rojo-Alvarez. Performance evaluation of a volcano monitoring system using wireless sensor networks. In *IEEE Latin-America Conference on Communications*, pages 1–6, Nov 2014. 6

E.-E.-L. Lau and W.-Y. Chung. Enhanced RSSI-based real-time user location tracking system for indoor and outdoor environments. In *International Conference on Convergence Information Technology*, pages 1213–1218, 2007. 13, 161

S. H. Lee, S. Lee, H. Song, and H. S. Lee. Wireless sensor network design for tactical military applications : Remote large-scale environments. In *Military Communications Conference*, pages 1 –7, 2009. 5, 7, 160

P. Levis. TinyOS programming. Technical report, 2006. 4

P. Levis, S. Madden, J. Polastre, R. Szewczyk, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. TinyOS: An operating system for sensor networks. In *Ambient Intelligence*. Springer Verlag, 2004. 4

D. Li, K. Wong, Y. H. Hu, and A. Sayeed. Detection, classification, and tracking of targets. *IEEE Signal Processing Magazine*, 19(2):17–29, 2002. 12, 70, 170

Y. Li and J. Li. Robust adaptive kalman filtering for target tracking with unknown observation noise. In *24th Chinese Control and Decision Conference (CCDC)*, pages 2075–2080, 2012. 71, 79, 171

T.-N. Lin and P.-C. Lin. Performance comparison of indoor positioning techniques based on location fingerprinting in wireless networks. In *Wireless Networks, Communications and Mobile Computing, International Conference on*, volume 2, pages 1569 – 1574, 2005. 11, 37, 163

D. Liu, Y. Xiong, and J. Ma. Exploit kalman filter to improve fingerprint-based indoor localization. In *International Conference on Computer Science and Network Technology*, volume 4, pages 2290–2293, 2011. 71, 73, 83, 121

Y. Liu, Z. Yang, X. Wang, and L. Jian. Location, localization, and localizability. *Journal of Computer Science and Technology*, 25(2):274–297, 2010. 8, 160

Q. Mamun. A qualitative comparison of different logical topologies for wireless sensor networks. *Sensors*, 12(11):14887–14913, 2012. 5, 160

D. Manolakis. Efficient solution and performance analysis of 3-D position estimation by trilateration. *IEEE Transactions on Aerospace and Electronic Systems*, 32(4):1239–1248, 1996. 10, 36, 99, 177

G. Mao, B. Fidan, and B. D. O. Anderson. Wireless sensor network localization techniques. *Computer Networks*, 51(10):2529–2553, 2007. 11, 64

S. Maqbool and N. Chandra. Real time wireless monitoring and control of water systems using zigbee 802.15.4. In *International Conference on Computational Intelligence and Communication Networks*, pages 150–155, Sept 2013. 6

A. Medeisis and A. Kajackas. On the use of the universal Okumura-Hata propagation prediction model in rural areas. *Vehicular Conference Proceedings*, 3, 2000. 9, 54, 83, 98, 102, 115, 161, 168, 177, 179

C. A. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Computation*, 17(1):177–204, 2005. 48, 167

F. Mourad. *Auto-localisation et suivi de cibles dans les réseaux de capteurs mobiles*. PhD thesis, University of technology of Troyes, France, December 2010. 5, 160

F. Mourad, H. Snoussi, F. Abdallah, and C. Richard. Anchor-based localization via interval analysis for mobile ad-hoc sensor networks. *IEEE Trans. on Signal Processing*, 57(8):3226–3239, 2009. 37, 163

F. Mourad, H. Snoussi, and C. Richard. Interval-based localization using RSSI comparison in manets. *IEEE Transactions on Aerospace and Electronic Systems*, 47(4): 2897–2910, 2011. 37, 163

F. Mourad, P. Honeine, and H. Snoussi. Polar-interval-based localization in mobile sensor networks. *IEEE Transactions on Aerospace and Electronic Systems*, 49(4): 2310–2322, 2013. 37

A. Neskovic, N. Neskovic, and G. Paunovic. Modern approaches in modeling of mobile radio systems propagation environment. *IEEE Communications Surveys Tutorials*, 3 (3):2–12, Third 2000. 99, 115, 177

D. Niculescu and B. Nath. Ad hoc positioning system (APS) using AOA. In *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 3, pages 1734–1743 vol.3, March 2003. 10, 161

N. Okello, F. Fletcher, D. Musicki, and B. Ristic. Comparison of recursive algorithms for emitter localisation using TDoA measurements from a pair of uavs. *IEEE Transactions on Aerospace and Electronic Systems*, 47(3):1723–1732, 2011. 9, 36, 161

