Sequential Monte Carlo Radio-Frequency Tomographic Tracking

Xi Chen

Department of Electrical & Computer Engineering
McGill University
Montreal, Canada

June 2011

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Master of Engineering.

Abstract

Target tracking in over a small-scale area using wireless sensor networks (WSNs) is a technique that can be used in applications ranging from emergency rescue after an earthquake to security protection in a building. Many target tracking systems rely on the presence of an electric device which must be carried by the target in order to reports back its location and status. This makes these systems unsuitable for many emergency applications; in such applications *device-free* tracking systems that where no devices are attached to the targets are needed. Radio-Frequency (RF) tomographic tracking is one such device-free tracking technique. This system tracks moving targets by analyzing changes in attenuation in wireless transmissions. The target can be tracked within the sensor network area without being required to carry an electric device.

Some previously-proposed device-free tracking approaches require a time-consuming training phase before tracking can be carried out, which is time-consuming. Others perform tracking by sacrificing part of the estimation accuracy. In this thesis, we propose a novel sequential Monte Carlo (SMC) algorithm for RF tomographic tracking. It can track a single target moving in a wireless sensor network without the system needing to be trained. The algorithm adopts a particle filtering method to estimate the target position and incorporates on-line Expectation Maximization (EM) to estimate model parameters. Based on experimental measurements, the work also introduces a novel measurement model for the attenuation caused by a target with the goal of improving estimation accuracy. The performance of the algorithm is assessed through numerical simulations and field experiments carried out with a wireless sensor network testbed. Both simulated and experimental results demonstrate that our work outperforms previous RF tomographic tracking approaches for single target tracking.

Abrégé

Suivi de cible dans la zone petite chelle en utilisant les rseaux de capteurs sans fil est une technique qui peut tre largement utilis dans des applications telles que le sauvetage d'urgence aprs un tremblement de terre, ou la protection de la scurit dans un btiment. Beaucoup de systmes de poursuite de cibles ncessitent un dispositif lectrique ralise par l'objectif de faire rapport de ses localisation instantane et le statut. L'inconvnient rend ces systmes ne conviennent pas pour des applications nombreuses interventions d'urgence, em dispositif sans systmes de suivi qui ne les priphriques connects sur les objectifs sont ncessaires. Radio-Frquence (RF) suivi tomographique est l'une des techniques dispositif de suivi-libres. Il s'agit d'un processus de suivi des cibles mobiles en analysant l'volution de l'attnuation dans les transmissions sans fil. La cible peut tre suivi dans la zone de rseau de capteurs, tandis que les appareils lectriques ne doivent tre effectus. Cependant, certaines approches prcdentes dispositif de suivi-libre ncessite une phase d'entranement avant de suivi, ce qui prend beaucoup de temps. Autres effectuer un suivi par scarification partie de prcision de l'estimation.

Dans cette thse, nous proposons une nouvelle Monte Carlo squentielles (SMC) algorithme de suivi RF tomographique. Il peut suivre une cible unique sans formation du systme dans un rseau de capteurs sans fil. L'algorithme de filtrage particulaire adopte la mthode pour estimer la position cible et intgre en ligne Expectation Maximization (EM) pour estimer les paramtres du modle. Sur la base de mesures exprimentales, le travail introduit galement un modle de mesure de roman pour l'attnuation provoque par une cible pour amliorer la prcision d'estimation. La performance de l'algorithme est value par des simulations numriques et expriences sur le terrain avec un rseau de capteurs sans fil banc d'essai. Les deux rsultats simuls et exprimentaux dmontrent que notre travail surpasse prcdente approche RF suivi tomographique pour le suivi de cible unique.

Acknowledgments

First and foremost, I am heartily thankful to my supervisor, Prof. Mark Coates, for his continuous help, guidance and support which have made my graduate studies at McGill University an enjoyable and educational experience. I have greatly benefitted from his constructive comments and feedbacks on my research work. His encouragement and patient advice have been invaluable.

Many thanks go to Prof. Michael Rabbat, for his kind, patient help and insightful comments in the project.

I would also like to thank members in the RF tomography project, especially Andrea, Yunpeng and Abhay. The positive work atmosphere lead to friendly collaboration, discussion and high work efficiency. Special thanks go to Ali for translating the abstract into French and to Andrea and Fred for proofreading my manuscript.

I want to express my appreciation to my friends and colleagues in the Computer Networks group, for the many friendships which have developed during this two years at McGill, especially to Zhe, Xuan, Abhay, Deniz, Haani, Boris, Hong, Salim, Konstantinos, Judith, Ali and Bassel. It has been my great pleasure to work among them.

Finally, my heartfelt gratitude goes to my dear parents. They were always there for me, with all the support they can give. Thanks for everything they did for me. And of course I am grateful to my girlfriend Yuxi, for all the love, happiness and wonderful moments we experienced together.

Contents

List of Figures

List of Tables

List of Acronyms

WSN	Wireless Sensor Network
RF	Radio-Frequency
SMC	Sequential Monte Carlo
RSS	Received Signal Strength
EM	Expectation and Maximization
ADC	Analog-to-Digital Converter
RSSI	Received Signal Strength Indicator
GPS	Global Positioning System
DFP	Device-Free Passive
RFID	Radio-Frequency IDentification
WLAN	Wireless Local Area Network
UWB	Ultra-Wide Band
LOS	Line-Of-Sight
NLOS	Non-Line-Of-Sight
APs	Access Points
SVM	Support Vector Machine
RTI	Radio Tomographic Imaging
KF	Kalman Filter
SIS	Sequential Importance Sampling
SIR	Sampling Importance Resampling
APF	Auxiliary Particle Filter
MCMC	Markov Chain Monte Carlo
RMSE	Root Mean Square Error

Chapter 1

Introduction

1.1 Motivation

A wireless sensor network (WSN) consists of several sensor nodes that can collect information related to the surrounding environment such as pressure, temperature, humidity, etc [3]. Sensors are able to communicate and cooperate with each other to finish their tasks using wireless communication techniques. In a wireless sensor network, signal attenuation occurs when a target moves between nodes in the network. For Radio-Frequency (RF) links connecting different pairs of nodes, the amount of attenuation caused by the target varies based on the proximity of the target to the links. These time-varying patterns of link attenuation provide information about the target location, allowing the network to track the target's motion. This procedure is referred as RF tomography.

Many target tracking systems require that small electric devices be attached to the tracked object. These devices are able to continuously communicate with the system and to report on the status of the tracked object. However, this type of system cannot satisfy the requirements of many applications (such as battlefield surveillance) where no devices can be carried by the tracked objects. An alternative way to solve this problem is to use RF tomography techniques.

RF tomography is a promising technique with many practical applications since it doesn't require that any electric devices be carried by the tracked target. For example, it is often difficult for first-responders in a disaster situation, such as during a fire or after an earthquake, to locate survivors. Quickly locating the survivors without rescuers needing to manually enter any structures to look for them will save both time and, potentially, the

survivors' lives. It can also reduce the responders' casualty rates. Similarly, RF tomography can also be used for security and surveillance applications such as through-the-wall imaging and perimeter monitoring. RF tomography could enable a smart home to control lighting, heating, and air-conditioning in order to save power when there are no people in a room. It can also be used by doctors to track elderly people and other patients remotely, either at home or at a hospital, without violating their privacy, providing the patient with comfortable and secure living conditions. Using wireless sensors also reduces the cost of the system in practical deployments, especially in small scale target tracking scenarios, since such a system only requires several low cost sensors and a laptop to process the data. Furthermore, it only takes a few minutes to place sensors around the area that needs to be monitored. The ability to easily and quickly deploy a network makes RF tomography useful in the emergency response scenarios mentioned above.

1.2 Thesis Problem Statement

RF tomographic tracking techniques analyze changes in RF signals. However, the transmission power of RF signals is restricted due to the limited battery power supply of sensor nodes. Thus, it is a challenge to abstract useful information from the received RF signals which are affected by background noise. Moreover, to avoid a time-consuming training phase during network deploying period, the RF tomographic technique uses must use real-time signals transmitted in the sensor network as the only reference source for tracking, which makes it even harder to obtain accurate estimations.

Previous RF tomographic systems [84] only image signal changes in an environment and identify peak points of the changes as the estimated locations of targets. These systems do not track the targets. This approach suffers from both an estimation accuracy problem and a robustness problem when the environment changes. Our goal is to find an RF tomography approach that enables our system to perform tracking without a training phase and to estimate target trajectories with high accuracy and robustness in various environments.

1.3 Thesis Contribution and Organization

Chapter 2 provides some background knowledge for readers to better understand the thesis work. We first introduce wireless sensor networks and the TelosB hardware platform that

we used in our field experiments. Afterwards, we summarize previous work related to tracking systems, including both device-based systems and device-free systems. Finally, we focus on the RF tomographic tracking systems that we will further explore in this thesis.

Chapter 3 proposes a novel algorithm for RF tomographic tracking. Unlike existing RF tomography schemes, it adopts a Sequential Monte Carlo (SMC) approach to track the motion of the target. In order to make the algorithm computationally efficient and to improve tracking accuracy, we introduce a new measurement model for the algorithm. The model is validated by experimental data. It does not quantize the region of interest into pixels, in contrast to the measurement models of [83, 84]. Our algorithm incorporates an on-line expectation maximization procedure to estimate unknown parameters of both the dynamic model and the measurement model which are used in the particle filtering. These parameters can vary significantly for different targets and environments, so the on-line expectation maximization algorithm provides an important self-calibration mechanism.

Chapter 4 provides both simulation and experimental results to validate our approach. The experiments are conducted with a wireless sensor network testbed set up on the McGill campus. We also provide results using previous RF tomographic imaging method as a comparison.

Chapter 5 summarizes the work and discusses potential future work.

1.4 Author's Work

Two papers based on content presented in this thesis will be published in the following international conference proceedings:

- Yunpeng Li*, Xi Chen*, Mark Coates, Bo Yang, "Sequential Monte Carlo Radio-Frequency Tomographic Tracking," in *Proc. Intl. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, to appear, May 2011. ***Equal first-authors.***

- Xi Chen, Andrea Edelstein, Yunpeng Li, Mark Coates, Michael Rabbat, Aidong Men, "Sequential Monte Carlo for Simultaneous Passive Device-Free Tracking and Sensor Localization Using Received Signal Strength Measurements," in *Proc. ACM/IEEE Intl. Conf. Information Processing in Sensor Networks (IPSN)*, to appear, Apr. 2011.

Chapter 2

Literature Review

In recent years, many different tracking algorithms based on wireless sensor networks have been developed by researchers. Some of the applications such as human motion detection, security in critical infrastructures, environmental and traffic monitoring have already been implemented in practical environments. Radio-Frequency target tracking using wireless sensor networks is one of the emerging technologies in the tracking research area. It utilizes the attenuation characteristic of RF signals to locate the motion of moving targets [11,84]. This chapter provides background knowledge on wireless sensor networks and the hardware platform we used in our experiments. Also, we review state-of-the-art research in the RF tracking field. Section 2.1 introduces wireless sensor networks, and then focuses on the TelosB sensor. Section 2.2 provides a more detailed discussion on the previous research that addresses the target tracking problem with wireless sensor networks, emphasizing RF tracking based on Received Signal Strength (RSS). It discusses the advantages and disadvantages of different tracking approaches. Section 2.3 introduces sequential Monte Carlo (particle filtering) tracking algorithms and discusses the performances of them. Section 2.4 introduces static parameter estimation and focuses on a on-line Expectation and Maximization (EM) algorithm. Both particle filtering and on-line EM approaches are used in our tracking algorithm.

2.1 Wireless Sensor Networks

2.1.1 Overview

Wireless sensor networks are constructed using multiple sensor nodes capable of collecting useful information from their surrounding environment [3]. In most cases, the nodes have limited resources, especially computing and storage capabilities and power supply. Networking a large number of wireless sensor nodes to collect, distribute and fuse information for a specific objective in different environments is an emerging technique in recent years [3, 53]. Rapid progress in the research of sensor networks has led to decreases in the price and increases in the capabilities of the sensors, making deployment increasingly practical. Nowadays, wireless sensor networks with low cost and high processing abilities are being explored [9, 68]. The capability of a sensor node to exchange information with its neighbor nodes and intelligently process the data makes sensor networks even more useful in some critical environments not suitable for human beings such as volcanos and earthquake sites [14, 52].

Sensor nodes can be roughly divided into two classes: passive sensors and active sensors [53]. Passive sensors generate electrical signals in response to the information sensed from the environment and they do not require additional power supply. Active sensors can actively sense the environment with a relatively higher sensing frequency, but they require continuous external power for their operation. A typical sensor node consists of four basic components: (1) a power unit that supports the sensor to perform all operations; (2) a transceiver unit that transmits and receives the data between itself and other sensor nodes; (3) a sensing unit that collects environmental measurements and uses an Analog-to-Digital Converter (ADC) that translate the analog signals into digital signals; and (4) a processing unit that stores and pre-processes the raw data [3].

In wireless sensor networks, sensors are connected by a wireless medium such as radio or infrared. Sensor network architectures can be broadly classified as centralized, distributed and hierarchical architectures [74]. Centralized wireless sensor networks have a sink node, also known as a data fusion center or root node. All pre-processed data from other sensor nodes are sent and fused at this sink node. Therefore the sink node needs to be a high performance computing unit and requires enhanced communication bandwidth and robustness. Distributed wireless sensor networks do not have this centralized unit and all the sensor nodes are capable of exchanging information locally with their neighbors and

processing the data themselves. Such an architecture is more robust but all sensor nodes need to have more computational capability to perform these sophisticated tasks. The hierarchical wireless sensor network architecture integrates the former two architectures together; it consists of several clusters of sensor nodes and each cluster has a local cluster sink node. It operates as a centralized architecture within each cluster and as a distributed architecture among different local sink nodes. This architecture is useful for large-scale deployment applications in which it is necessary to collect and process a large amount of information.

There are many applications for wireless sensor networks: battlefield surveillance [3, 9], habitat monitoring [13], medical care and in-home monitoring of elderly patients [58, 68], highway automation [36] and infrastructure monitoring [86]. The purpose is to perform distributed event detection, tracking, and information collection in different environments. The limitations of battery power duration, computational and memory resources still are the main challenges for practical deployment. Another challenge is robustness and the adaptive capability of the sensor nodes; sensor nodes are easily lost, damaged or stolen, resulting in dynamic network topologies.

Several deployments and experiments using wireless sensor networks have focused on tracking applications. For example, in [33], He et al. described a large-scale sensor network system (named VigilNet) that used 200 XSM sensor nodes to perform target tracking, detection and classification. Wilson and Patwari used more than 30 TelosB nodes surrounding an area to sense the motion of human beings [83, 84]. Ahmed et al. deployed a grid of MicaZ nodes in a 2 meters $\times$ 3.5 meters small test environment for ground surveillance application [1].

2.1.2 TelosB and TinyOS

The Crossbow TelosB is one of the RF wireless sensor nodes used in the experiments described in this thesis (see Figure 2.1). The standards of TelosB we introduce in this subsection are described in [18].

The TelosB is an open source platform which has an 8MHz MSP430 micro-controller with 10kB RAM. Using a pair of AA batteries, its operating ambient temperature range is -40 to 85 $^{\circ}C$. There are 128 bytes in RAM for buffering the packets which need to be sent, and an extra 128 bytes for those that have been recently received. In serial mode without

buffering, the TelosB can transmit at a maximum data rate of 250 kbps. There are 8 steps of transmission power level, ranging from 0 dBm to -24 dBm and the transmission data latency is about 2 μs.

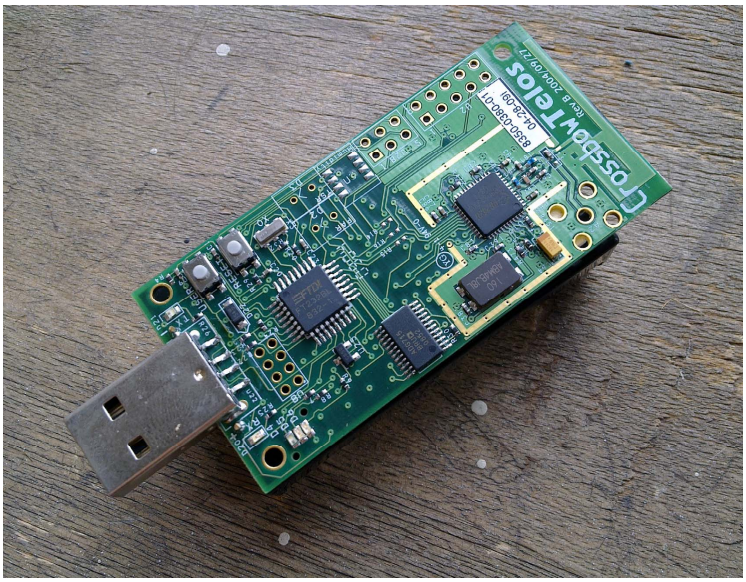


Fig. 2.1 TelosB Sensor Node

The TelosB mote conforms to the IEEE 802.15.4 [45] radio standard and operates in a frequency range between 2.4 and 2.4835 GHz. There are 16 channels within this band spaced in steps of 5 MHz. At the highest transmission power level, its claimed sensing range is 75-100 meters. With a maximum 127 bytes packet length, a packet includes 9 bytes of header, 0-20 bytes of address information and a 2 byte frame check sequence. The rest is frame payload.

The TelosB has a built-in RSSI (Received Signal Strength Indicator), which provides a digital value that can be read in a form of 8 bit, signed complement value. This can be converted into the corresponding decimal value RSSI_VAL and the Radio-Frequency power P in dBm can be calculated from the following equation: P = RSSI_VAL + RSSI_OFFSET, where the RSSI_OFFSET value is found empirically during system development. It has a typical value of approximately -45 dBm. For example, if a value of -20 is read from the RSSI register, the RF input power is approximately -65 dBm.

TelosB motes run TinyOS, an open source operating system designed for low-power wireless devices, such as those used in sensor networks, ubiquitous computing, personal

area networks, smart buildings, and smart meters [51]. It features a component-based architecture which enables rapid innovation and implementation. TinyOS minimizes code size as required by the severe memory constraints inherent in sensor networks. TinyOS's component library includes network protocols, distributed services, sensor drivers, and data acquisition tools. It was originally developed as a research project at the University of California Berkeley, but has since grown to have an international community of developers and users.

The programs for the motes are written in a language called nesC, which is an extension to programming language C, and is designed to embody the structuring concepts and execution model of TinyOS [28].

2.2 Tracking Systems

2.2.1 Overview

Generally, a tracking system is used to monitor or observe targets such as people, animals or vehicles by collecting and analyzing related information and display the motion of those targets. Tracking systems can be classified either as 'lag-time' systems such as the bar-code in the supermarket which requires a person to scan it, or 'real-time' systems such as the widely used satellite-based Global Positioning System (GPS) [35]. Some 'real-time' tracking systems [2, 11, 79, 80] use devices connected to local area networks or the Internet to perform tracking in an indoor area such as the home or office. However, GPS does not provide accurate estimates of location for indoor environments. Other tracking systems [14, 52, 54, 84, 89] utilize data collected by pre-deployed wireless sensor networks to estimate the motion of targets. Sensors are able to transmit and receive wireless radio signals that can travel through the air. Their power strength is affected by transmission distance and obstructions they travel through. Tracking systems can locate the moving target within the network sensing area by calculating the angle or distance based on features of them.

Tracking systems mainly use the following two techniques to perform target tracking: triangulation [34] and fingerprinting [72]. In two dimensions flat, triangulation technique computes a target's location by measuring its distance from at least three non-collinear reference points. As shown in Figure 2.2, Point A, B and C are three reference points and a target presents in the triangular area formed by these three points. Given the distance

between the target and three reference points, we can easily determine where the target is. There are three main schemes to calculate the distance in triangulation approaches [31,34]: time of arrival, attenuation and angle of arrival. Time of arrival scheme uses the different arrival times of sensor packets to calculate the distance between target and reference points. The attenuation scheme calculates the distance by measuring signal strength of emitted signals. The power level of a signal decreases as the distance increases, therefore it is possible to approximate the distance given a function by establishing a relation between distance and attenuation level. The angle of arrival scheme calculates angles between target and sensors to estimate target location. This scheme requires two angle measurements between reference points and the target, and also one distance measurement among reference points.

Fingerprinting is a two-step process for tracking using 802.11 wireless local area network. First, the RSS values of different locations are sensed and recorded during an off-line calibration procedure. The 'off-line' calibration is a procedure to calibrate important signal signatures of both sensed area and target to train the system. Second, these signatures are stored in this off-line phase for further comparison in on-line tracking phase. Targets may or may not carry a device to transmit signals; the signal patterns are recorded when a target stands in different locations. In the tracking phase, the real-time received signal strength are collected and compared with those stored signatures using a probabilistic tracking algorithm to estimate current locations.

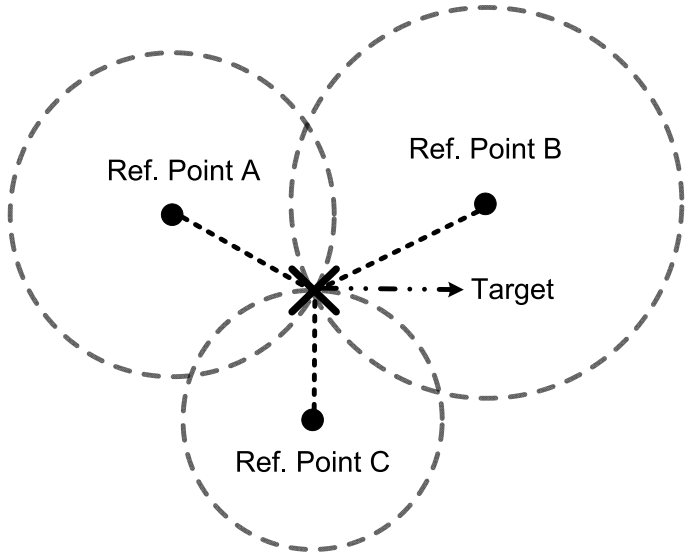


Fig. 2.2 Triangulation method in 2D tracking

Tracking systems can be divided into two categories depending on whether the targets carry tags or transceivers: device-based tracking systems and Device-Free Passive (DFP) tracking systems. Device-based techniques such as the Infrared-based systems [2,80] Radio-Frequency IDentification (RFID) detection system [79] and some RF-based systems [11], require the tracked targets to cooperate and carry devices which can transmit or reflect beacon signals. This approach is not applicable in many scenarios, e.g., emergency response situations like earthquakes or fires. The term "device-free passive" was coined by Youssef, Mah, and Agrawala [87] to describe tracking systems that do not require the target to carry any devices, including passive tags. Device-free passive systems use images, sounds or electric signals to detect and track targets moving through the sensing region.

2.2.2 Device-based Tracking Systems

A number of device-based systems have been developed. They can be classified according to the sensors that are employed: infrared, ultrasound, radio-frequency, magnetic, audible sound. The RF device-based systems can be further divided into RFID, Wireless Local Area Network (WLAN) and Ultra-Wide Band (UWB) categories. Some sensors are used in both device-based and device-free passive systems.

Infrared systems

Infrared tracking systems were one of the earliest tracking systems developed. Their performance is limited by the restriction that infrared sensors can only provide Line-Of-Sight (LOS) measurements. Early systems like Active Badge [80] can cover an area in a building and provide position information by estimating the location of each active badge. Another infrared indoor system, Scout [2], uses triangulation to calculate the location based on the measurement of arrival angles. It supports the localization of multiple targets with tags attached.

Infrared systems are easy to install and maintain, and the devices are cheap and well developed. However, the performance can be greatly degraded by sunlight and obstructions like walls and trees [26].

Ultrasound systems

Ultrasound tracking systems work at frequencies beyond the audible range of the human ear. They measure the changes of air pressure when there are obstructions present and perform calculations to determine the positions of the obstructions [32]. Priyantha et al. proposed Cricket, a location support system with ultrasound sensors installed on the wall or ceiling in indoor environment and receivers attached to each target [63], . Cricket measures the arrival times of two ultrasonic pulses from known beacons to calculate the location of a target; this system has been further developed to determine the target's orientation [64]. Ultrasound-based applications suffer a loss of coverage in Non-Line-Of-Sight (NLOS) scenarios; obstructions like walls can greatly attenuate the ultrasound signals [67].

Although there are other techniques that use similar air pressure measurements similar to the ultrasound systems, such as acoustic and vibration sensor systems [27], their performance is generally poorer. Environmental noise generated by machines such as fax machines and air-conditioners can greatly disturb the accuracy of these systems.

RF systems

As radio waves can travel through obstructions including walls and human bodies, the coverage of tracking systems using RF technology is less of a problem. RF systems often possess a coverage range of more than 50 meters $\times$ 50 meters. This remarkable feature makes RF a promising technology to be used in small-scale area tracking. We now introduce four types of device-based systems that use RF technology.

RFID

RFID tag is a microchip combined with an antenna in a small compact tag that can store identification information using electromagnetic techniques. RFID tracking systems are commonly used in indoor scenarios. Passive RFID tags [16] only act as a receiver, while active RFID tags [60] act as a transceiver. RFID systems were initially designed to replace identification technologies such as barcodes, but they are now employed for a diverse range of applications including animal tracking [78], indoor target tracking [31] and health-care applications [78].

For example, the WhereNet system [31] is an RFID-based tracking systems that uses a sophisticated differential time of arrival algorithm to calculate the locations of the tags and achieve multi-target tracking. It can be used in both indoor and outdoor real-time

tracking. The dimensions of the RFID tags in this system are 6.6 cm × 4.4 cm × 2.1 cm while the weight is 53 g. The operating duration of a single tag is about 7 years depending on the transmission rate of the RF signals which varies from every five seconds to every hour. However, this system suffers from an estimation error of 2 m to 3 m when tracking a target in indoor environment. Also, it requires a considerable number of components to be installed before operation. Voulodimos et al. proposed a farm management system for animal identification using RFID technology that can track animals and detect diseases [78] . Wang et al. developed a health-care tracking system using RFID in a hospital [79], . The system can help doctors track patients and improve the safety of a hospital.

WLAN

WLAN tracking systems intelligently utilize the existing WLAN infrastructures in indoor environments, which greatly reduce the cost of deployment.

RADAR [11] is a WLAN tracking system developed by Microsoft Research for indoor tracking. It is able to perform 2-D target tracking by using a triangulation location technique based on signal strength. RADAR is easy to deploy since it reuses the WLAN devices already installed in an indoor environment. On the other hand, this can be a limitation – the system can only be used in places where WLAN devices have been deployed.

COMPASS [44] is another WLAN tracking system that employs a location fingerprinting technique, a probabilistic tracking algorithm and a digital compass to measure the target's orientation. The digital compass minimizes the negative influence on tracking performance caused by the attenuation due to the human body.

The WLAN tracking systems are cost effective and reuse existing wireless devices. They can also be integrated in mobile devices such as laptops and cell phones. However, the complex environments consisting of different WLAN sources make the accuracy of WLAN tracking systems worse than other tracking systems using RF technologies. Additionally, the fingerprinting technique also needs a pre-training procedure that makes deployment of some of the WLAN tracking systems time-consuming.

UWB

Ultra-WideBand (UWB) systems can improve the accuracy of tracking by reducing the multi-path effects caused by obstructions. The Ubisense system [69] uses a triangulation scheme to calculate the target's position based on the difference of signals' arrival time. By measuring both the angles and arrival times of the signals, Ubisense has good performance in complex indoor environments with an accuracy of the order of tens of centimeters.

However, the Ubisense system needs a calibration procedure before running in a new environment, which prevents application in emergency scenarios.

Jourdan et al. [39] proposed a tracking system that uses UWB and a particle filter. The system collects the data using UWB with a bandwidth of 1 GHz. In the experiments reported, their UWB tracking system shows good performance when tracking target walking at a constant speed of $1m/sec$. The system achieves a reasonable good accuracy of trajectory estimation in through-wall tracking scenario.

UWB signals exhibit low multi-path distortion and can penetrate through obstructions. These features make UWB a good choice for indoor/through-wall tracking systems. Tracking systems using UWB have good performance in terms of accuracy and robustness.

RSS

Some tracking systems track targets by measuring received signal strength of the wireless sensor nodes. Emitted signals are easily affected by obstructions and distances. Thus, changes in received signal strength can be used to estimate target locations. It is an easy and cheap way to perform target tracking in wireless sensor networks.

In [58], Lorincz and Welsh developed 'MoteTrack', a tracking system that used Mica2 [4] sensor motes. Based on received signal strength, the system is able to track targets in a large-scale deployment using reference signatures collected in an off-line process. MoteTrack requires some beacon nodes to be fixed in known locations of the sensing area. Targets with a mobile node walk within this area to acquire reference signature sets that represent the target positions. These reference signatures are divided into subsets and distributed to beacon nodes. To perform tracking, new received signal strength are compared with the stored signatures to approximate the target location. Each beacon node only computes data based on the signature subset they locally stored. In the experiment, the MoteTrack system has been deployed in one floor of a building measuring approximately 1742 square meters. The results show a relatively high estimation error: among 74 separate location estimation experiments, only 20 percent of them achieve an error of less than 1 meter. The rest of the experiments suffer from errors from 1 meter to 7 meters.

An et al. presented 'Online Person Tracking', an indoor tracking system using TelosB sensor nodes in [5] . It has the same deployment procedure as MoteTrack; a calibration step is performed using mobile node carried by a target to collect the signature of the received signal strength. Target tracking is performed based on the reference signature data. The experimental testbed is a $70m \times 12m$ floor with a narrow corridor of $60m \times 2m$.

Over 30 independent experiments, 25 percent of them achieve an error of 0.6 meters and 75 percent of them had an error smaller than 1.7 meters.

In [85], Woyach et al. analyzed the physical phenomena like fading and shadowing loss that may affect the strength of received signals in indoor environment when tracking targets. They proposed a 'sensorless sensing' concept in which any wireless networks emitting wireless signals can act as sensor networks for motion detection. They used Mica2 and MicaZ hardware platforms to validate their proposal through indoor experiments and presented a velocity detection application. However, they only showed a simple experiment in which a toy car carrying a Mica2 mote travels through a hallway. It is an interesting idea to establish a relationship between received signal strength and the motion of a target without using sensor nodes. However, signature patterns of the received signal strength caused by different motions such as walk or run, and different obstructions such as walls and trees should be further exploited.

2.2.3 Device-Free Passive Tracking Systems

In practice, many applications require tracking without active participation from the targets (e.g. carrying tags) and with high privacy protection. Device-Free passive tracking systems provide a promising solution to these challenges. It is not necessary for targets to carry any devices to communicate with these systems and the targets being monitored may not even be aware of the existence of the tracking system. Device-Free passive systems provide a higher privacy protection since the systems can often detect and track a target but cannot identify the target. Vision-based Device-free passive systems use cameras to capture images of the target. Fingerprinting systems track targets by comparing initial training data and real-time data. Signature-free systems can perform DFP target tracking without a training phase.

Vision-based systems

The vision-based systems use multiple cameras to localize and track objects by processing the video recorded by the cameras [48]. The cameras can record the clear motion and detailed appearance of targets which is important for identification applications such as distinguishing criminals. When perform tracking, the recorded images are compared to images in a pre-installed database. The database contains images captured during an

installation process. Based on the comparison, the vision-based device-free systems can track the targets. Many vision-based tracking systems have been developed [37]. Both [55] and [49] describe real-time tracking systems which can localize and track multiple targets over a long period of time.

There are three main disadvantages of vision-based systems. First, they can only perform in line-of-sight scenarios since the camera cannot 'see' through opaque structures like a wall. Second, they require good lighting conditions which make them useless in dim or dark environments. Finally, vision-based systems are not suitable in applications that require privacy protection since video records provide clear identification of individuals.

Fingerprinting systems

As mentioned in the previous subsection, the fingerprinting approaches [1, 59, 66, 76, 77] require a significant initial training phase, with data gathered when there are targets at known locations. It makes rapid deployment in, e.g., emergency response scenarios, much more challenging.

In [1], Ahmed et al. proposed a tracking scheme using a particle filter and fingerprinting method to track a single target. The experiment used MicaZ [15] nodes to sense acoustic signals in a 2 meters $\times$ 3.5 meters very small test environment. After an off-line training phase, the system tracked a electric toy car with a speed varying from 0.2 to 0.35 m/sec. Although the particle filter method improves the accuracy of the tracking estimation, the tracking system still has a mean error of about 0.2 meters in that small-scale area. Further research in a larger scale experimental area is needed to access whether this approach has any practical potential.

Moussa et al. describes a WLAN detection system that employs the time-windowed average of the means and variances of RSS data to detect the motion of multiple targets [59]. During the off-line training, two distributions of received signal strength measurements are established, one for a vacant network with no targets moving within the sensing area, and one with a target moving inside. They also proposed an algorithm based on maximum likelihood estimation to determine which state was most likely in on-line detecting phase. However, this system can only detect moving targets; it cannot track targets.

The work in [59] was further extended by Seifeldin and Youssef in [66], they presented Nuzzer: A large-scale tracking system for wireless environments. In off-line phase, Nuzzer

measures a received signal strength histogram for each link with targets standing in different locations inside the sensing area. During the on-line phase, new samples are compared with stored received signal strength radio map to estimate target locations. In the field experiment, Nuzzer was deployed in an office building with a floor of 1500 square meters. Three existing Cisco Aironet 1130AG access points (APs) [17] and two laptops were used. During the off-line phase, a person stood at 53 different locations and Nuzzer recorded samples during 60 seconds for each location. Over measurements of 32 independent tests in different locations, Nuzzer achieved a median distance error of 1.82 meters. However, Nuzzer can only track a single target and the estimation accuracy still needs to be improved.

Viani et al. described a RSS tracking system using Support Vector Machine (SVM) [76, 77]. Support vector machines are a set of methods that analyze data and recognize patterns, used for classification [71]. During the training phase, received signal strength data were collected to train a support vector machine. The area of interest is divided into several small squares. During the in on-line tracking phase, the support vector machine is used to choose a small square based on new received signal strength measurements.

In device-free scenario, the major drawback of fingerprinting tracking systems is the mandatory training phase before tracking. It is time-consuming for training the system and the stored data is useless once the topology of the sensor network changes. Another challenge relies on computational cost and hardware storage for multi-target tracking, since the amount of data needed for training increases exponentially with the number of people. These issues need to be solved before a real implementation of fingerprinting tracking systems.

Signature-free systems

A number of other device-free passive tracking algorithms have been developed based on models of received signal strength [42, 54, 73, 82–84, 89, 90]. These approaches eliminate the need for extensive training. By analyzing the changes in RSS values, these algorithms can track the targets as soon as they are deployed in a new environment. The Radio Tomographic Imaging (RTI) system proposed by Wilson and Patwari [84] makes use of image reconstruction methods to estimate a map of attenuation in the region of interest at sequential points in time. This work was extended in [83] to use the empirical RSS variance on each link, rather than the mean RSS, to determine the presence of the targets, and this

promising approach has demonstrated the ability to track people through walls. The recent thesis [82] by Wilson also describes a particle filter for RF tomographic tracking. The filter employs sequential importance resampling, and model parameters such as noise variance, attenuator parameters, and sensor locations are assumed known.

Tseng et al. proposed a protocol to perform signature-free DFP tracking by using a 'mobile agent' when a target is detected [73]. A mobile agent can follow the path of the target by hopping between sensor nodes, cooperating with nodes nearby the target to choose the node closest to the target based on the RSS values. Thus an approximate trajectory can be extracted from the path of the nodes traversed. However, the work does not discuss the accuracy of target detection in practical experiment and it only reports simulation results. This may cause problem when a target can not be properly detected. Moreover, the accuracy of the estimation trajectory greatly depends on the topology of the network and it requires an intensive sensor deployment to achieve a good tracking, which is not practical.

Zhang et al. [89] use Mica2 sensors placed on the ceiling to localize a single target moving below. They propose a dynamic model to capture the RSS changes; the variation of the RF signals is used to determine the presence of the target. The experimental testbed comprised a 44 RF-based grid array covering 36 square meters. Recently, in further research [90], they proposed a new algorithm based on 'dynamic clustering' to address multi-target tracking. The 'dynamic clustering' is a scheme that can automatically detects and groups sensor nodes affected by targets. Each group is a cluster, the size and the number of nodes in a cluster are dynamically changing depending on targets' locations. The algorithm greatly improves the estimation accuracy that the system achieves an accuracy of 0.2-0.85 meters in a 36 square-meter area. However, the approach needs a calibration stage to set model parameters and the coverage is limited (targets must be within 2 meters of a sensor node).

2.3 Sequential Monte Carlo Algorithms

The tracking problem can generally be considered a dynamic state estimation problem, where the state represents the parameters of the target being tracked, such as its position and velocity. The goal is to estimate the next state of the target over time based on its previous states and the observation information collected by the sensors at each time step. From a Bayesian point of view, the tracking problem is to calculate a posterior probability

density function (pdf) of the target's state and to derive an optimal solution of the state based on the pdf, which contains all available information about the state [10].

For many problems, the state of the target can be accurately estimated by adopting a linear model with Gaussian noise. The observation information can also be modeled as a linear function of the state with additive Gaussian noise. For these linear/Gaussian Bayesian tracking problems, the posterior distribution is a Gaussian distribution and it can be parameterized by a mean and covariance. Both the Kalman Filter (KF) [41] and grid-based filters [10] provide an analytical solution. However, in many other situations, these assumptions do not hold, and approximations are needed for these nonlinear/non-Gaussian Bayesian tracking problems. The extended Kalman filter [81] and unscented Kalman filter [40] can approximate the optimal Bayesian solution.

The above methods fail when the problem is highly nonlinear and/or non-Gaussian. Sequential Monte Carlo methods, also known as particle filters, are able to perform better in these situations. Sequential Monte Carlo methods are flexible simulation-based approaches used for estimating the state in a Bayesian model in a sequential manner. There exists a considerable number of particle filter algorithms. The basic sequential Monte Carlo methods based on Sequential Importance Sampling (SIS), were proposed in the 1950s [56], and became more practical after a resampling stage was included in the algorithm in 1993 by Gordon et al. [30]. Based on this new Sampling Importance Resampling (SIR) procedure, more improvements were introduced and new sequential Monte Carlo algorithms were developed [12, 21, 57, 61, 75]. Among the different particle filtering algorithms, the Auxiliary Particle Filter (APF) [38,61] performs better in environment with small process noise. The auxiliary particle filter builds on sequential sampling importance resampling procedure, it attempts to improve the performance by modifying the sampling step and minimizing the variance. The details of these particle filtering methods are discussed in the following section. We begin with the construction of state-space model.

2.3.1 Overview

In a general state-space model, we first consider a discrete-time Markov process $\{\mathbf{x}_k; k \in \mathbb{N}\}$ where $\mathbf{x}_k \in \mathbb{R}^x$. $\mathbf{x}_k$ represents a state sequence of the system at time k, and the x in $\mathbb{R}^x$ denotes the dimension of the state vector (we use similar notation for dimension in this thesis). The transition probability is $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ and the initial distribution of $\mathbf{x}_k$ is

$p(\mathbf{x}_0)$. This process is unobserved(hidden), and we can only access another relative process $\{\mathbf{y}_k; k \in \mathbb{N}\}$ where $\mathbf{y}_k \in \mathbb{R}^y$. Given the states, observations are conditionally independent with measurement distribution $p(\mathbf{y}_k|\mathbf{x}_k)$. Therefore, a transition model and a measurement model can be constructed. Both models may be nonlinear/non-Gaussian. The models can be expressed as

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{v}_{k-1}) \tag{2.1}$$

$$\mathbf{y}_k = h(\mathbf{x}_k, \mathbf{s}_k), \tag{2.2}$$

where $\mathbf{v}_{k-1} \in \mathbb{R}^v$ is the process noise and $\mathbf{s}_k \in \mathbb{R}^s$ is the measurement noise. The mapping functions $f : \mathbb{R}^x \times \mathbb{R}^v \to \mathbb{R}^x$ and $h : \mathbb{R}^x \times \mathbb{R}^s \to \mathbb{R}^y$ represent the transition and measurement models.

The posterior density $p(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})$, where $\mathbf{x}_{0:k} \triangleq \{\mathbf{x}_0, \ldots, \mathbf{x}_k\}$ and $\mathbf{y}_{1:k} \triangleq \{\mathbf{y}_1, \ldots, \mathbf{y}_k\}$, constitutes the complete solution to the sequential estimation problem. Thus, the goal is to estimate the posterior density or its marginal distribution $p(\mathbf{x}_k|\mathbf{y}_{1:k})$.

In Monte Carlo simulation, a set of particle points $\mathbf{x}^i_{0:k}, i = 1, \ldots, N$, where N is the total number of particles, are drawn from the posterior distribution mentioned above to map integrals to discrete sums. For some function of interest $G_k(.)$, any expectation of the form

$$I(G_k) = \int G_k(\mathbf{x}_{0:k}) p(\mathbf{x}_{0:k}|\mathbf{y}_{1:k}) \, \mathrm{d}\mathbf{x}_{0:k}, \tag{2.3}$$

may be approximated by the following estimate

$$\hat{I}_N(G_k) = \frac{1}{N} \sum_{i=1}^{N} G_k(\mathbf{x}^{(i)}_{0:k}). \tag{2.4}$$

According to the strong law of large numbers, $\hat{I}_N(G_k) \xrightarrow[N\to\infty]{a.s} I(G_k)$ [29], where $\xrightarrow[N\to\infty]{a.s}$ denotes almost sure convergence.