J. Ong, Z. You, J. Mills-Beale, E. L. Tan, B. Pereles, and K. G. Ong. A wireless, passive embedded sensor for real-time monitoring of water content in civil engineering materials. *IEEE Sensors Journal*, 8(12):2053–2058, Dec 2008. 7

O. Ozdemir, R. Niu, and P. Varshney. Tracking in wireless sensor networks using particle filtering: Physical layer considerations. *IEEE Trans. on Signal Processing*, 57(5):1987–1999, 2009. 37, 163

A. Pal. Localization algorithms in wireless sensor networks: current approaches and future challenges. *Network Protocols and Algorithms*, 2(1):45–76, 2010. 9, 161

N. Patwari, J. Ash, S. Kyperountas, A. Hero, R. Moses, and N. Correal. Locating the nodes: cooperative localization in wireless sensor networks. *IEEE Signal Processing Magazine*, 22(4):54–69, July 2005. 8, 9, 98, 102, 115, 179

H. Ramamurthy, B. S. Prabhu, R. Gadh, and A. Madni. Wireless industrial monitoring and control using a smart sensor platform. *IEEE Sensors Journal*, 7(5):611–618, 2007. 5

T. Reusing. Comparison of Operating Systems TinyOS and Contiki. *Sensor Nodes–Operation, Network and Application (SN)*, 7, 2012. 4

P. Rong and M. Sichitiu. Angle of arrival localization for wireless sensor networks. In *3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks. SECON*, volume 1, pages 374 –382, sept. 2006. 10, 36, 161

G. F. S. De Vito. Wireless chemical sensor networks for air quality monitoring. In *14th International Meeting on Chemical Sensors - IMCS 2012*, pages 641 – 644, may 2012. 5, 6, 160

H. Saeed, S. Ali, S. Rashid, S. Qaisar, and E. Felemban. Reliable monitoring of oil and gas pipelines using wireless sensor network (wsn)-remong. In *International Conference on System of Systems Engineering*, pages 230–235, June 2014. 6

F. Salvadori, M. De Campos, P. Sausen, R. De Camargo, C. Gehrke, C. Rech, M. Spohn, and A. Oliveira. Monitoring in industrial systems using wireless sensor network with dynamic power management. *IEEE Transactions on Instrumentation and Measurement*, 58(9):3104–3111, 2009. 5

A. Sandeep, Y. Shreyas, S. Seth, R. Agarwal, and G. Sadashivappa. Wireless network visualization and indoor empirical propagation model for a campus wi-fi network. *World Academy of Science, Engineering and Technology*, 2(6):706 – 710, 2008. 99, 177

T. Sarkar, Z. Ji, K. Kim, A. Medouri, and M. Salazar-Palma. A survey of various propagation models for mobile communication. *IEEE Antennas and Propagation Magazine*, 45(3):51–82, June 2003. 99

C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the 15th International Conference on Machine Learning*, pages 515–521. Morgan Kaufmann, 1998. 32, 33, 38, 43, 163, 165

A. Sayed, A. Tarighat, and N. Khajehnouri. Network-based wireless location: challenges faced in developing techniques for accurate wireless location information. *IEEE Signal Processing Magazine*, 22(4):24–40, July 2005. 8

B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *Proc. of the 14th Annual Conference on Computational Learning Theory and 5th European Conference on Computational Learning Theory*, pages 416–426, London, UK, 2001a. Springer-Verlag. ISBN 3-540-42343-5. 30, 43, 165, 180

B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Comput.*, 13(7): 1443–1471, July 2001b. 33, 133, 188

S. Seidel and T. Rappaport. 914 mhz path loss prediction models for indoor wireless communications in multifloored buildings. *IEEE Transactions on Antennas and Propagation*, 40(2):207–217, Feb 1992. 99, 177

A. Shareef and Y. Zhu. Localization using extended kalman filters in wireless sensor networks. *Graduate Student Scholarly and Creative Submissions. Paper 5*, 2009. 99, 177

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521813972. 24, 28, 29

S. Shin, T. Kwon, G.-Y. Jo, Y. Park, and H. Rhy. An experimental study of hierarchical intrusion detection for wireless industrial sensor networks. *IEEE Transactions on Industrial Informatics*, 6(4):744–757, Nov 2010. 6

A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, Aug. 2004. 47, 48, 166

K. Sohraby, D. Minoli, and T. Znati. *Wireless Sensor Networks: Technology, Protocols, and Applications*. John Wiley, 2007. 2, 159

M. Stone. Cross-validatory choice and assessment of statistical predictions. *J. Royal Stat. Soc.*, 36(2):111 – 147, 1974. 53, 83, 145