Bayesian Importance Sampling

However, it is always difficult to sample directly from the posterior density function itself. An alternative way to avoid this problem is to sample from a known proposal distribution (also known as importance distribution) $\pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})$. According to the Bayesian Rule, we

have

$$p(\mathbf{x}_{0:k}|\mathbf{y}_{1:k}) = \frac{p(\mathbf{y}_{1:k}|\mathbf{x}_{0:k})p(\mathbf{x}_{0:k})}{p(\mathbf{y}_{1:k})}. \tag{2.5}$$

Substitute it into equation (2.3) we can obtain

$$I(G_k) = \int G_k(\mathbf{x}_{0:k}) \frac{\omega_k(\mathbf{x}_{0:k})}{p(\mathbf{y}_{1:k})} \pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k}) \, \mathrm{d}\mathbf{x}_{0:k}, \tag{2.6}$$

where ω_k is the unnormalized importance weight

$$\omega_k = \frac{p(\mathbf{y}_{1:k}|\mathbf{x}_{0:k})p(\mathbf{x}_{0:k})}{\pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})}. \tag{2.7}$$

Then we can obtain

$$I(G_k) = \frac{\int G_k(\mathbf{x}_{0:k})\omega_k(\mathbf{x}_{0:k})\pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k}) \, \mathrm{d}\mathbf{x}_{0:k}}{\int p(\mathbf{y}_{1:k}|\mathbf{x}_{0:k})p(\mathbf{x}_{0:k}) \frac{\pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})}{\pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})} \, \mathrm{d}\mathbf{x}_{0:k}} \tag{2.8}$$

$$= \frac{\int G_k(\mathbf{x}_{0:k})\omega_k(\mathbf{x}_{0:k})\pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k}) \, \mathrm{d}\mathbf{x}_{0:k}}{\int \omega_k(\mathbf{x}_{0:k})\pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k}) \, \mathrm{d}\mathbf{x}_{0:k}}. \tag{2.9}$$

Therefore, the particle samples can be drawn from the importance distribution, and we can then approximate the expectation of interest by

$$\hat{I}_N(G_k) = \frac{\frac{1}{N}\sum_{i=1}^{N} G_k(\mathbf{x}_{0:k}^{(i)})\omega_k^{(i)}}{\frac{1}{N}\sum_{i=1}^{N} \omega_k^{(i)}} = \sum_{i=1}^{N} G_k(\mathbf{x}_{0:k}^{(i)})\widetilde{\omega}_k^{(i)}, \tag{2.10}$$

where the normalized importance weights $\widetilde{\omega}_k^{(i)}$ are given by

$$\widetilde{\omega}_k^{(i)} = \frac{\omega_k^{(i)}}{\sum_{i=1}^{N} \omega_k^{(i)}}. \tag{2.11}$$

2.3.2 Sequential Importance Sampling (SIS)

In the sequential case of the previous method, we can adopt the following importance distribution

$$\pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k}) = \pi(\mathbf{x}_{0:k-1}|\mathbf{y}_{1:k-1})\pi(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{y}_{1:k}), \tag{2.12}$$

to recursively compute the sequential estimate of the posterior distribution. Given that the states are Markovian and the observations are conditionally independent, we can obtain

$$p(\mathbf{x}_{0:k}) = p(\mathbf{x}_0)\prod_{j=1}^{k} p(\mathbf{x}_j|\mathbf{x}_{j-1}) \tag{2.13}$$

$$p(\mathbf{y}_{1:k}|\mathbf{x}_{0:k}) = \prod_{j=1}^{k} p(\mathbf{y}_j|\mathbf{x}_j). \tag{2.14}$$

By substituting (2.12), (2.13) and (2.14) into (2.7), we can get the importance weight

$$\omega_k \propto \frac{p(\mathbf{y}_{1:k}|\mathbf{x}_{0:k})p(\mathbf{x}_{0:k})}{\pi(\mathbf{x}_{0:k-1}|\mathbf{y}_{1:k-1})\pi(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})} \tag{2.15}$$

$$\propto \omega_{k-1}\frac{p(\mathbf{y}_{1:k}|\mathbf{x}_{0:k})p(\mathbf{x}_{0:k})}{p(\mathbf{y}_{1:k-1}|\mathbf{x}_{0:k-1})p(\mathbf{x}_{0:k-1})}\frac{1}{\pi(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})} \tag{2.16}$$

$$\propto \omega_{k-1}\frac{p(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{x}_{k-1})}{\pi(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})}. \tag{2.17}$$

This relationship allows us to sequentially update the weights given the importance distribution, $\pi(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})$. It is important to design the appropriate form of this importance distribution to better assist sampling.

Choosing the Importance Distribution

In [21], Doucet discussed the choice of importance distribution. The principle is to choose a distribution that minimizes the variance of the important weights. As explained in [88],

the optimal choice is

$$\begin{aligned}\pi(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})_{optimal} &= p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{y}_k) \\ &= \frac{p(\mathbf{y}_k|\mathbf{x}_k, \mathbf{x}_{k-1})p(\mathbf{x}_k|\mathbf{x}_{k-1})}{p(\mathbf{y}_k|\mathbf{x}_{k-1})}. \end{aligned} \tag{2.18}$$

Substituting (2.18) into (2.17), we obtain:

$$\omega_k \propto \omega_{k-1} p(\mathbf{y}_k|\mathbf{x}_{k-1}). \tag{2.19}$$

However, it is often difficult to generate the analytical form of the solution when sampling from $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{y}_k)$. If $\mathbf{x}_k$ is a member of a finite set, the integral can be calculated as a sum of finite components [23]. Another case where an analytical solution can be obtained is under the assumption of Gaussian state space model with non-linear transition model and linear measurement model [21]. In practice, a common method is to choose the importance distribution as the prior distribution, which results in a simple sampling procedure and weight updating procedure.

$$\pi(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{y}_{1:k}) = p(\mathbf{x}_k|\mathbf{x}_{k-1}) \tag{2.20}$$

$$\omega_k \propto \omega_{k-1} p(\mathbf{y}_k|\mathbf{x}_k). \tag{2.21}$$

Although it results in important weights with higher variance than the optimal solution (since it does not incorporate the most recent observations), it is usually easier for implementation [21, 56].

The Degeneracy Problem of SIS

The drawback of the SIS algorithm is the increasing variance of the importance weights over time. To show this we can expand the importance weight:

$$\omega_k = \frac{p(\mathbf{y}_{1:k}|\mathbf{x}_{0:k})p(\mathbf{x}_{0:k})}{\pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})} \tag{2.22}$$

$$= \frac{p(\mathbf{y}_{1:k}, \mathbf{x}_{0:k})}{\pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})} \tag{2.23}$$

$$\propto \frac{p(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})}{\pi(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})}. \tag{2.24}$$

(2.24) is the importance ratio, and it is shown in [22] that the variance increases over time.

The degeneracy problem is a result of this increasing variance. After multiple time steps running the algorithm, the normalized weight of one particle may tend to 1, while the normalized weights of the rest of particles tend to 0. This makes the algorithm discard most of the particles, therefore, the degeneracy of particles will eventually cause a failure of the particle filter estimation.

2.3.3 Sequential Importance-sampling Resampling (SIR)

Algorithm 1: Residual Resampling

1 Recalculate the weights

$$\bar{\omega}^i = \frac{N\omega^i - \lfloor N\omega^i \rfloor}{N - R}, i = 1, \ldots, N \tag{2.25}$$

where N is the number of particles, $\lfloor \quad \rfloor$ denote the integer part, and $R = \sum_{i=1}^{N} \lfloor N\omega^i \rfloor$.

2 Calculate the deterministic component

$$N_d^i = \lfloor N\bar{\omega}^i \rfloor. \tag{2.26}$$

3 Calculate the multinomial component: $\{N_m^i\}$ are distributed according to the multinomial distribution $MULT(N - R; \bar{\omega}^1, \ldots, \bar{\omega}^n)$.

4 $N^i = N_d^i + N_m^i$, N^i is the number of copies of particle $\mathbf{x}_k^{(i)}$.

In order to address this degeneracy problem, a resampling method [20] is adopted with a basic aim of discarding the particles with lower weights and replacing them by particles with higher weights. Douc described four popular resampling methods [20]: multinomial resampling [30], residual resampling [57], stratified resampling [46] and systematic resampling [46]. Since the choice of resampling scheme does not greatly affect the performance of particle filter [20], here we only present the residual resampling scheme in Algorithm 1. This scheme is shown in [20] to achieve weights with a lower conditional variance. To better describe the algorithm, a SIR particle filter is described in Algorithm 2.

Algorithm 2: SIR Particle Filter

`// Initialization at time` $k = 0$

Sample the particles $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0), (i = 1, \ldots, N)$;

for $k = 1, \ldots, T$ **do**

- Sample $\widetilde{\mathbf{x}}_k^{(i)} \sim p(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)}), (i = 1, \ldots, N)$;
- Set weights: $\omega_k^{(i)} \propto p(\mathbf{y}_k|\widetilde{\mathbf{x}}_k^{(i)})$;
- Normalize weights: $\widetilde{\omega}_k^{(i)} = \frac{\omega_k^{(i)}}{\sum_{i=1}^{N} \omega_k^{(i)}}$;

`// Resample`

- Resample N particles $(\mathbf{x}_k^{(i)}; i = 1, \ldots, N)$ from the set $(\widetilde{\mathbf{x}}_k^{(i)}; i = 1, \ldots, N)$ according to the importance weights $\widetilde{\omega}_k^{(i)}$;

end for

2.3.4 Auxiliary Particle Filter

A major problem of the SIR particle filter is the fact that if the samples $\mathbf{x}_k$ at time step k vary significantly from the previous samples $\mathbf{x}_{k-1}$, the variance of the weights will be too high, which makes the SIR algorithm ineffective. The auxiliary particle filter, introduced in [61], modifies the sampling step in an attempt to improve performance. The auxiliary particle filter is a method that tries to make use of future information for prediction. At time k, it will predict which particles will be located in regions of high probability masses at time $k + 1$. The filter calculates a *first-stage weight* $\rho_k^{(i)}$ for each particle based on how well the particle can explain the observations $\mathbf{y}_k$. Ideally, this weight should be a good approximation to the likelihood

$$\widehat{p}(\mathbf{y}_k|\mathbf{x}_{k-1}^{(i)}) = \int p(\mathbf{y}_k|\mathbf{x}_k) p(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)})\, d\mathbf{x}_k, \tag{2.27}$$

i.e. $\rho_k^{(i)} \approx p(\mathbf{y}_k|\mathbf{x}_{k-1}^{(i)})$.

Algorithm 3: Auxiliary Particle Filter

// Initialization at time $k = 0$

for $i = 1, \ldots, N$ **do**

  Sample the particles $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)$;

  Set weights $\omega_0^{(i)} = \frac{p(\mathbf{y}_1|\mathbf{x}_0^{(i)})p(\mathbf{x}_0^{(i)})}{\pi(\mathbf{x}_0^{(i)})}$;

end for

Normalize weights $\widetilde{\omega}_0^{(i)} = \frac{\omega_0^{(i)}}{\sum_{i=1}^N \omega_0^{(i)}}$;

// For times $k > 0$

for $k = 1, \ldots, T$ **do**

  // First-stage weights

  for $i = 1, \ldots, N$ **do**

    Sample $\rho_k^{(i)} \sim p(\cdot)$;

    Set weights $W_k^{(i)} = \widetilde{\omega}_{k-1}^{(i)} \times \rho_k^{(i)}$;

  end for

  Normalize weights $\widetilde{W}_k^{(i)} = \frac{W_k^{(i)}}{\sum_{i=1}^N W_k^{(i)}}$;

  // Resample

  Resample from the set $\left\{\mathbf{x}_{k-1}^{(i)}, \widetilde{W}_k^{(i)}\right\}_{i=1}^N$ to obtain $\left\{\mathbf{x}_{k-1}^{\prime(i)}, \frac{1}{N}\right\}_{i=1}^N$;

  for $i = 1, \ldots, N$ **do**

    Set $\mathbf{x}_{1:k-1}^{(i)} = \mathbf{x}_{1:k-1}^{\prime(i)}$ and $\rho_k^{(i)} = \rho_k^{\prime(i)}$;

    Sample $\mathbf{x}_k^{(i)} \sim \pi(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)})$;

    Set weights $\omega_k^{(i)} = \frac{p(\mathbf{z}_k|\mathbf{x}_k^{(i)})p(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)})}{\rho_k^{(i)}\pi(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)})}$;

  end for

  Normalize weights $\widetilde{\omega}_k^{(i)} = \frac{\omega_k^{(i)}}{\sum_{i=1}^N \omega_k^{(i)}}$;

  // Optional second resample

  Resample $\left\{\mathbf{x}_{1:k}^{(i)}, \widetilde{\omega}_k^{(i)}\right\}_{i=1}^N$ to obtain $\left\{\mathbf{x}_{1:k}^{(i)}, \frac{1}{N}\right\}_{i=1}^N$;

end for

The auxiliary particle filter then resamples the particles $\mathbf{x}_{k-1}^{(i)}$ according to the first-stage weights, this allows the auxiliary particle filter to be adapted in a more efficient way. Here an auxiliary variable $\prime(i)$ is introduced to aid during the resampling step. After

the resampling step, the particles are propagated according to an importance function $\pi(\mathbf{x}_k|\mathbf{x}_{k-1}^{\prime(i)})$, and here we set it as $p(\mathbf{x}_k|\mathbf{x}_{k-1}^{\prime(i)})$ for simplicity consideration, and the new weights ω can then be calculated. The original auxiliary particle filter in [61] includes two stage resampling. The auxiliary particle filter algorithm is improved in [12] to perform resampling once at each time step. This version of the auxiliary particle filter algorithm reduces the variance of the particle estimations and is widely used.

The algorithm is described in Algorithm 3. Although it is not an ideal choice because it can lead to unbounded variance in the estimates, it is not so sensitive to outliers as the standard SIR filter [24]. In the algorithm, we use the following first-stage weights:

$$\rho_k^{(i)} = p(\mathbf{y}_k|\mu_k^{(i)}), \tag{2.28}$$

where $\mu_k^{(i)}$ is some characterization of $\mathbf{x}_k$ given $\mathbf{x}_{k-1}^i$. In our algorithm, $\mu_k^{(i)}$ is the mean of $p(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)})$. This is one of the suggested approaches in [61], and it achieves good performance in practice for many tracking tasks.

2.4 Static Parameter Estimation

2.4.1 Overview

Sequential Monte Carlo approaches can only operate when all parameters in the general state-space model are known. In this case, it is possible to use sequential Monte Carlo methods to obtain solutions. These problems with known parameters are called optimal filtering problem. However, in many practical applications, some critical parameters such as process noise $\mathbf{v}_k$ in transition model and measurement noise $\mathbf{s}_k$ in measurement model at time step k are unknown in prior. Therefore, we can group all these unknown parameters as a set θ, and it is necessary for us to estimate θ before perform particle filtering [8].

Static parameter estimation technique tries to estimate the set θ. In the past few years, many methods have been proposed. They can be classified into two classes [43]: Bayesian approaches and Maximum Likelihood approaches. Both can be implemented in an on-line or an off-line manner.

In the Bayesian realm, the parameter set θ is first assigned a proper prior distribution. Then, using the observed data and a suitable prior density $p(\theta)$ for θ, the joint posterior distribution $p(\mathbf{x}_{0:k}, \theta|\mathbf{y}_{0:k})$ is characterized. Markov Chain Monte Carlo (MCMC) [6, 70] is

considered as a standard approach to approximate this joint posterior distribution. However, in a general non-linear/non-Gaussian state space model, it is difficult to obtain the distribution using a Markov chain Monte Carlo approach. A particle Markov Chain Monte Carlo technique [7] is proposed to solve this problem by Andrieu et al.. It is an off-line Bayesian method which combines Markov chain Monte Carlo and standard sequential Monte Carlo method. The sequential Monte Carlo method is used to design efficient importance distributions for Markov chain Monte Carlo algorithms. Unlike standard Markov chain Monte Carlo, the particle MCMC method is not required to design complex proposals for state $\mathbf{x}_{0:k}$, it only needs to design a distribution for the parameter set θ. This simpler distribution is used to obtain a sequential Monte Carlo approximation of the unknown set θ. More precisely, to obtain a proposal distribution that is an approximation of the ideal joint posterior distribution $p(\mathbf{x}_{0:k}, \theta|\mathbf{y}_{0:k})$. This method achieves good performance with minimal tuning even when the transition density is used as the importance distribution in the sequential Monte Carlo algorithm. However, its computational cost is high, it is $O(NT)$ per MCMC step, where N is the number of particles in the sequential Monte Carlo algorithm. Another 'artificial dynamics method' [47] adds random noise to the parameters, thus making these parameters part of the state vector. However, it is difficult to tune the newly introduced dynamics. There is a risk of changing the original problem setting. The Resample-Move method [6, 43, 70] successfully avoids modification of the distribution, but it also suffers from the degeneracy problem.

On the other hand, in Maximum Likelihood technique, the estimate of θ is the maximizing argument of the marginal likelihood of the observation. Consider the observation vector $\mathbf{y}_{1:T}$ where T is the total number of time steps over which filtering is performed. Then the marginal likelihood $l(\theta)$ of the observation is:

$$l(\theta) = \log p_\theta(\mathbf{y}_{1:T}), \tag{2.29}$$

and the estimate of θ is:

$$\hat{\theta} = \arg\max_{\theta \in \Theta} l(\theta). \tag{2.30}$$

A commonly used method of Maximum Likelihood parameter estimation is gradient approach [62]. Its drawback is the difficulty of tuning the step size. Another method is

expectation maximization approach [19], which is more stable and typically more computationally efficient for a high dimensional parameter set θ.

The EM algorithm can be divided into two steps to maximize $l(\theta)$. First is the expectation step or E-step, which evaluates the posterior probabilities using the current estimated values for the parameters. It computes:

$$Q(\hat{\theta}_k, \hat{\theta}) = \int \log p_{\hat{\theta}}(\mathbf{x}_{0:T}, \mathbf{y}_{0:T}) p_{\hat{\theta}_k}(\mathbf{x}_{0:T}, \mathbf{y}_{0:T}) dx_{0:T}, \tag{2.31}$$

where $\hat{\theta}_k$ is the updated estimate set at time step k, and $Q(\cdot)$ function is used to denote the integral function at the right side of the equation. The second step is the maximization step or M-step, which uses the probabilities computed in the E-step to re-estimate the parameter set $\hat{\theta}_k$:

$$\hat{\theta}_{k+1} = \arg\max_{\theta} Q(\hat{\theta}_k, \hat{\theta}), \tag{2.32}$$

so that the results calculated by the EM method can satisfy: $l(\hat{\theta}_{k+1}) \geq l(\hat{\theta}_k)$. The EM approach outperforms the gradient approach when the M-step can be computed analytically.

2.4.2 On-line EM using Pseudo-Likelihood Function

When the amount of observation data to be processed is large, or the application has real-time constraints, it will be difficult for the expectation maximization method to process the batch data. An alternative is to use recursive processing so that the observation data is processed sequentially. In other words, if we have an estimate for the first k time steps, we only need to update the new estimate $\hat{\theta}_{k+1}$ after receiving the new data $\mathbf{y}_{k+1}$. Based on the EM approach, this is the so-called on-line EM algorithm.

On-line EM algorithm follows the same two step procedure as in the EM approach. It has a computational cost of $O(N^2)$ per parameter update [43]. In addition, it requires estimations of 'sufficient statistics' from joint posterior distribution $p(\mathbf{x}_{0:k}, \theta|\mathbf{y}_{0:k})$. A sufficient statistic describes a statistic that provides sufficient information to a statistical model and its associated unknown parameters. In other words, no other statistic can offer any additional information than a sufficient statistic based on the same samples. Similar to sequential Monte Carlo technique, such an approach suffers from the problem that the performance of estimation degrades as the number of time step T increases.