N. Suryadevara and S. Mukhopadhyay. Wireless sensor network based home monitoring system for wellness determination of elderly. *IEEE Sensors Journal*, 12(6):1965–1972, 2012. 7, 160

J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Process. Lett.*, 9(3):293–300, June 1999. 45, 166

D. Svensson. *Target tracking in complex scenarios.* PhD thesis, Chalmers University of Technology, Göteborg, Sweden, 2010. 13, 70, 161, 170

D. M. J. Tax and R. P. W. Duin. Support vector data description. *Mach. Learn.*, 54(1): 45–66, Jan. 2004. 33, 127, 186, 188

J. Teng, H. Snoussi, and C. Richard. Decentralized variational filtering for target tracking in binary sensor networks. *IEEE Trans. Mob. Comput.*, 9(10):1465–1477, 2010. 37, 163

A. N. Tikhonov and V. Y. Arsenin. *Solutions of ill-posed problems.* V. H. Winston & Sons, Washington, D.C.: John Wiley & Sons, New York, 1977. Translated from the Russian, Preface by translation editor Fritz John, Scripta Series in Mathematics. 23, 24

D. Trinchero, R. Stefanelli, D. Brunazzi, A. Casalegno, M. Durando, and A. Galardini. Integration of smart house sensors into a fully networked (web) environment. In *IEEE Sensors*, pages 1624–1627, Oct 2011. 6

A. T. Tu. Overview of Sarin terrorist attacks in Japan. In *Natural and Selected Synthetic Toxins*, pages 304–317, December 1999. 15, 126, 162

A. UmaMageswari, J. Joseph Ignatious, and R. Vinodha. A comparative study of Kalman filter, extended Kalman filter and unscented Kalman filter for harmonic analysis of the non-stationary signals. *International Journal of Scientific & Engineering Research*, 3, July 2012. 14, 70, 99, 171, 177

V. N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag New York, Inc., New York, NY, USA, 1995. ISBN 0-387-94559-8. 22, 33, 38, 47, 163, 166

V. N. Vapnik. *Statistical Learning Theory.* John Wiley and Sons, 1998. 22

J. P. Vert, K. Tsuda, and B. Scholkopf. A primer on kernel methods. *Kernel Methods in Computational Biology*, pages 35–70, 2004. 24, 25, 28

M. Vieira, C. Coelho, J. da Silva, D.C., and J. da Mata. Survey on wireless sensor network devices. In *IEEE Conference on Emerging Technologies and Factory Automation*, volume 1, pages 537–544 vol.1, Sept 2003. 2, 159

H. Wang, H. Lenz, A. Szabo, J. Bamberger, and U. Hanebeck. Wlan-based pedestrian tracking using particle filters and low-cost mems sensors. In *Positioning, Navigation and Communication, 2007. WPNC '07. 4th Workshop on*, pages 1–7, March 2007. 99, 177

L. Wang, Y. Liu, X. Xu, and X. Wang. Wsn multilateration algorithm based on landweber iteration. In *International Conference on Electronic Measurement Instruments*, pages 1–250–1–254, August 2009. 10, 115

Y. Wang, X. Jia, H. K. Lee, and G. Y. Li. An indoors wireless positioning system based on wireless local area network infrastructure. *Technology Including Mobile Positioning*, 2003. 99, 177

N. Watthanawisuth, A. Tuantranont, and T. Kerdcharoen. Design for the next generation of wireless sensor networks in battlefield based on zigbee. In *Defense Science Research Conference and Expo*, pages 1–4, Aug 2011. 7, 160

G. Welch and G. Bishop. An Introduction to the Kalman Filter. http://www.cs.unc.edu, UNC-Chapel Hill, TR95-041, February 2001. 14, 70, 79, 161, 170

J. Wendeberg, J. Muller, C. Schindelhauer, and W. Burgard. Robust tracking of a mobile beacon using time differences of arrival with simultaneous calibration of receiver positions. In *Int. Conf. on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–10, 2012. 13

Z.-L. Wu, C.-H. Li, J.-Y. Ng, and K. Leung. Location estimation via support vector regression. *IEEE Transactions on Mobile Computing*, 6(3):311–321, 2007. 37

E. Xu, Z. Ding, and S. Dasgupta. Target tracking and mobile sensor navigation in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 12(1):177–186, 2013. 13, 161

Y. Z. Y. Shang, W. Ruml and M. P. J. Fromherz. Localization from mere connectivity. *MobiHoc'03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking and computing*, pages 201–212, 2003. 10, 37, 64, 163, 169