To overcome this problem, an on-line EM approach incorporates a 'pseudo-likelihood function' [8]. The pseudo-likelihood function [65] is an approximation to the likelihood function based on observations. It can either simplify the problem for estimation or provide explicit estimates to unknown model parameters. To formulate the pseudo-likelihood function, we first divide the whole observation data set $\mathbf{y}_{0:T}$ into blocks. Each block contains L measurements (or L time steps in this thesis). The block of both states and observations can be define as

$$X_b \triangleq \mathbf{x}_{bL+1:(b+1)L} \tag{2.33}$$

$$Y_b \triangleq \mathbf{y}_{bL+1:(b+1)L}. \tag{2.34}$$

where b is the index of the block.
The vectors $\{X_b, Y_b\}$ are identically distributed. Thus, we can obtain their common distribution $p_{\hat{\theta}}(\mathbf{x}, \mathbf{y})$ as follows

$$p_{\hat{\theta}}(\mathbf{x}, \mathbf{y}) = \pi_{\hat{\theta}}(\mathbf{x}_{bL+1}) h_{\hat{\theta}}(\mathbf{y}_{bL+1}|\mathbf{x}_{bL+1}) \prod_{k=bL+2}^{(b+1)L} f_{\hat{\theta}}(\mathbf{x}_k|\mathbf{x}_{k-1}) h_{\hat{\theta}}(\mathbf{y}_k|\mathbf{x}_k), \tag{2.35}$$

where $\pi_{\hat{\theta}}(.)$ is the importance distribution function, $h_{\hat{\theta}}(.)$ is the function of measurement model and $f_{\hat{\theta}}(.)$ is the function of the transition model.
The likelihood of the observation $p_{\hat{\theta}}(Y_b)$ at block b is

$$p_{\hat{\theta}}(Y_b) = \int_{\mathbf{X}^L} p_{\hat{\theta}}(\overline{\mathbf{x}}_b, Y_b) \mathbf{d}\overline{\mathbf{x}}_b, \tag{2.36}$$

where $\mathbf{X}$ is the space that process $\{\mathbf{x}_k\}$ defined on, and $\mathbf{X} \subseteq \mathbb{R}^x$. If process $\{\mathbf{x}_k\}$ is stationary and ergodic, then it is shown in [8] that the average log pseudo-likelihood $\overline{l}(\hat{\theta})$ satisfies

$$\overline{l}(\hat{\theta}) = \int_{Y^L} \log p_{\hat{\theta}}(\mathbf{y}) p_{\theta}(\mathbf{y}) \mathbf{d}\mathbf{y}. \tag{2.37}$$

We therefore apply on-line EM to recursively maximize $\overline{l}(\hat{\theta})$ by updating the estimate $\hat{\theta}$ via

$$\hat{\theta}_b = \arg\max_{\hat{\theta} \in \Theta} Q(\hat{\theta}_b, \hat{\theta}), \tag{2.38}$$

where

$$Q(\hat{\theta}_b, \hat{\theta}) = \int_{X^L \times Y^L} \log(p_{\hat{\theta}}(\mathbf{x}, \mathbf{y})) p_{\hat{\theta}_b}(\mathbf{x}|\mathbf{y}) p_\theta(\mathbf{y}) \mathbf{dxdy} \tag{2.39}$$

In a general state space model, it is impossible to directly compute $Q(\hat{\theta}_b, \hat{\theta})$. This motivate us to compute $Q(\cdot)$ via a set of sufficient statistics $\mathbf{\Omega}(\hat{\theta}_b, \hat{\theta})$. We use $\hat{\theta}_b = \Lambda(\Omega(\hat{\theta}_{b-1}, \hat{\theta}))$ as a mapping function from the sufficient statistics $\Omega(\hat{\theta}, \hat{\theta})$ to $\hat{\theta}$ that maximizes $Q(\cdot)$.

The on-line pseudo-likelihood EM method has a computational cost of $O(LN)$ per on-line EM step, which is better then using the standard on-line EM approach. In addition, it avoids the degeneracy problem when the block of time steps L is not large [50], such as ranges from 10 to 20.

Chapter 3

On-line Sequential Monte Carlo Tracking in Wireless Sensor Networks

In this chapter, we formulate the RF tomographic tracking problem. Also, we propose a novel measurement model to establish the relationship between RSS values collected from the sensor data and the state vector of the tracked target. The aim is to avoid quantizing the region of interest into pixels when modeling the effects of attenuating objects on a given link. We also present a previously proposed pixelized measurement model [83, 84] as a comparison. The proposed pixel-free model, when incorporated in our tracking algorithm, improves the algorithm computational efficiency and tracking accuracy.

We also propose a novel RF tomographic tracking algorithm by incorporating a auxiliary particle filtering approach with an on-line expectation maximization procedure. The particle filter tracking method improves the accuracy of the estimation results and the on-line EM algorithm is used to estimate the unknown parameters of the transition model and measurement model. These parameters vary significantly for different targets and environments and remain unknown to our tracking system. For example, the noise level of an outdoor grassy field is different from that of an indoor office. In this manner, while we are tracking the target position, we simultaneously estimate unknown parameters such as the noise variance and parameters in the target attenuation model.

3.1 Problem Statement

Received signal strength measurements on Radio-Frequency signal links connecting nodes in a sensor network reflect information about both (1) the pair-wise distances between sensor nodes and (2) objects moving through the sensed region. In particular, obstructions inside the area can absorb, scatter or reflect part of the signals. As the target moves, different links will be affected, revealing information about the location of the target within the region. The changes in mean attenuation of the signals transmitted between multiple pairs of sensor nodes can be used to estimate the position of the moving target. Moreover, the precise nature of the received signal strength measurements depends on a number of model parameters which are generally not known a priori. Our goal is to use measurements of received signal strength on the links between many pairs of nodes and over multiple time steps to jointly track a target moving through the region of interest and estimate model parameters such as the noise variance.

We consider a wireless sensor network of K nodes and $M = \frac{K^2-K}{2}$ bidirectional links. In each measurement interval, the K nodes successively broadcast packets and all neighboring nodes measure the received signal strength. The received signal strength value of bidirectional link j at time step k is denoted $\gamma_j(k)$. We take $\gamma_j(k)$ to be the average of the received signal strength values along both the forward $\gamma_j^F(k)$ and reverse $\gamma_{(j)}^R(k)$ links

$$\gamma_j(k) = \frac{1}{2}\big(\gamma_j^F(k) + \gamma_j^R(k)\big). \tag{3.1}$$

The precise measurement model for $\gamma_j(k)$ is described in section 3.2 below. The average two-way received signal strength values of a particular link $\gamma_j(k)$ depends on

$$\gamma_j(k) = P_0 + P_d + y_k^{(j)} + \zeta_k^{(j)}, \tag{3.2}$$

where P_0 is the transmitted power in dB on the link, P_d represent the path loss in dB within a large-scale area due to the distance between two nodes, $y_k^{(j)}$ is the shadowing loss in dB due to the moving target and $\zeta_k^{(j)}$ is the summation of the fading loss caused by the interference of narrow-band signals in multi-path environments and the static losses from distance, antenna behavior, etc, in dB.

We assume that there is a window where we can gather measurements on all links when

no target is present, so this average received signal strength $\bar{\gamma}_j$ on link j is

$$\bar{\gamma}_j = P_0 + P_d + \bar{\zeta}_{(j)}, \tag{3.3}$$

where $\bar{\zeta}_j$ is the average noise summation over time in dB. Nodes successively broadcast packets at relatively small time intervals (e.g., every 5 ms), gathering received signal strength measurements which we can stack into the vector Γ_k at time step k and Γ_{avg} over the time window when no target is present

$$\Gamma_k = [\gamma_1(k), \ldots, \gamma_j(k), \ldots, \gamma_M(k)]^T \tag{3.4}$$

$$\Gamma_{avg} = [\bar{\gamma}_1, \ldots, \bar{\gamma}_j, \ldots, \bar{\gamma}_M]^T. \tag{3.5}$$

This allow us to estimate $y_k^{(j)}$ from later measurements by subtracting off $\bar{\gamma}_j$ from $\gamma_j(k)$. We now can obtain the vector of received signal strength changes $\mathbf{y}_k$ caused by the moving target

$$\mathbf{y}_k + \mathbf{n}_k = \Gamma_k - \Gamma_{avg}, \tag{3.6}$$

where $\mathbf{n}_k$ is the noise vector after subtraction. Our goal is to track a single moving target described by state $\mathbf{x}_k$, with motion specified by a Markovian dynamic model $f(\mathbf{x}_k|\mathbf{x}_{k-1})$. In order to do this, we strive to maintain a particle approximation of the marginal posterior $p(\mathbf{x_k}|\mathbf{y}_{1:k})$ and estimate the expected value of $\mathbf{x}_k$ under this distribution. Simultaneously, we seek to estimate measurement model parameters θ.

3.2 Measurement Models

The measurement model describes the relationship between the true state, the sensor locations and the measurement values. Wilson et al. proposed a pixelized measurement models for RF tomography [83,84]. Since these were employed in an imaging framework, the models were pixelized, i.e. the area under surveillance was divided into fixed size pixels. Our goal is tracking, not imaging, so there is no need to introduce pixels. Such an introduction is undesirable because it necessarily leads to additional quantization error. We therefore develop a pixel-free model that is better suited to the sequential Monte Carlo methods we adopt, significantly enhancing the computational efficiency and leading to improved

tracking accuracy. Both models are introduced as follows.

For the bidirectional link j between a node pair, consider an ellipse with foci at the transmitter c and receiver e. Define

$$\lambda_k^{(j)} \triangleq d_j^c(\mathbf{x}_k) + d_j^e(\mathbf{x}_k) - d_j, \tag{3.7}$$

where $d_j^c(\mathbf{x}_k)$, $d_j^e(\mathbf{x}_k)$ are the distances from the target's position to the transmitter and receiver, respectively. The parameter $\lambda_k^{(j)}$ is equal to the major diameter of an ellipse passing through $\mathbf{x}_k$ with foci at the transmitter c and receiver e of the jth link, minus the length of jth link, d_j. Note that this quantity captures information about the position of the target relative to link j at time step k.

3.2.1 Pixelized Model

As shown in Figure 3.1, the pixelized model [84] divides the sensed area into several small square pixels with a pixel width Δp. We define total number of pixels as A and index of pixel as a. We then assign a weight $u_k^{(a)}$ to each pixel, the weight equals to either 0 or 1 depending on the location of the target at time step k. For example, assume that there is a target traveling within the sensor network. Some links are affected when the target stands right on the links' transmission path or within their ellipse sensing areas. Thus it causes changes on the received signal strength of those affected links. Pixels within the ellipses of these affected links are assigned a weight 1, otherwise assigned 0. Obviously, each pixel belongs to more than one ellipse, these ellipses represent links that are formed by different node pairs. The weight $u_k^{(a)}$ of a pixel a is assigned to 1 when one or more of its links are affected by the motion of the target. The weight, which depends on the presence of the target, is defined as

$$u_k^{(a)} = \begin{cases} 1, & \text{if motion of target affects pixel's related links.} \\ 0, & \text{otherwise.} \end{cases} \tag{3.8}$$

And a pixel weight vector $\mathbf{u_k}$ is defined as

$$\mathbf{u}_k \triangleq \left[\; u_k^1 \ldots u_k^{(a)} \ldots u_k^A \; \right]^T. \tag{3.9}$$

Since the location of the target at time k is described by the state vector $\mathbf{x}_k$, we can also define a 'convert function' $\eta(\cdot)$ for simulation such that $\mathbf{u}_k = \eta(\mathbf{x}_k)$. The function converts target status to a binary weight matrix that identifies the pixels affected by the target at time step k.

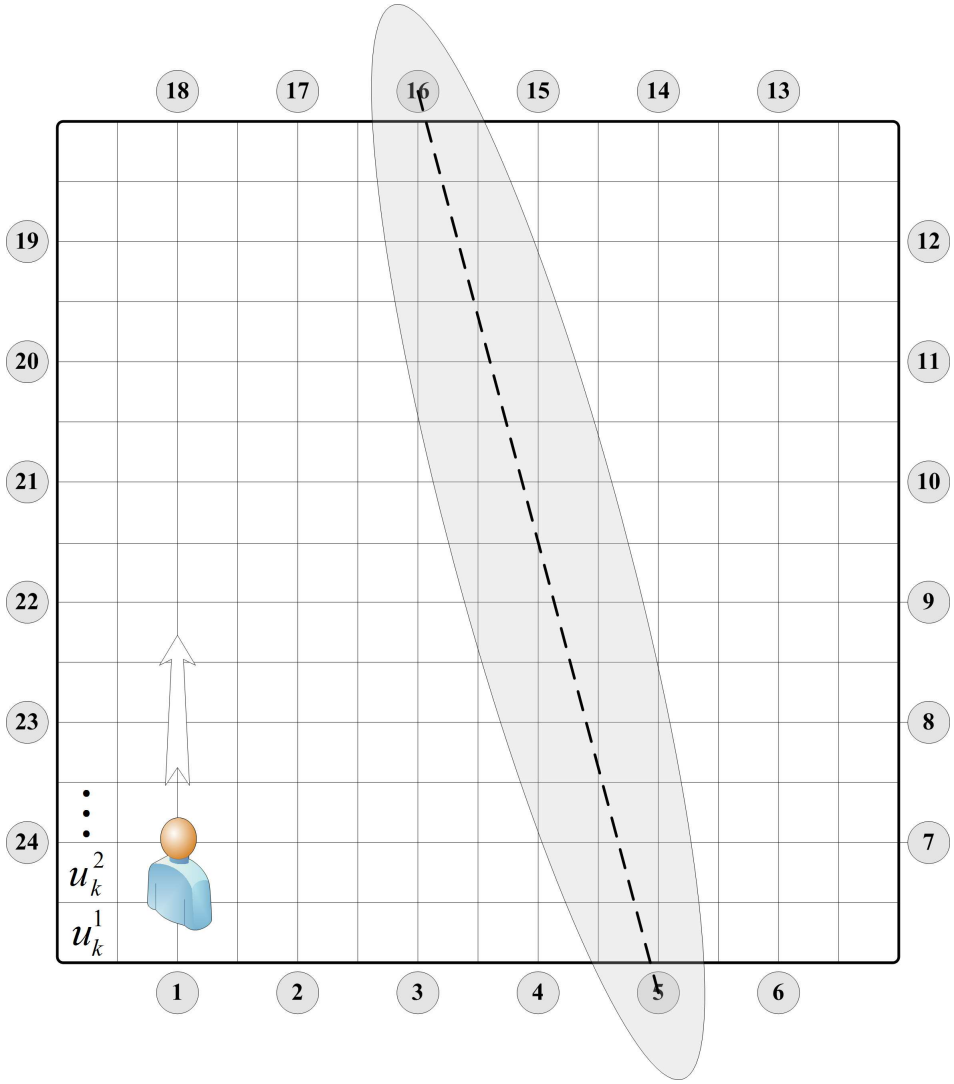


Fig. 3.1 An example for pixelized model: 24 nodes are deployed around a square layout. The area is pixelized into smaller pixel squares. Each node pair forms an ellipse sensing area, the RSS value of a link is affected if a target presents within the elliptical area.

The pixelized model for the attenuation caused by an object can be expressed as follows:

$$\mathbf{y}_k = \phi \boldsymbol{\Upsilon} \mathbf{u}_k + \mathbf{s}_k \tag{3.10}$$

$$= \phi \boldsymbol{\Upsilon} \mathbf{u}_k + \sigma_s \mathbf{S}_k, \tag{3.11}$$

here $\mathbf{y}_k$ is a $M \times 1$ vector represents measurement values. ϕ is the (modeled) value of mean attenuation at $\lambda = 0$ (i.e., when the target is directly obstructing the link), and $\mathbf{S}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{M\times 1})$ is additive white Gaussian noise. The parameter σ_s is the standard deviation that captures the variance of the noise, which is modeled as independent of λ. $\boldsymbol{\Upsilon}$ is an $M \times A$ matrix that describes how much a pixel is affected by the object. It is constructed using the elliptical model in equation (3.7). For a unique link j of a node pair and a square pixel a, the component Υ_{ja} of $\boldsymbol{\Upsilon}$ is

$$\Upsilon_{ja} = \frac{1}{\sqrt{d_j}} \begin{cases} 1, & \text{if } d^c_{ja} + d^e_{ja} < d_j + \lambda^{(j)} \\ 0, & \text{otherwise,} \end{cases} \tag{3.12}$$

where d_j is the distance between two nodes, d^c_{ja} and d^e_{ja} are the distances from the center of pixel a to the two node locations on link j. Similar to $\lambda^{(j)}_k$ in equation (3.7), $\lambda^{(j)}$ equals to the major diameter of an ellipse passing through pixel a with foci at nodes c and e of the jth link, minus d_j.

3.2.2 Pixel-free Model

The form of the proposed pixel-free model is motivated by experimental data recorded in a sensor network deployed in multiple outside environments with relatively few obstructions (some trees and a statue). The experiments involved a human walking around a region surrounded by sensor nodes (see Chapter 4 for more details of the sensor deployment).

For every measurement set, we calculated $\lambda^{(j)}$ for all links and measured the corresponding attenuation $y^{(j)}$ on the jth link (the measured received signal strength value minus the background mean, determined from a set of measurements conducted when the monitored area was empty). Figure **??** shows these attenuation values as a function of λ. Also shown are the mean attenuation (calculated over all points within bins of λ-range 0.01) and our proposed model, detailed below. The model involves three parameters (in the graph, these have been determined by using straightforward regression to minimize the mean-squared

error).

Figure 3.2 suggests that the mean attenuation level decays approximately following an exponential decreasing function with respect to the ellipse parameter λ, which details below.

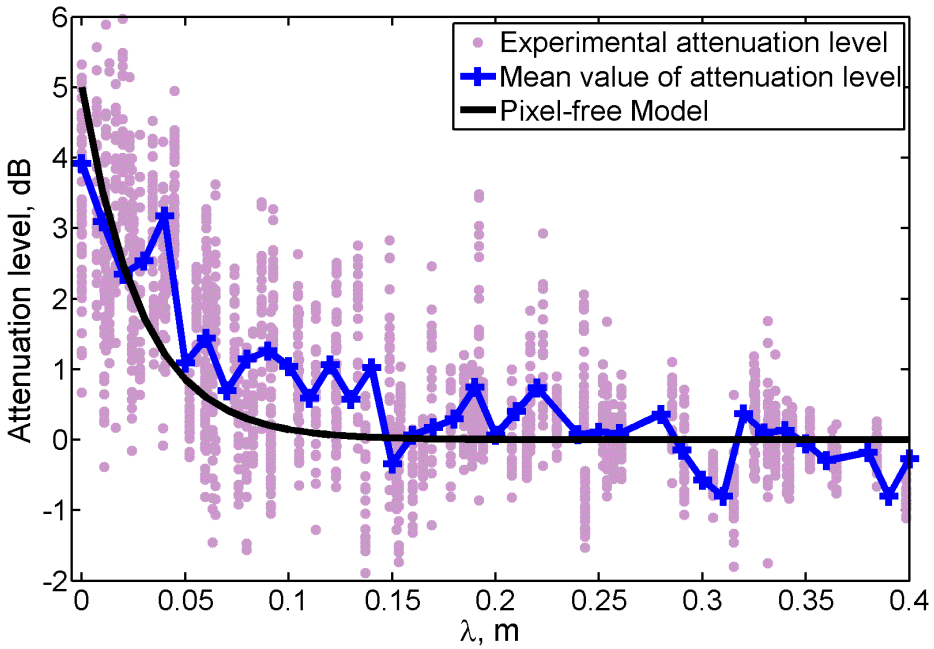


Fig. 3.2 Attenuation level versus λ for the proposed pixel-free model (a comparison between the model and experimental measurements.)

The pixel-free model for the attenuation caused by an object can then be described as follows:

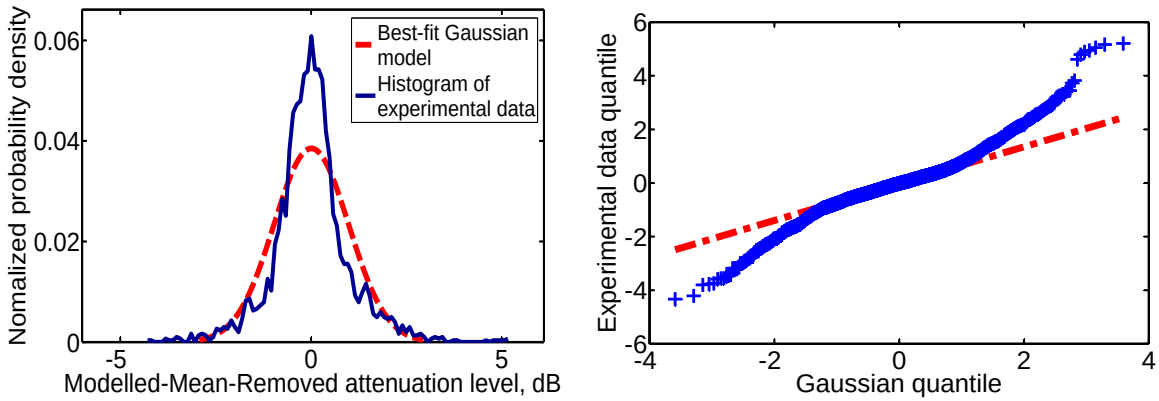
$$\mathbf{y}_k = \phi \mathbf{g}_k + \sigma_s \mathbf{S}_k. \tag{3.13}$$

again, here ϕ is the (modeled) value of mean attenuation at $\lambda = 0$. The $M \times 1$ vector $\mathbf{g}_k$ is defined as

$$\mathbf{g}_k \triangleq \left[\; g_k^1 \ldots g_k^{(j)} \ldots g_k^M \; \right]^T. \tag{3.14}$$

and

$$g_k^{(j)} \triangleq \exp\{-\frac{\lambda_k^{(j)}}{2\sigma_\lambda}\}. \tag{3.15}$$



(a) Histogram of Modeled-Mean-Removed attenuation level

(b) Quantile-quantile plot of experimental data versus standard Gaussian model

Fig. 3.3 Histogram and QQ plot of experimental data

The parameter σ_λ controls the rate of decay of the mean attenuation with respect to λ for link j at time step k. Once given the target state $\mathbf{x}_k$, λ can be calculated by equation (3.7). In equation (3.13), σ_s and $\mathbf{S}_k$ represent the measurement "noise" when we calculate the attenuation levels for different λ.

To examine the statistics of the noise for pixel-free model, we first calculate the modeled-mean through equation (3.15). By subtracting the modeled-mean from the individual attenuation level, we convert the scatter plot of "Experimental attenuation level" in Figure 3.2 to "Histogram of experimental data" in Figure 3.3(a). The histogram curve shows the Modeled-Mean-Removed attenuation level while the dashed curve is the best-fit Gaussian model based on the measurement data. Figure 3.3(b) is a quantile-quantile plot also comparing the mean-removed attenuation level with a Gaussian distribution $\mathcal{N}(0, \sigma_g^2)$, where σ_g is the standard deviation of experimental data. It is clear that the experimental noise distribution has more mass near the mean and lighter tails than the best fit Gaussian, but the distribution is approximately symmetric about the mean. An additive noise model (in the decibel domain) appears to be an appropriate choice. In filtering problems, tracking performance is often improved if the adopted noise model has slightly heavier tails than

the true noise distribution. For this reason, and because it reduces the computational complexity, we adopt a Gaussian model for the observation noise. Thus, in equation (3.13), $\mathbf{S}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{M\times 1})$; the parameter σ_s is the standard deviation of the noise independent of λ.

As a result, a relationship between the target state $\mathbf{x}_k$ and the measurement $\mathbf{y}_k$ is established by pixel-free model. The proposed attenuation measurement model has three unknown parameters, ϕ, σ_λ, and σ_s. After conducting multiple experiments, we have concluded that the value of σ_λ that provides the best fit to the observed data varies little for different (human) targets and surveillance environments. We have observed considerably more variation in the best-fit values of ϕ and σ_s.

3.3 On-line Sequential Monte Carlo Algorithm

We adopt a Sequential Monte Carlo (particle filtering) framework to perform the tracking and use an on-line EM approach to sequentially update the estimates of unknown static parameters. These include noise standard deviation and model parameter of measurement model, σ_s and ϕ respectively, and one parameter that represents the noise standard deviation of the transition model, σ_v. The parameter in the measurement model, σ_λ, which controls the decay rate of the mean attenuation is set to a constant empirical value, according to the experimental data. Therefore, we can denote the static parameters θ as

$$\theta = [\sigma_v, \sigma_s, \phi]. \tag{3.16}$$

The transition model we use is a one-tap autoregressive (AR-1) Gaussian model:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{v}_k) \tag{3.17}$$

$$= a\mathbf{x}_k + \sigma_v \mathbf{V}_k. \tag{3.18}$$

where $\mathbf{x}_k$ is the state vector of a target in the 2D plane, $\mathbf{x}_k = [\alpha_k, \beta_k]^T$. $\mathbf{V}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{2\times 1})$, and σ_v is assigned a initial value and estimated by on-line EM. The constant $a < 1$ models a (small) drift towards the center of the surveillance region; we choose a as a constant that is close to 1, so that the drift is very small.

There are two main motivations for the adoption of this model: (i) it assumes little

knowledge about the nature of the motion; (ii) the on-line EM methodology we adopt requires that the target process is stationary and ergodic (which eliminates a pure random walk process).

The principal measurement model we use in the on-line sequential Monte Carlo algorithm is the pixel-free model:

$$\mathbf{y}_k = h(\mathbf{x}_k, \mathbf{s}_k) \tag{3.19}$$

$$= \phi \mathbf{g}_k + \sigma_s \mathbf{S}_k. \tag{3.20}$$

At time step k, the state $\mathbf{x}_k$ determines the elliptical parameter $\lambda_k^{(j)}$ which we described in section 3.2, and thus $\mathbf{g}_k$ describes the influence of target state $\mathbf{x}_k$ on M links ($j = 1, \cdots, M$). For example, if the location of the object $\mathbf{x}_k$ is far away from the link j, it results in a large $\lambda_k^{(j)}$ value, and an associated small attenuation level. In other words, the greater the distance between the target location and the principal line of ellipse on link j, the smaller the attenuation that is assigned via $\mathbf{g}_k$.

Both $\mathbf{y}_k$ and $\mathbf{g}_k$ have dimension $M \times 1$ in our simulation, which depends on the link number M in sensor network at time step k.

3.3.1 Auxiliary Particle Filtering

We apply auxiliary particle filtering to track the marginal posterior distribution $p_\theta(\mathbf{x}_k|\mathbf{y}_{1:k})$. As we mentioned in section 2.3, we suppose that samples $\{\mathbf{x}_{0:k}^{(i)}; i = 1, \cdots, N\}$ are drawn independently from an importance distribution. The auxiliary particle filter, which builds on the SIR particle filter, outperforms the standard particle filter, especially in noisy environments.

Based on the chosen state dynamics and measurement functions $f(\cdot)$ and $h(\cdot)$, we can perform pointwise evaluation of the likelihood functions $p(\mathbf{y}_{1:k}|\mathbf{x}_{1:k})$ given the true observation $\mathbf{y}_{1:k}$. The particle filter strives to calculate at each time step k a weighted particle approximation $\{\mathbf{x}_{1:k}^{(i)}, \omega_k^{(i)}\}_{i=1}^N$ to the posterior of interest $p(\mathbf{x}_{1:k}|\mathbf{y}_{1:k})$. In our algorithm, the importance density $\pi(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})$ is chosen to be the prior distribution for simplicity (as explained in section 2.3):

$$\pi(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{y}_{1:k}) = p(\mathbf{x}_k|\mathbf{x}_{k-1}). \tag{3.21}$$

Given this choice, the weights of particles can be obtained

$$\omega_k^{(i)} \propto \omega_{k-1}^{(i)} p(\mathbf{y}_k|\mathbf{x}_k^{(i)}). \tag{3.22}$$

A resampling procedure is then applied in every time step. It replicates particles with high weights and eliminates those with low weights. A comparison in [20] shows that the residual resampling method, which we discussed in section 2.3, has a better performance with particle weights exhibiting lower conditional variance in general. Thus we apply the residual resampling method in our algorithm. After the resampling performed at each time step, we have $\omega_{k-1}^{(i)} = 1/N$.

Particle filter can perform poorly if the importance function $\pi(\cdot)$ does not adequately take into account the information available in the observation $\mathbf{y}_{1:k}$. Therefore, the filtering can be inefficient and the robustness to outliers will be weak. In addition, the resampling procedure in such a case makes a quick loss of diversity of the particles and will results in the degeneracy problem.

The auxiliary particle filter algorithm is first initialized at time step $k = 0$ with samples $\mathbf{x}_0^{(i)}$ drawn from distribution $p(\mathbf{x}_0)$. It then modifies the sampling step in an attempt to improve performance. After the initialization, the auxiliary particle filter additionally calculates the *first-stage weights* $\rho_k^{(i)}$ for each particle, it will be used as a 'reference' afterwards to evaluate how well each particle can represent the observations $\mathbf{y}_{1:k}$. Thus we have

$$\rho_k^{(i)} = p(\mathbf{y}_k|\mu_k^{(i)}). \tag{3.23}$$

as mentioned in section 2.3, the $\mu_k^{(i)}$ is some characterization of $\mathbf{x}_k$ given $\mathbf{x}_{k-1}^i$. In our tracking algorithm, $\mu_k^{(i)} = \mathbb{E}[\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)}]$. Based on $\rho_k^{(i)}$, normalized weights $\widetilde{W}_k^{(i)}$ can then be obtained. The weighted particles are resampled in the residual resampling step, a auxiliary variable $\prime(i)$ is introduced in this phase. We then update the states $\mathbf{x}_{1:k-1}^{(i)}$ and the first-stage weights $\rho_k^{(i)}$ according to new index $\prime(i)$. Eventually, new particle weights $\omega_k^{(i)}$ can be calculated using both likelihood distribution $p(\mathbf{y}_k|\mathbf{x}_k^{(i)})$ and the first-stage weights. The auxiliary particle filter algorithm then recursively approximates the samples, we calculate the mean value over these samples at each time step to obtain an estimated target states. The auxiliary particle filter is specified in Algorithm 4.

Algorithm 4: Auxiliary Particle Filter

`// Initialization at time` $k = 0$

for $i = 1, \ldots, N$ **do**
  Sample the particles $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)$;
  Set weights $\omega_0^{(i)} = p(\mathbf{y}_1|\mathbf{x}_0^{(i)})$;
end for
Normalize weights $\widetilde{\omega}_0^{(i)} = \frac{\omega_0^{(i)}}{\sum_{i=1}^N \omega_0^{(i)}}$;

`// For times` $k > 0$

for $k = 1, \ldots, T$ **do**
  `// First-stage weights`
  for $i = 1, \ldots, N$ **do**
    Calculate $\mu_k^{(i)} = \mathbb{E}[\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)}]$;
    Sample $\rho_k^{(i)} \sim p(\mathbf{y}_k|\mu_k^{(i)})$;
    Set weights $W_k^{(i)} = \widetilde{\omega}_{k-1}^{(i)} \times \rho_k^{(i)}$;
  end for
  Normalize weights $\widetilde{W}_k^{(i)} = \frac{W_k^{(i)}}{\sum_{i=1}^N W_k^{(i)}}$;
  `// Residual Resampling`
  Resample from the set $\left\{\mathbf{x}_{k-1}^{(i)}, \widetilde{W}_k^{(i)}\right\}_{i=1}^N$ to obtain $\left\{\mathbf{x}_{k-1}^{\prime(i)}, \frac{1}{N}\right\}_{i=1}^N$;
  for $i = 1, \ldots, N$ **do**
    Set $\mathbf{x}_{1:k-1}^{(i)} = \mathbf{x}_{1:k-1}^{\prime(i)}$ and $\rho_k^{(i)} = \rho_k^{\prime(i)}$;
    Sample $\mathbf{x}_k^{(i)} \sim p(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)})$;
    Set weights $\omega_k^{(i)} = \frac{p(\mathbf{y}_k|\mathbf{x}_k^{(i)})}{p(\mathbf{y}_k|\mu_k^{(i)})}$;
  end for
  Normalize weights $\widetilde{\omega}_k^{(i)} = \frac{\omega_k^{(i)}}{\sum_{i=1}^N \omega_k^{(i)}}$;
  `// Optional second resample`
  Resample $\left\{\mathbf{x}_{1:k}^{(i)}, \widetilde{\omega}_k^{(i)}\right\}_{i=1}^N$ to obtain $\left\{\mathbf{x}_{1:k}^{(i)}, \frac{1}{N}\right\}_{i=1}^N$;
end for

3.3.2 On-line EM

The unknown parameter vector θ in the auxiliary particle filter needs to be initialized and estimated before running the auxiliary particle filter algorithm. Since we do not have knowledge of the parameters, we need to estimate them on-line while tracking the target. We use an on-line EM algorithm described in section 2.4 to form estimates of the parameters.

The on-line EM algorithm maximizes a pseudo-likelihood function in order to form point estimates of the parameters θ. Recursive maximization of the likelihood functions themselves, $p(\mathbf{y}_{1:k}|\theta)$, would require estimation of statistics based on probability distributions whose dimension is growing in time. The substitution of the pseudo-likelihood leads to calculations in a fixed dimension.

The on-line EM algorithm updates the parameters every L time steps. We define $\mathbf{X}_b \triangleq \mathbf{x}_{bL+1:(b+1)L}$ and $\mathbf{Y}_b \triangleq \mathbf{y}_{bL+1:(b+1)L}$, where b is the index of the block. The log pseudo-likelihood function employed in [8] is defined, for m blocks, as

$$l(\hat{\theta}) = \sum_{b=1}^{m} \log p_{\hat{\theta}}(\mathbf{Y}_b), \tag{3.24}$$

where

$$p_{\hat{\theta}}(\mathbf{Y}_b) = \int p_{\hat{\theta}}(\mathbf{x}, \mathbf{Y}_b) d\mathbf{x}, \tag{3.25}$$

If the process $\mathbf{x}_k$ is stationary and ergodic, the average log pseudo-likelihood satisfies

$$\bar{l}(\hat{\theta}) = \int \log p_{\hat{\theta}}(\mathbf{y}) p_{\theta}(\mathbf{y}) d\mathbf{y}, \tag{3.26}$$

where $\hat{\theta}$ is the estimate value of θ [8]. Rydén showed in [65] that an algorithm which can maximize $\bar{l}(\hat{\theta})$ will identify the true value of θ. We therefore apply on-line EM to recursively maximize $\bar{l}(\hat{\theta})$ by updating the estimate of θ via

$$\hat{\theta}_b = \arg\max_{\theta \in \Theta} Q(\theta, \hat{\theta}_{b-1}), \tag{3.27}$$

where

$$Q(\theta, \hat{\theta}_{b-1}) = \int \log(p_{\hat{\theta}}(\mathbf{x}, \mathbf{y})) p_{\hat{\theta}_{b-1}}(\mathbf{x}|\mathbf{y}) p_{\theta}(\mathbf{y}) \mathbf{dxdy} \tag{3.28}$$

The direct computation of Q cannot be performed, but we can replace (3.27) by the update $\hat{\theta}_b = \Lambda(\mathbf{\Omega}(\hat{\theta}_{b-1}, \theta))$, where $\mathbf{\Omega}(\hat{\theta}_b, \theta)$ is a set of sufficient statistics and Λ is a mapping function from the sufficient statistics $\mathbf{\Omega}(\hat{\theta}, \theta)$ to the $\hat{\theta}$ that maximizes Q.

Four sufficient statistics are required for the three parameters ϕ, σ_s, and σ_v. These are of the form

$$\begin{aligned} \mathbf{\Omega}(\hat{\theta}_{b-1}, \theta) &= [z_1, z_2, z_3, z_4] \\ &= E_{\hat{\theta}_{b-1}, \theta}[\psi_1, \psi_2, \psi_3, \psi_4]. \end{aligned} \tag{3.29}$$

The expectation is with respect to $p_{\hat{\theta}_{b-1}}(\mathbf{x}|\mathbf{y}) p_{\theta}(\mathbf{y})$ and

$$\psi_1(\mathbf{X}_b, \mathbf{Y}_b) = \sum_{k=bL+2}^{(b+1)L} ((\mathbf{x}_k - \mathbf{x}_{k-1})^T (\mathbf{x}_k - \mathbf{x}_{k-1})) \tag{3.30}$$

$$\psi_2(\mathbf{X}_b, \mathbf{Y}_b) = \sum_{k=bL+1}^{(b+1)L} ((\mathbf{y}_k - \phi \mathbf{g}_k)^T (\mathbf{y}_k - \phi \mathbf{g}_k)). \tag{3.31}$$

$$\psi_3(\mathbf{X}_b, \mathbf{Y}_b) = \sum_{k=bL+1}^{(b+1)L} (\mathbf{y}_k^T \mathbf{g}_k) \tag{3.32}$$

$$\psi_4(\mathbf{X}_b, \mathbf{Y}_b) = \sum_{k=bL+1}^{(b+1)L} ||\mathbf{g}_k||_2^2. \tag{3.33}$$

The maximization function Λ is defined as

$$\sigma_{vb} = \sqrt{\frac{z_1(\hat{\theta}_{b-1}, \theta)}{2(L-1)}} \tag{3.34}$$

$$\sigma_{sb} = \sqrt{\frac{z_2(\hat{\theta}_{b-1}, \theta)}{ML}} \tag{3.35}$$

$$\phi_b = \frac{z_3(\hat{\theta}_{b-1}, \theta)}{z_4(\hat{\theta}_{b-1}, \theta)}. \tag{3.36}$$

The expectations cannot be computed, because they are with respect to a measure that involves the unknown true value θ. But the sufficient statistics can be recursively estimated. The ergodicity and stationarity assumptions for the process imply that the blocks $\mathbf{Y}_b$ are samples from $p_\theta(\mathbf{y})$ and they can therefore be used for Monte Carlo integration. We can thus form the following update of the statistics

$$\hat{\mathbf{\Omega}}_b = (1-\alpha_b)\hat{\mathbf{\Omega}}_{b-1} + \alpha_b E(\Phi(\mathbf{X}, \mathbf{Y}_b)|\mathbf{Y}_b) \tag{3.37}$$

where the expectation is with respect to $p_{\hat{\theta}_{b-1}}(\mathbf{x}|\mathbf{Y}_b)$. We then substitute $\hat{\mathbf{\Omega}}_{b-1}$ for $\mathbf{\Omega}(\hat{\theta}_b, \theta)$ and obtain $\hat{\theta}_b = \Lambda(\hat{\mathbf{\Omega}}_b)$. Setting $\hat{\theta}_b$ to a constant value and $\alpha_b = 1/b$, $\hat{\mathbf{\Omega}}_b$ will simply compute the arithmetic average of $E(\Phi(\mathbf{X}, \mathbf{Y}_b)|\mathbf{Y}_b)$, this ensures convergence of $\hat{\mathbf{\Omega}}_b$ to $\mathbf{\Omega}(\hat{\theta}_b, \theta)$ [8]. The maximization step then becomes $\hat{\theta}_b = \Lambda(\hat{\Omega}_b)$.

As one final approximation, since $E(\Phi(\mathbf{X}, \mathbf{Y}_b)|\mathbf{Y}_b)$ does not have an analytical solution, we can use importance sampling, using the particle tracks and weights calculated by the auxiliary particle filter

$$\hat{\mathbf{\Omega}}_b = (1-\alpha_b)\hat{\mathbf{\Omega}}_{b-1} + \alpha_b \sum_{m=1}^{K} \omega_b^{(m)} \psi(\mathbf{X}_b^{(m)}, \mathbf{Y}_b). \tag{3.38}$$

3.3.3 On-line Sequential Monte Carlo Algorithm

The complete algorithm, combining the auxiliary particle filter and the on-line EM, is described in Algorithm 5 below.

The computational cost consist of two parts: auxiliary particle filtering and on-line EM. The complexity of the particle filter tracking algorithm is $O(MN)$ per time step, where M is the number of links and N is the number of particles used for tracking. To specify, the number of operations per auxiliary particle filtering step is $36MN$, including particle approximation using measurement model ($26MN$) and calculation of particle weights ($6MN$). The on-line EM algorithm, which is only executed every L time-steps, has a complexity of $O(LMN)$, the number of operations per on-line EM step is $4MNL$. In other words, the per time-step complexity of on-line EM algorithm is $O(MN)$. In our typical experimental setting, the block length to execute on-line EM is $L = 10$ time steps, link number $M = 276$ (using 24 sensor nodes) and we use $N = 1000$ particles per time step. The computational cost is sufficiently small that the tracking system can collect the data packets and to per-

form real-time tracking using a standard laptop (CPU: Core 2 Duo T5670 1.8GHz, RAM 1GB in our experiment).

Algorithm 5: SMC RF Tomographic Tracking

`// Initialization at time` $k = 0$

Initialize θ_0 and set $b = 1$;
for $i = 1, \ldots, N$ **do**
  Sample the particles $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)$;
  Set weights $\omega_0^{(i)} = p(\mathbf{y}_1|\mathbf{x}_0^{(i)})$;
end for
Normalize weights $\widetilde{\omega}_0^{(i)} = \frac{\omega_0^{(i)}}{\sum_{i=1}^{N} \omega_0^{(i)}}$;
for $k = 1, 2, \ldots, T$ **do**
  `// Auxiliary Particle Filtering`
  for $i = 1, \ldots, N$ **do**
    $\{\mathbf{x}_k^{(i)}, \omega_k^{(i)}\} = APF(\{\mathbf{x}_{k-1}^{(i)}, \omega_{k-1}^{(i)}\})$;
  end for
  `// On-line EM`
  if $k \mod L = 0$ **then**
    `// E-step`
    for $i = 1, \ldots, N$ **do**
      Calculate $\omega_b^{(i)} = \frac{p_{\theta_{b-1}}(\mathbf{X}_b^{(i)}|\mathbf{Y}_b)}{\pi_{\theta_{b-1}}(\mathbf{X}_b^{(i)}|\mathbf{Y}_b)}$;
    end for
    Normalize weights $\{\omega_b^{(i)}\}$ such that $\sum_{i=1}^{N} \omega_b^{(i)} = 1$;
    Update $\hat{\mathbf{\Omega}}_b = (1 - \alpha_b)\hat{\mathbf{\Omega}}_{b-1} + \alpha_b \sum_{m=1}^{N} \omega_b^{(m)} \psi(\mathbf{X}_b^{(m)}, \mathbf{Y}_b)$;
    `// M-step`
    Set $\theta_b = \Lambda(\hat{\mathbf{\Omega}}_\mathbf{b})$ and $b = b + 1$;
  end if
end for

Chapter 4

Simulations and Experiments

In this chapter we present the results of both simulations and experiments conducted to explore the performance of the proposed algorithm. First, we simulated different sensor network layouts in Matlab including a square, a circle and an irregularly shaped layout. In each case, we had a single target walking one of two trajectories (a square route or a zigzag route). Based on the simulated data sets, estimated trajectories were generated and compared with the ground-truth. Numerical evaluations of tracking performance are provided and a comparison with the previous RF tomographic method from [84] under the same setting is also presented. Second, in the experimental section, we provide a description of the field experiments conducted in outdoor fields, both with and without obstructions (trees). The experimental sensor network layouts and the target walking paths are similar to those in simulation. We present both graphical and numerical evaluations of tracking performance based on experimental data sets. The on-line SMC algorithm operates in Matlab for both simulation and experiment data.