J. Yang and Y. Chen. Indoor localization using improved rss-based lateration methods. In *IEEE Global Telecommunications Conference GLOBECOM.*, pages 1–6, Nov 2009. 99, 111, 177

L. Yu, N. Wang, and X. Meng. Real-time forest fire detection with wireless sensor networks. In *Wireless Communications, Networking and Mobile Computing. Proc. International Conference on*, volume 2, pages 1214 – 1217, Sept. 2005. 5, 6, 160

M. Zakrzewski, S. Junnila, A. Vehkaoja, H. Kailanto, A.-M. Vainio, I. Defee, J. Lekkala, J. Vanhala, and J. Hyttinen. Utilization of wireless sensor network for health monitoring in home environment. In *IEEE International Symposium on Industrial Embedded Systems*, pages 132–135, July 2009. 7

G. Zanca, F. Zorzi, and A. Zanella. RSSI measurements in indoor wireless sensor networks. 53, 60, 114

A. Zanella and A. Bardella. Rss-based ranging by multichannel rss averaging. *IEEE Wireless Communications Letters*, 3(1):10–13, February 2014. 9, 98, 115, 177

L. Zhang, Y. H. Chew, and W.-C. Wong. A novel angle-of-arrival assisted extended kalman filter tracking algorithm with space-time correlation based motion parameters estimation. In *9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 1283–1289, 2013. 13, 161

Sandy MAHFOUZ

Doctorat : Optimisation et Sûreté des Systèmes

Année 2015

Méthodes à noyaux pour le suivi de cibles et la surveillance de l'environnement dans les réseaux de capteurs

Cette thèse porte sur les problèmes de localisation et de surveillance de champ de gaz à l'aide de réseaux de capteurs sans fil. Nous nous intéressons d'abord à la géolocalisation des capteurs et au suivi de cibles. Nous proposons ainsi une approche exploitant la puissance des signaux échangés entre les capteurs et appliquant les méthodes à noyaux avec la technique de fingerprinting. Nous élaborons ensuite une méthode de suivi de cibles, en se basant sur l'approche de localisation proposée. Cette méthode permet d'améliorer la position estimée de la cible en tenant compte de ses accélérations, et cela à l'aide du filtre de Kalman. Nous proposons également un modèle semi-paramétrique estimant les distances inter-capteurs en se basant sur les puissances des signaux échangés entre ces capteurs. Ce modèle est une combinaison du modèle physique de propagation avec un terme non linéaire estimé par les méthodes à noyaux. Les données d'accélérations sont également utilisées ici avec les distances, pour localiser la cible, en s'appuyant sur un filtrage de Kalman et un filtrage particulaire. Dans un autre contexte, nous proposons une méthode pour la surveillance de la diffusion d'un gaz dans une zone d'intérêt, basée sur l'apprentissage par noyaux. Cette méthode permet de détecter la diffusion d'un gaz en utilisant des concentrations relevées régulièrement par des capteurs déployés dans la zone. Les concentrations mesurées sont ensuite traitées pour estimer les paramètres de la source de gaz, notamment sa position et la quantité du gaz libéré.

Mots clés : réseaux de capteurs (technologie) - apprentissage automatique - modèles non linéaires (statistique) - traitement du signal - Kalman, filtrage de.

Kernel-based Machine Learning for Tracking and Environmental Monitoring in Wireless Sensor Networks

This thesis focuses on the problems of localization and gas field monitoring using wireless sensor networks. First, we focus on the geolocalization of sensors and target tracking. Using the powers of the signals exchanged between sensors, we propose a localization method combining radio-location fingerprinting and kernel methods from statistical machine learning. Based on this localization method, we develop a target tracking method that enhances the estimated position of the target by combining it to acceleration information using the Kalman filter. We also provide a semi-parametric model that estimates the distances separating sensors based on the powers of the signals exchanged between them. This semi-parametric model is a combination of the well-known log-distance propagation model with a non-linear fluctuation term estimated within the framework of kernel methods. The target's position is estimated by incorporating acceleration information to the distances separating the target from the sensors, using either the Kalman filter or the particle filter. In another context, we study gas diffusions in wireless sensor networks, using also machine learning. We propose a method that allows the detection of multiple gas diffusions based on concentration measures regularly collected from the studied region. The method estimates then the parameters of the multiple gas sources, including the sources' locations and their release rates.

Keywords: sensor networks - machine learning - non-linear models - signal processing – Kalman filtering.

Thèse réalisée en partenariat entre :

RÉGION CHAMPAGNE ARDENNE

utt université de technologie Troyes

Ecole Doctorale "Sciences et Technologies"