4.1 Simulations

The Matlab simulation mimics a wireless sensor network with 24 nodes, similar to the one that we have used for experiments. All sensor nodes are able to transmit and receive data packets. There exists a node No.25 attached to the processing center (a laptop computer). This acts as a master node (or sink node). This master node receives all the data packets sent by the rest of the nodes, collecting them for further processing. It also broadcasts the commands from the controller to the sensor network, such as reset or launch. An example layout is shown in Figure 4.1. This example shows a person walking clockwise along a specified square route (the blue dashed line with solid circles) starting from the bottom left corner.

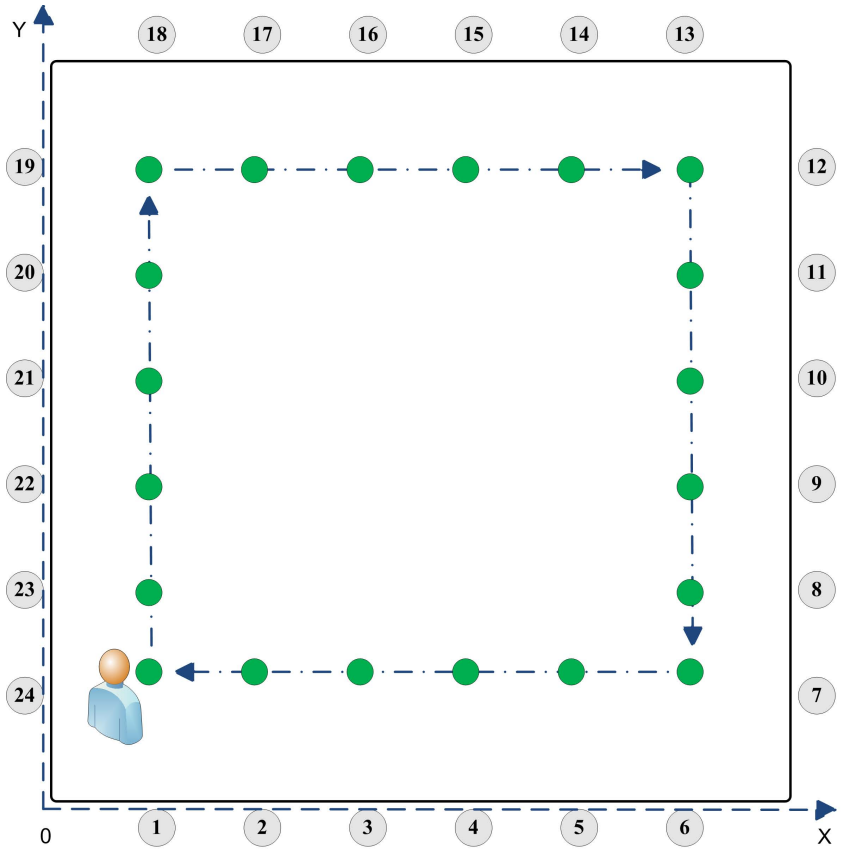


Fig. 4.1 Scenario example: 24 nodes are placed around the square area, a single target follows a square trajectory (dashed arrows with circles) clockwise with a square sensor network layout. The trajectory starts from the bottom left corner and ends at the same point after one cycle.

The ground-truth target trajectory and the unknown parameters in set θ are set before the tracking in run. When the target walks across the ellipse link area of a pair of nodes at time step k, an attenuation occurs on its received signal strength and the numerical changes of the signal strength are calculated according to the measurement model. In each scenario, we generated 100 realizations for 3 different measurement noise standard deviations, with these noise values being set according to data from real outdoor experiments. Every set of measurements in our simulation was recorded once per second (i.e, each time step corresponded to one second).

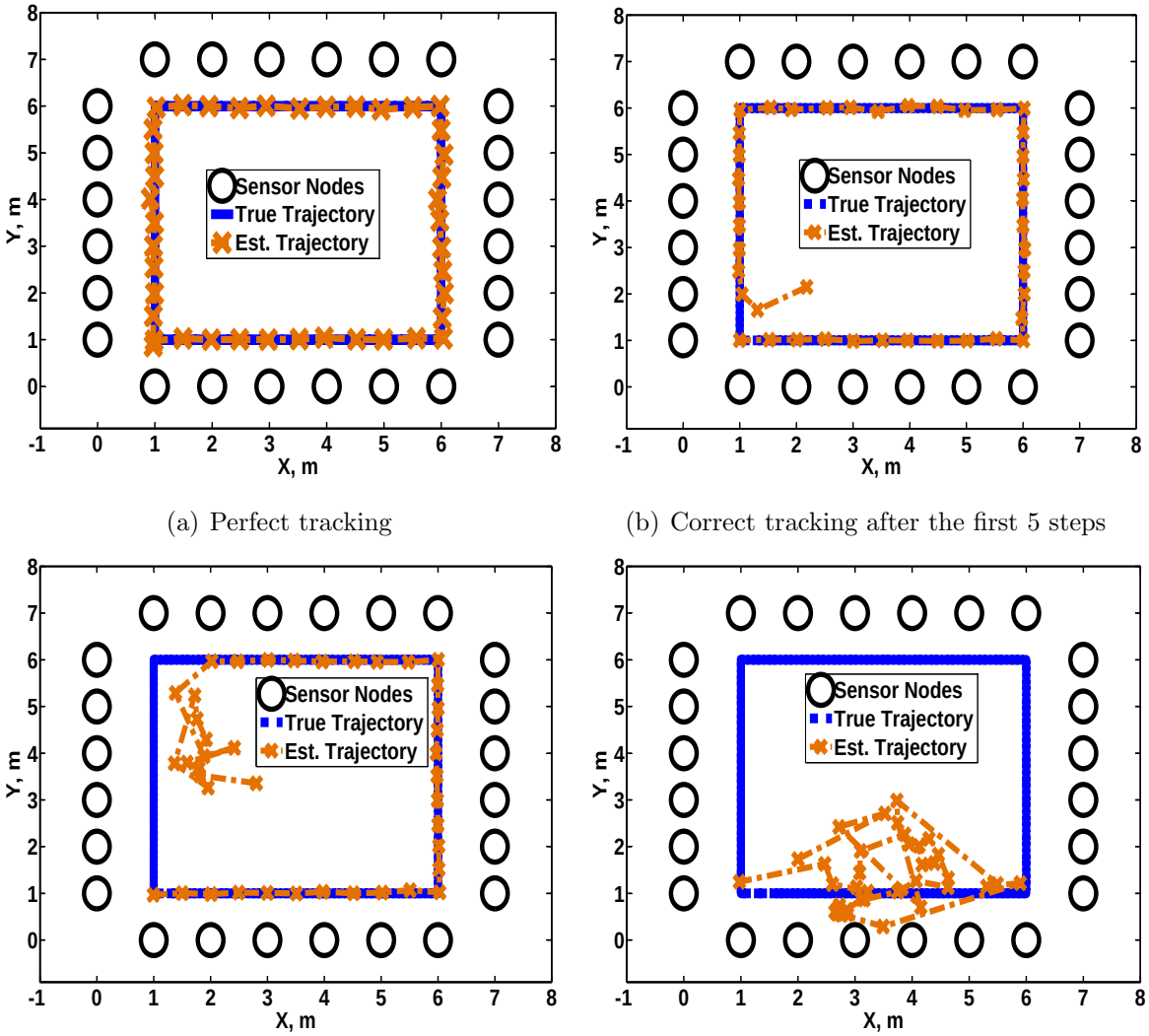
In the rest of this section, we will show our results including tracking results, parameter estimation and the Root Mean Square Error (RMSE) of the example scenario in Figure 4.1. We will also compare the numerical RMSE values access varying noise levels and square sensor networks dimensions. To demonstrate the advantage of our method, we then compare our performance with that of tracking using the previous proposed imaging method and the pixelized measurement model [84]. Afterwards, we present results from a zigzag trajectory in the same square layout and also from other scenarios including a circular layout and an irregular layout with both square and zigzag trajectories. Numerical RMSE values of different scenarios are presented in tables for comparison. We simulated all the above scenarios using the proposed pixel-free measurement model and tracked the target using the on-line sequential Monte Carlo algorithm.

All the simulations mentioned above assume that the node positions are known to the tracking system. However, sometimes the node positions may not be known or may only be partially known in emergency applications or in large-scale sensor deployment, where people may only have a short time to deploy the sensors in a new environment. This requires a robustness in the tracking system so that the target trajectory can still be estimated with reasonable accuracy when the positions of most nodes are not known accurately. Therefore, at the end of the simulation section, we present the performance of the on-line sequential Monte Carlo algorithm in target tracking when noise is added to the positions of the sensor nodes.

4.1.1 Square Trajectory in a Square

As a basic example, we first simulate sensor nodes that deployed in a $7m \times 7m$ square with a spacing of 1 metre. A person walked clockwise within the network area along the

square route specified in Figure 4.1 with a speed of $0.5m/sec$. We set the bottom left corner point of the square as the origin of the 2-D coordinate system, so the person started walking from point (1,1) follows a square trajectory, and stopped at the same point after one cycle. Since we know in prior the exact locations of the object at each time step, the square trajectory is set as 'ground-truth' trajectory in this simulation scenario.



(a) Perfect tracking

(b) Correct tracking after the first 5 steps

(c) Correct tracking after the first 15 steps

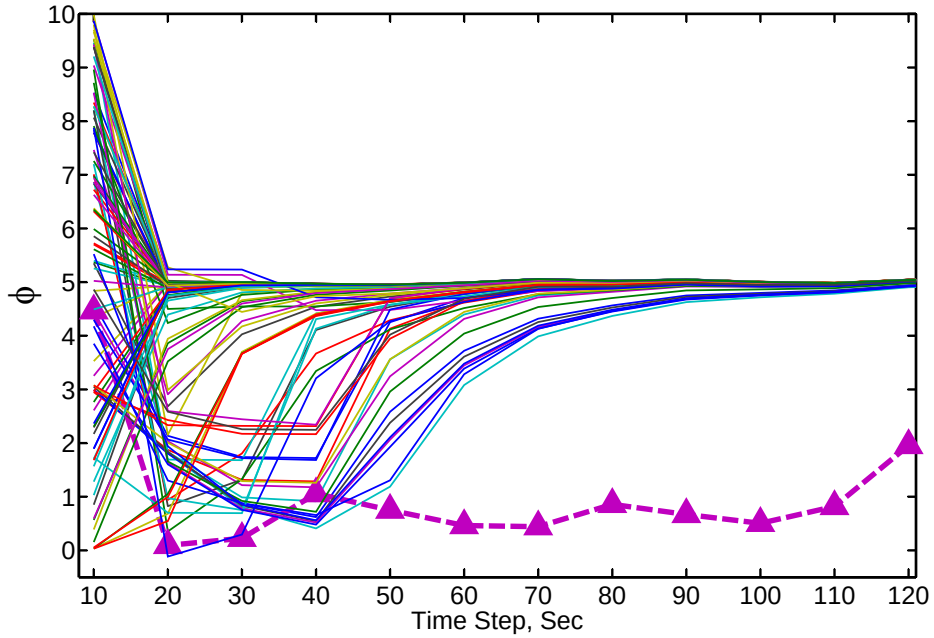
(d) Lost track

Fig. 4.2 Simulation examples: On-line SMC estimation of a square target trajectory in a $7m \times 7m$ square sensor network layout with $\sigma_s = 1$, $\sigma_v = 0.3$ and $\phi = 5$.

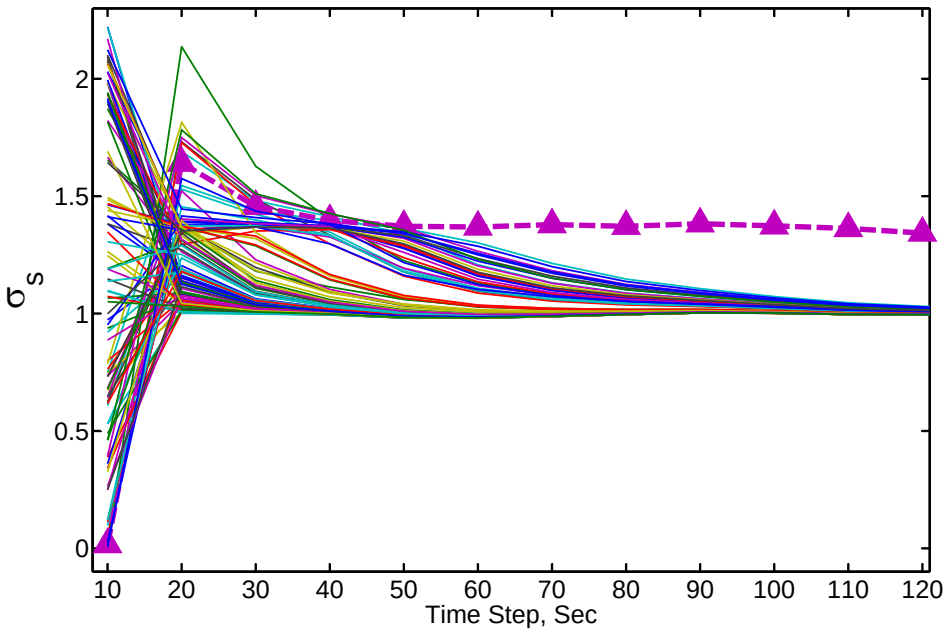
Given this ground-truth trajectory, observation data for 100 realizations were generated from each of the various measurement noise levels: $\sigma_s = 0.5, 1, 2, \sqrt{5}$, using the pixel-free model. Other parameters are set as: $\sigma_v = 0.3$ in the transition model, and $\sigma_\lambda = 0.02$ and $\phi = 5$ in the pixel-free model. These values were chosen because they provided a good fit to our experimental data when a real person walked at a speed $0.5m/sec$ in an outdoor field. In the sequential Monte Carlo algorithm, we used 1000 particles. The unknown parameter values in the on-line EM algorithm were initialized by drawing from the following uniform distributions: $\sigma_s \sim U(0, \sqrt{5}]$, $\phi \sim U(0, 10]$ and $\sigma_v \sim U(0, 1]$.

Four examples of the SMC-estimated trajectory for the case $\sigma_s = 1$ are shown in Figure 4.2. Depending on the initial guess and the initial particle distribution, correct tracking may start at different time steps. Figure 4.2 shows selected results (from (a) to (d)) from the best case to the worst case. With the initial guess of both θ and particle samples drawn from different uniform distributions, the on-line sequential Monte Carlo algorithm is able to track the target in the vast majority of realizations, with only 1 lost track occurring in the 100 realizations. We define 'lost track' as having occurred when the average square error after the 60th time step is larger than $1m$. For example, if there is a total of 120 time steps, we calculate the average square error from the 60th time step to the 120th time step to determine whether a lost track has occurred. Under the same parameter settings, we generated simulated observations for 100 times, the trajectories were estimated using on-line sequential Monte Carlo algorithm. Over all these realizations, the lost track rate is 2%. It is clear that, under this setting, most tracks provide accurate approximations of the ground-truth trajectory.

For almost all the tracks, the on-line EM succeeds in providing acceptably accurate estimates of the unknown parameters, leading to improved performance over time. Figure 4.3 shows the evolution of the estimates of ϕ and σ_s over the same simulated set of data with $\sigma_s = 1$ in the $7m \times 7m$ square. Initialized with different values draw from uniform distributions $\sigma_s \sim U(0, \sqrt{5}]$ and $\phi \sim U(0, 10]$, almost all estimates converge to values close to the true value after 50-60 seconds (5-6 update steps in on-line EM). The triangle dashed lines in each sub-figure of Figure 4.3 denote the estimates which came from the lost track. This illustrates the performance of the on-line EM and sequential Monte Carlo tracking algorithm depend on each other.

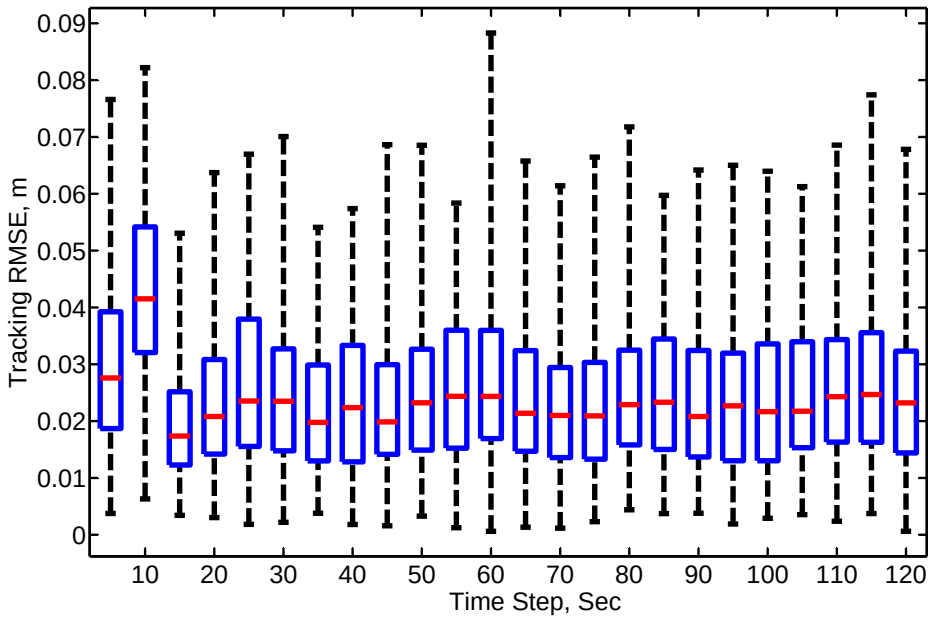


(a) Estimation of ϕ(ground-truth $\phi = 5$)

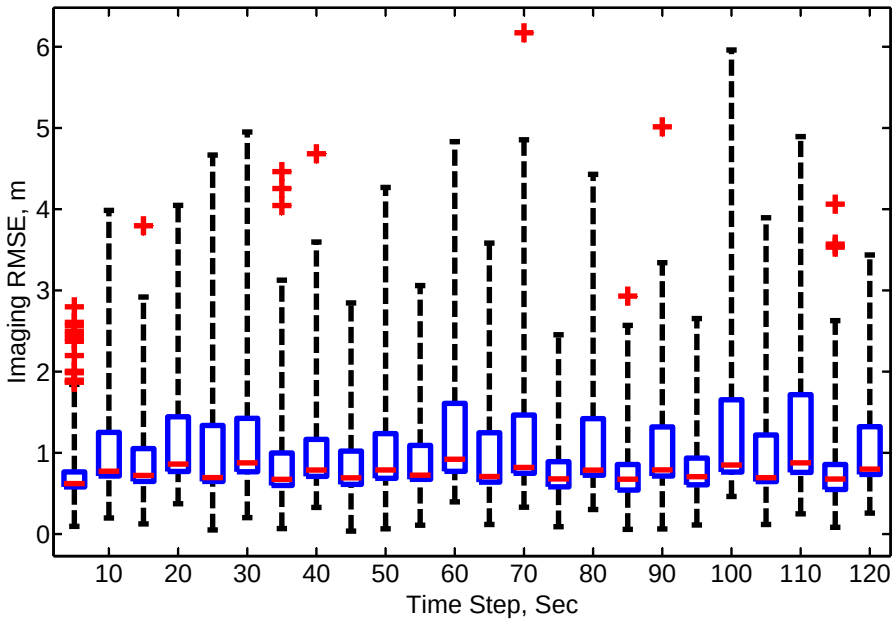


(b) Estimation of σ_s(ground-truth $\sigma_s = 1$)

Fig. 4.3 Behaviour of the parameter estimation of *phi* and σ_s for the square target trajectory in a $7m \times 7m$ square sensor network layout in the On-line SMC estimation. The triangle dashed lines in both figures indicate the failure of parameter estimation in the lost track realization.



(a) RMSE of On-line SMC algorithm



(b) RMSE of imaging algorithm

Fig. 4.4 RMSE of (a) On-line SMC estimation, (b) imaging plus Kalman filter estimation, for the square target trajectory in a $7m \times 7m$ square sensor network layout with $\sigma_s = 1$. The boxes range from the 25th to 75th quantiles, the whiskers extend 3 times the interquartile range, the median is marked as a line within the box, and the pluses indicate outliers.

Figure 4.4(a) shows a box-and-whisker plot of the RMSE for a set of 99 realizations (the same set of 100 realizations with the lost track eliminated). The RMSE decreases rapidly after the first 10 time steps and stays at a constant low level with a median value ranges from 0.02 to 0.05 metres in the $7m \times 7m$ square area.

Comparison between On-line SMC and Imaging

We compare the performance of the sequential Monte Carlo algorithm with that of the imaging plus Kalman filter algorithm of [83]. Under the same scenario (target speed $0.5m/sec$ in a $7m \times 7m$ square with $\sigma_s = 1, \sigma_v = 0.3$ and $\phi = 5$), we implemented the imaging plus Kalman filter method and the pixelized model. For the pixelized model, pixel width $\Delta p = 0.15$m, $\lambda = 0.02$, and $\phi = 15$ (again these values provide a good fit to experimental data). In the imaging plus Kalman filter algorithm, the regularization parameter α is set to 200. The transition noise σ_v for the Kalman filter is also set to 0.3. Figure 4.4(b) shows the RMSE performance of imaging estimation. Compared with RMSE of tracking estimation in Figure 4.4(a), we can observe that imaging has a relatively constant RMSE level, at approximately 0.8 m, throughout the whole estimation process.

Tables 4.1 and 4.2 compare the RMSE of the sequential Monte Carlo and the imaging plus Kalman filter algorithms using the data generated from the pixelized model and the pixel-free model, respectively. When perform target estimation, the on-line sequential Monte Carlo algorithm operates with the pixel-free model and the imaging plus Kalman filter algorithm operates with the pixelized model. The measurement noise level varies from 0.5 to $\sqrt{5}$. At a given noise level, the RMSE values of sequential Monte Carlo algorithm are much lower (7 or 8 times) than those of the imaging plus Kalman filter algorithm in both tables.

Noise Std. Dev.	SMC (m)	Imaging (m)
0.5	0.0927	0.7852
1	0.0964	0.7852
2	0.1015	0.8033
$\sqrt{5}$	0.1257	0.8152

Table 4.1 RMSE comparison between SMC and imaging algorithms using data from pixelized model

When data is generated from the pixelized model, we can observe that the RMSE values of both algorithms increase more slowly than those based on data from the pixel-free model. Even using data from the pixelized model, the sequential Monte Carlo algorithm still significantly outperforms the imaging plus Kalman filter algorithm.

Noise Std. Dev.	SMC (m)	Imaging (m)
0.5	0.0316	0.3071
1	0.0436	0.6993
2	0.0988	1.1451
$\sqrt{5}$	0.1664	1.4848

Table 4.2 RMSE comparison between SMC and imaging algorithms using data from pixel-free model

RMSE Performance With Varying Noise Levels and Network Dimensions

To further evaluate the performance of the algorithm, we varied both the measurement noise levels and the network dimensions and calculated the average RMSE for the target estimation. The noise standard deviation ranged from 0.5 to 3, and the network dimensions ranges from $7m \times 7m$ to $35m \times 35m$. The other parameters and settings were the same as in subsection 4.1.1, and data was generated from the pixel-free model. We ran 100 realizations for each. To calculate the RMSE value, we first calculated the mean RMSE over all time steps in each realization, then we identified the realizations for which the tracker lost the target trajectory, as we described in the previous subsection. The final average RMSE was calculated based on the mean RMSE of all realizations excluding the lost tracks.

RMSE of tracking estimation using On-line SMC algorithm for a zigzag trajectory in various (a square, a circle and a irregular shape) layouts of roughly $50m^2$ each and a square trajectory in a $7m \times 7m$ square layout, with varying noise level σ_s. The legend follows the form "Trajectory/Node Layout".

Figure 4.5 shows the trend of the RMSE performance over varying noise levels and network dimensions. As measurement noise σ_s increases, the RMSE values under the same network dimension increase as well. Increasing the network dimensions while keep the total number of nodes the same leads to an increase in distance between each pair of nodes and a decrease in node density. The attenuation is inversely proportional to the distance between

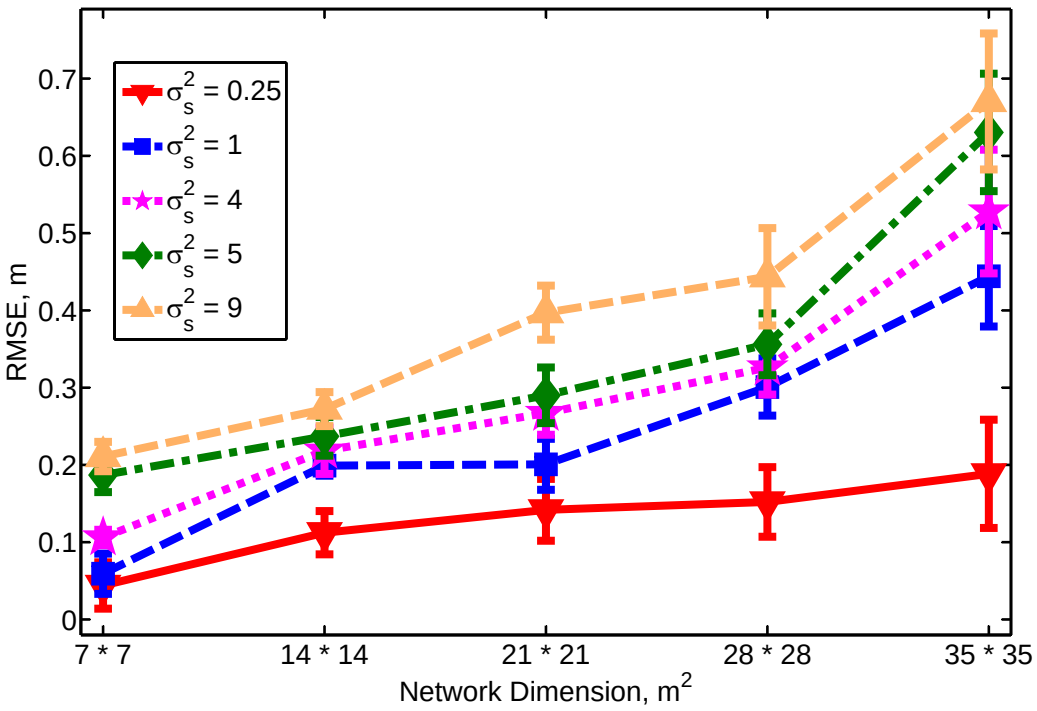


Fig. 4.5 RMSE of tracking estimation using On-line SMC algorithm for a square trajectory in a $7m \times 7m$ square layout, with varying noise levels σ_s and network dimensions.

the sensors, so the λ will grow larger as the network size increases, thus the attenuation caused by a target in a particular link will decrease. Therefore the signal to noise ratio decreases, leading to higher errors.

However, although the target estimation is poorer with lower node density, the RMSE values still show that we are providing a good approximation of the ground-truth trajectory. The error of the target trajectory is relatively low compared to the network size. Most estimated trajectories are able to follow the ground-truth trajectory when the target moves in straight lines, even with a high noise standard deviation of $\sigma_s = 3$, and the RMSE remains at an acceptable 0.7 m for the $35m \times 35m$ network dimension.

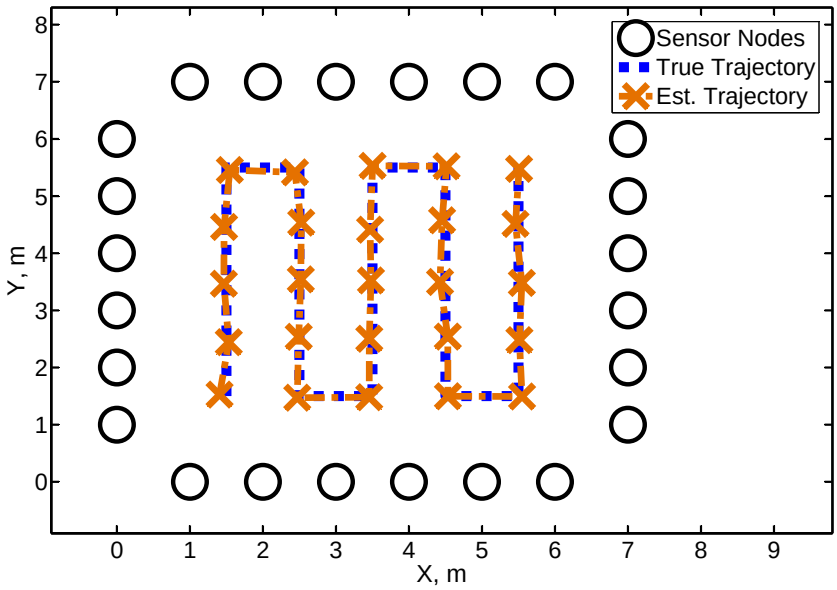
Over 100 realizations for each of the setting, the lost track ratio increases when either noise level increases or network dimension increases. For a same $7m \times 7m$ network dimension, the lost track ratio increases from 1% to 4% when the noise level increase from $\sigma_s = 0.5$ to $\sigma_s = \sqrt{5}$, and it has a considerable increase to 10% when $\sigma_s = 3$. On the other hand, when we fix the noise level at $\sigma_s = 0.5$, the lost track ratio has a slightly increase from

1% to 2% when the network dimension increases from $7m \times 7m$ to $35m \times 35m$. Fortunately, the true noise level in our experiments hovers around 2 in the outdoor fields, which means that the algorithm can still achieve good performance in practical cases.

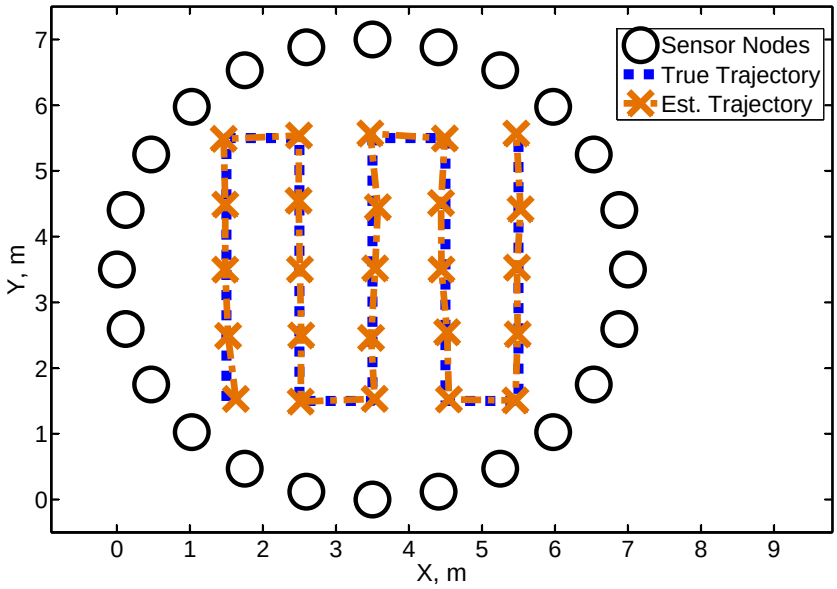
4.1.2 Various Layouts and Trajectories

In addition to the square trajectory in the square layout scenario that we showed before, several other scenarios were simulated under different measurement noise levels. Figure 4.6 and Figure 4.7 show three samples with different layouts and trajectories. Figure 4.6 (a) and (b) present a zigzag route within both square and circular network layouts, with a target speed of $0.5m/sec$. Figure 4.7 shows a scenario in which a target follows a square route in an irregular network deployment. This irregular layout we present here is generated from a standard $7m \times 7m$ square layout by slightly perturbing some of the nodes 1 or 2 metres from their original positions, with a purpose of mimicking the irregular deployment which would be necessary in a practical office or home. Therefore, we known exactly the ground-truth node location of the irregular sensor layout, and the layout is approximately $50m^2$. In this thesis, we only examine square and zigzag trajectories for these layouts. This is done because these trajectories are capable of representing most walking behaviors of human beings including both walks in a straight line and frequently turning, and because it is easier to obtain a ground-truth trajectory for structured (as opposed to random) paths in field experiments.

One of the initial purposes of designing the proposed pixel-free model was to address the irregular sensor layout, since the original pixelized model better suits a regularly shaped layout. In an arbitrary sensor layout, the pixelized model cannot divide the edges of the network area into perfect pixels, therefore it needs to sacrifice part of the estimation accuracy to track the target at the edges of the network area. The pixel-free model does not suffer from this problem. Figure 4.8 shows the performance for the zigzag trajectory in three different layouts: square, circular and irregularly shaped. In addition, we show the performance of the square trajectory in the square layout scenario for comparison. All four data sets are generated using the pixel-free model, and the network size is about $50m^2$ for all of the scenarios (to be precise, the network dimensions are $7m \times 7m$ for square layout, $7m$ diameter for circular layout, and approximately $50m^2$ for the irregular layout).



(a) Zigzag trajectory in a square node layout



(b) Zigzag trajectory in a circular node layout

Fig. 4.6 Simulation examples: tracking estimation using On-line SMC algorithm for (a) a zigzag trajectory in a $7m \times 7m$ square layout, (b) a zigzag trajectory in a circular layout with $7m$ diameter, with $\sigma_s = 2$. A target follows a ground-truth zigzag trajectory starts from point $(1.5, 1.5)$ and stops at point $(5.5, 5.5)$ with a speed of $0.5m/sec$.

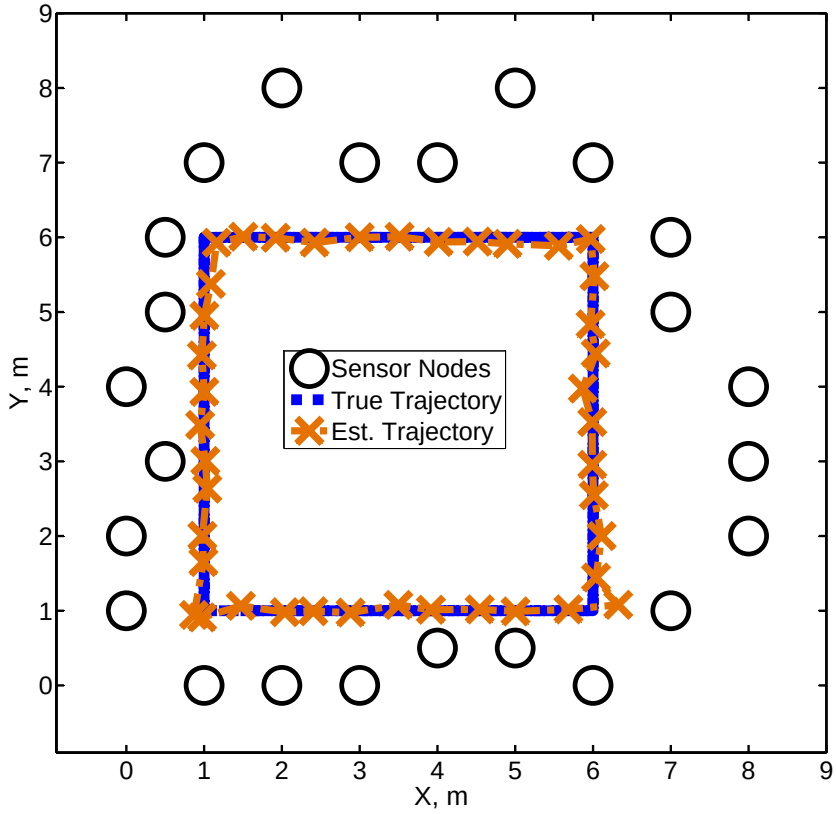


Fig. 4.7 Simulation example: tracking estimation using On-line SMC algorithm for a square trajectory in an irregular shape layout with $\sigma_s = 2$. The network covers an approximately $50m^2$ area. A target follows a ground-truth square trajectory starts from point $(1, 1)$ and stops at the same point after one cycle with a speed of $0.5m/sec$.

In Figure 4.8, we can clearly observe that the average RMSE value for the zigzag trajectory in the square layout is significantly higher than that for the square trajectory in the square layout. This is caused by the property of the transition model we use. An abrupt turn is a much less likely event in the AR-1 model employed by the on-line sequential Monte Carlo algorithm, so fewer particles are able to track the trajectory at the time step when the target changes direction at the corners. Therefore a higher estimation error occurs at the corner segments of the trajectory, which leads to a higher RMSE when doing zigzag trajectory estimation. Comparing the same zigzag trajectory in three different layouts, we find that the RMSE in the circle is the largest, followed by that in the irregularly shaped

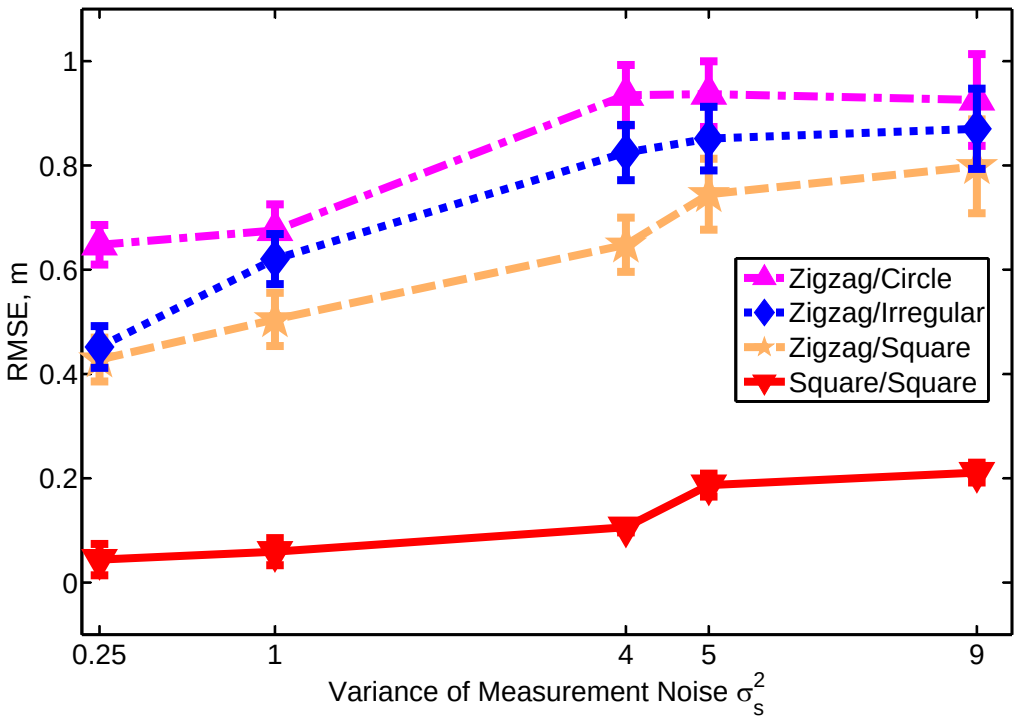


Fig. 4.8 RMSE of tracking estimation using On-line SMC algorithm for a zigzag trajectory in various (a square, a circle and a irregular shape) layouts of roughly $50m^2$ each and a square trajectory in a $7m \times 7m$ square layout, with varying noise level σ_s. The legend follows the form "Trajectory/Node Layout".

layout, with the RMSE in the square layout being the smallest. In general, all the RMSE values stay below 1 m in the $7m \times 7m$ network area. The lost track ratio follows the same trend over 100 realizations for each of the noise levels. With fixed $\sigma_s = 0.5$, the lost tracks ratio of square layout is 1%, while the ratio are 3% and 4% for circle layout and irregular layout respectively.

4.1.3 Tracking with Uncertain Node Locations

In practical applications, especially in emergency scenarios, it is often difficult to know the exact locations of most sensor nodes. For example, when deploying a sensor network around a building of size $100m \times 100m$, workers may drop sensors one by one along each side of the building in approximately the right position instead of measuring the accurate locations. In this case, sensors will not be positioned exactly as planned, although we can

assume they are deployed close to the planned location. The tracking system should be robust enough to perform reasonably well when given only the approximate locations of the sensor nodes. Alternatively, we may also be ignorant of the nodes locations, relying on them to employ a node localization algorithm that can autonomously determine the actual sensor locations to some imperfect degree of accuracy while simultaneously tracking the target, based on the real-time data collected by sensors. In this subsection, we will first test the robustness of our on-line sequential Monte Carlo algorithm to uncertainties in the node locations, afterwards, we will incorporate a node localization algorithm to see tracking performance in this scenario.

Tracking without Node Localization

To simulate the uncertain locations of the sensor network, we added Gaussian noise to the planned node locations when generating the simulated data. In our simulations, we employ independent distributions for the node locations, each being a two dimensional circularly-symmetric Gaussian with mean equal to the planned node location and standard deviation σ_p. This noise standard deviation was varied from 0.1 to 1 with all simulations being carried out in the $7m \times 7m$ square layout scenario. When tracking, the system used the planned node locations. 10 sets of 100 realizations were run (totally 1000 realizations) for each noise level in order to get an average lost track ratio.

Noise Std. Dev. σ_p	RMSE (m)	Lost Track Ratio
0.1	0.3511	0.5%
0.2	0.4933	0.6%
0.3	0.6263	0.9%
0.4	0.6967	0.9%
0.5	0.8722	1.4%
0.7	1.1530	3.2%
1	1.6897	52.6%

Table 4.3 RMSE and lost track ratio comparison for On-line SMC tracking, a person follows a square trajectory in a $7m \times 7m$ square layout with varying noise standard deviation σ_p added on planned node locations.

In Table 4.3, we show the RMSE and lost track ratio as a function of the standard

deviation of the noise added to all node locations. In practice, we sample noise from a Gaussian distribution with zero mean and a given noise standard deviation σ_p. According to Gaussian distribution, about 95 percent of the sampled noisy distances, which will be added to planned node locations, range between $-2\sigma_p$ and $2\sigma_p$. Our tracking system exhibits a reasonably good performance for target estimation when the prior noise σ_p stays below 0.5. Figure 4.9 shows one realization of the tracking result under with $\sigma_p = 0.5$.

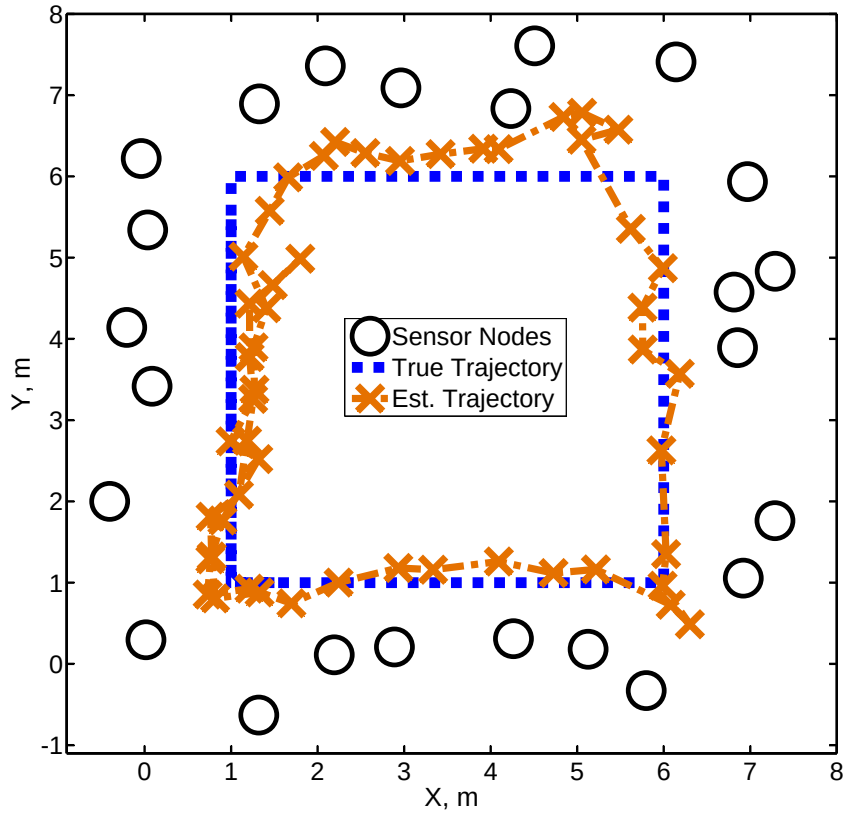


Fig. 4.9 Simulation example: tracking estimation using On-line SMC algorithm for a square trajectory in a $7m \times 7m$ square layout with a noise (standard deviation $\sigma_p = 0.5$) added onto node locations. A target follows a ground-truth square trajectory with a speed of $0.5m/sec$.

Although the accuracy of the tracking is worse than that with perfect node location, especially at the corners of the target trajectory, the tracking is still acceptable; the estimated trajectory generally follows the ground-truth trajectory. When the noise increases to 0.7, the lost track ratio doubles from that seen at $\sigma_p = 0.5$. We also see an increase of

the average RMSE to above 1 m. When σ_p rises to 1, the tracking algorithm in half of the realizations is broken. This means the proposed algorithm can not perform tracking when 95% of the nodes moves within 2 metres. We observed that in the successful realizations, tracking only tends to find the right track after 60 time steps (out of 120 time steps). Note that under the scenario with perfect location knowledge, most realizations find the right track at 20 to 30 time steps and the unknown parameter estimation converges at $50 \sim 60$ time steps.

To summarize, the proposed tracking algorithm is sufficiently robust to obtain acceptable estimation results when a small noise is added to the planned node locations, but the tracking system fails to track when a relatively high noise is added.

Tracking with Node Localization

In order to see how our algorithm might function in a case where node locations are not accurately recorded by a human but rather determined autonomously by a node localization algorithm, we incorporated the algorithm proposed in [25] with our on-line sequential Monte Carlo tracking algorithm. To test the tracking performance, we used our square trajectory in a square layout scenario, varying the dimensions of the square. The target moved at a speed of 0.5 m/s and one complete set of measurements (all 24 sensors) was recorded every 120 ms. The particle filter used 1000 particles and the unknown parameters were initialized by drawing from the previously mentioned uniform distributions; i.e., $\sigma_s \sim U(0, \sqrt{5}]$, $\phi \sim U(0, 10]$ and $\sigma_v \sim U(0, 1]$. Other parameters remained the same as in section 4.1.1.

The selected node localization algorithm requires that the location of a small number of nodes must be known before the tracking commences. These are the 'anchor nodes'. The locations of the rest of the nodes remain unknown. The four corner nodes (nodes 1, 7, 13, 19 in Figure 4.1) in our sensor network served as the anchor nodes. For the non-anchor sensor nodes, we also add a noise on their planned locations, the noise standard deviation σ_p varies depending on the dimension of the square layout. With increasing network dimension, the proportion of noisy distance decreases. Therefore we increase σ_p^2 from 1 to 5 when the network dimension increases from $7m \times 7m$ to $35m \times 35m$.

In Figure 4.10, we show the RMSE of the target tracking algorithm in the square layout for the square target trajectory example, averaged over 100 realizations with noise standard deviation $\sigma_p = 2$. This RMSE stays quite low (about 0.3 m) throughout most of the target's

route in the 28 m × 28 m square.

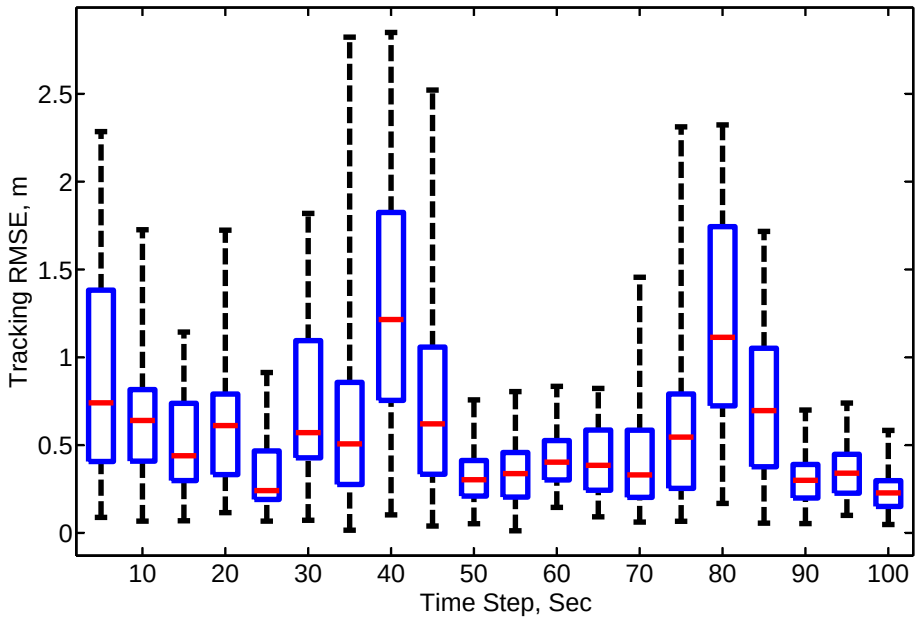


Fig. 4.10 RMSE of On-line SMC tracking estimation for a simulated target moving in a square trajectory within a 28 m × 28 m square with a noise (standard deviation $\sigma_p = 2$) added onto node locations. The boxes range from the 25th to 75th quantiles, the whiskers extend 3 times the interquartile range, the median is marked as a line within the box.

Using the aforementioned parameters for the measurement and dynamic models, we simulated different square layouts (ranging from 7 m × 7 m to 35 m × 35 m). The initial uncertainty in the node locations, governed by the value σ_p, was scaled in accordance with the area of the square. Tables 4.4 shows the RMSE of the target tracking in these different scenarios. As expected, the error increases slightly as the network size increases (i.e., as the density of the sensor nodes decreases). The accuracy is acceptable in all cases, however, with an average tracking RMSE of approximately 1m in the case of a 35 m × 35 m network.

In Table 4.5, we see how the sequential Monte Carlo tracking with integrated node-localization compares to tracking carried out with perfect a priori knowledge of the node locations. As expected, there is a clear performance penalty when the node locations are initially unknown: The RMSE values seen with unknown node locations are 6-7 times higher than those with known node locations. However, the estimation error experienced when node locations are unknown is still acceptable (with a median error of 1 m for a

Network size	σ_p	1st step RMSE	Final step Average	RMSE RMSE
7 m × 7 m	1	0.5828	0.0860	0.2830
14 m × 14 m	$\sqrt{2}$	1.0107	0.3175	0.5722
21 m × 21 m	$\sqrt{3}$	1.3734	0.3621	0.6774
28 m × 28 m	2	1.9778	0.4538	0.8472
35 m × 35 m	$\sqrt{5}$	2.7020	0.3980	1.0028

Table 4.4 RMSE (in m) of SMC tracking between the first and final step. The RMSE over time represents an average of the RMSEs over all 161 time steps.

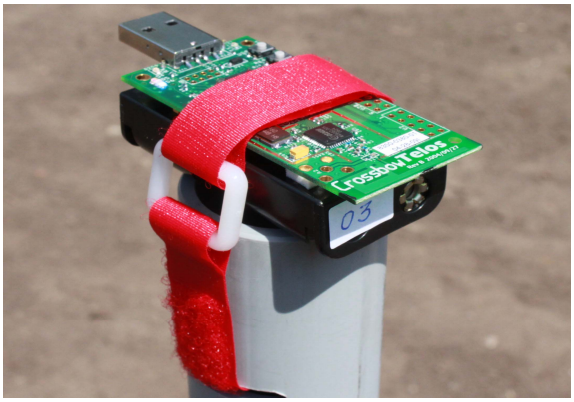
Network size	σ_p	RMSE with known node locations	RMSE with initially unknown node locations
7 m × 7 m	1	0.0436	0.2830
14 m × 14 m	$\sqrt{2}$	0.0728	0.5722
21 m × 21 m	$\sqrt{3}$	0.0975	0.6774
28 m × 28 m	2	0.1233	0.8472
35 m × 35 m	$\sqrt{5}$	0.1732	1.0028

Table 4.5 Comparison of the tracking RMSE (in m) averaged over all 161 time steps when node locations are known vs. when they are initially unknown.

relatively large network). Note that we refer here to the average RMSE over all time steps. The average error towards the end of the trajectory, when node location and model parameter estimates are more accurate, is significantly smaller (with a median error of 0.4 m).

4.2 Experiments

This section presents an evaluation of the proposed algorithm using measurements from a wireless sensor network test bed. We conducted a measurement campaign by collecting received signal strength measurements with a set of 24 sensor nodes. As shown in Figure 4.12(b), 24 sensor nodes were used to construct the sensor network and to measure received signal strength data. We used another node (node 25) which performed as a sink node (or master node) to collect the data packets. Node 25 was connected to a laptop through a USB port in order to send the data back and to receive orders. The laptop was responsible for processing the received data packets. The master node was placed within the transmission range of the other sensors in order to collect their data packets and convey them to the laptop.



(a) Sensor node

(b) Setting up an experiment in Field 2

(c) Grassy Field 1

Fig. 4.11 Experiment photos taken in McGill campus outdoor fields.

In the experiment, each sensor was equipped with a plastic stand as shown in Figure 4.12(a). All the nodes were Crossbow TelosB motes running TinyOS and using the IEEE 802.15.4 standard for communication in the 2.4 GHz frequency band. A simple token-ring transmission protocol was developed using nesC and each node was assigned a fixed node ID at compile time. Data packets broadcast by each node contain this node ID along with the time of transmission and the measured inter-node received signal strength values which were received by that node from other nodes in the network. Since the nodes transmit in a strictly increasing order. Based on node ID, whenever a node received a broadcast from another node, it checked the ID of the sender to see whether it was slated to broadcast during the next transmission slot. The interval between each transmission was set to 20 ms. Since we had 25 sensor nodes in the network, so it took about 0.5 second to collect a complete data set from all nodes. Therefore, 50 samples (or 2 complete sets) were recorded every second.

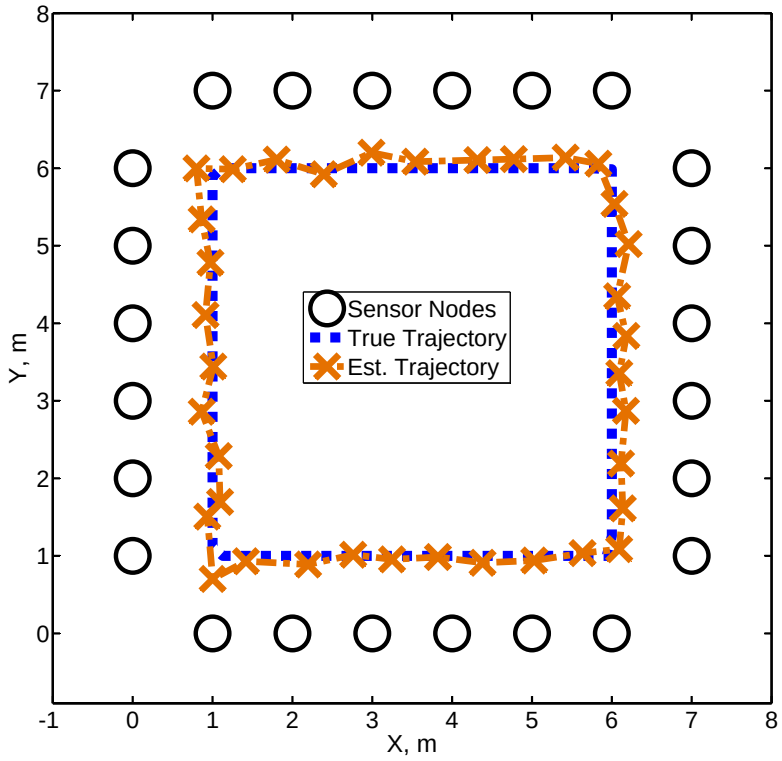


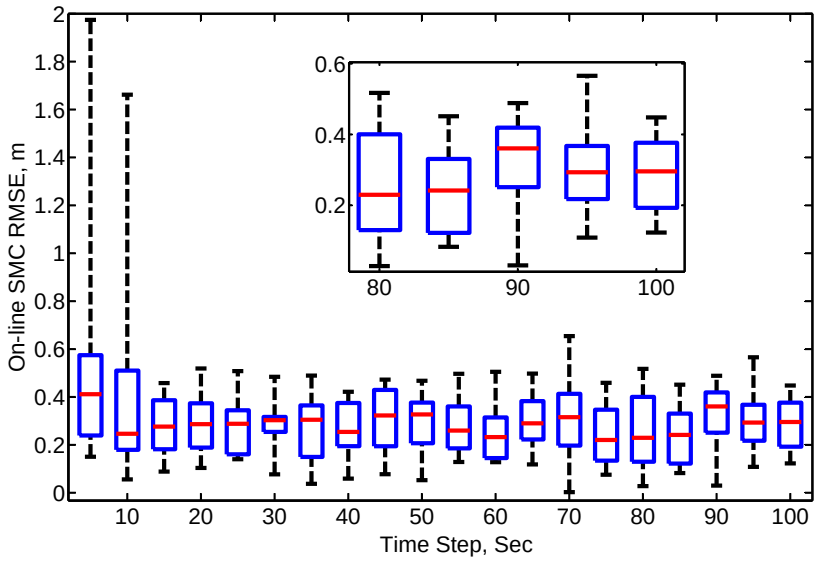
Fig. 4.12 Experimental example: tracking estimation using On-line SMC algorithm for a square trajectory in a 7 m $\times$ 7 m square layout in Field 1. A target follows a marked square trajectory with a speed of $0.5m/sec$.

The sensor network itself was set up in an outdoor field with a tree in the center of it (Field 1) as shown in Figure 4.12(c). The sensors were all placed on stands so that they were 1 m off the ground as shown in Figure 4.12(a), and these stands were placed in a 7 m $\times$ 7 m square, mimicking the network we had heretofore been simulating. Markers were placed at 20 positions within the square so that the person walking through the network would be aware of—and be able to follow—the predetermined ground-truth path. Before a person was brought into the network, however, the system sensed the vacant network area for roughly 3 minutes in order to generate the average received signal strength vector $\boldsymbol{\gamma}_{\text{avg}}$. After this data had been collected, a person walked through the network following the ground-truth path while the sensors collected more received signal strength measurements.

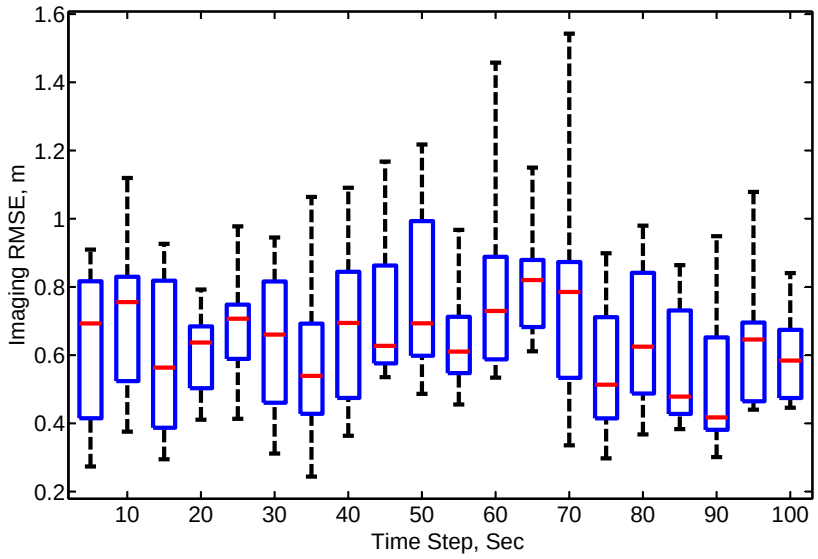
Algorithm	RMSE for Field 1 (m)	RMSE for Field 2 (m)
SMC	0.4905	0.3214
Imaging with KF	0.8566	0.6404

Table 4.6 RMSE values for different algorithms run on the experimental results obtained in a 7 m $\times$ 7 m square area with a tree in the center.

Figure 4.12 shows the experimental result of tracking a target walked in a $7m \times 7m$ square sensor layout in Field 1. The target started from a calibrated point (1, 1) and stopped at the same point after one cycle, following a marked square trajectory with a speed of $0.5m/sec$. 25 identical experiments were repeated. We compare the performance of the proposed on-line sequential Monte Carlo algorithm to the imagining plus Kalman filter approach described in [84]. The RMSE of both algorithms are shown in Figure 4.13. The RMSE of both algorithms remains stable during the experiments. As shown in Figure 4.13(a), the tracking RMSE stays at about 0.3 m. Meanwhile, using the imaging with KF algorithm, the tracking RMSE in Figure 4.13(b) is about 0.6 m under the same condition.



(a) Target tracking RMSE using the On-line SMC algorithm.



(b) Target tracking RMSE using the imaging with KF algorithm.

Fig. 4.13 Box-and-whisker plot of RMSE as a function of time for the tracking of a real target moving in a square trajectory within a 7 m $\times$ 7 m square in Field 1 using both (a) the On-line SMC algorithm and (b) the imaging with Kalman filter algorithm. The boxes range from the 25th to 75th quantiles, the whiskers extend 3 times the interquartile range, the median is marked as a line within the box.

We also carried out a similar tracking experiment for the same scenario in another field which had no tree in it (Field 2), again having a target walk 25 times following a marked square trajectory with a speed of $0.5m/sec$. A numerical comparison of the tracking performance between Field 1 and Field 2 can be seen in Table 4.6. We observe that both algorithms perform better when there are no additional obstructions besides the one we are tracking and that on-line sequential Monte Carlo outperforms imaging with KF in both scenarios.

Parameter	Field 1 (value with st.d.)	Field 2 (value with st.d.)
ϕ	4.0394 $\pm$ 0.0476	6.1413 $\pm$ 0.0985
σ_v	0.3588 $\pm$ 0.0524	0.2021 $\pm$ 0.0058
σ_w	2.3109 $\pm$ 0.0393	2.0029 $\pm$ 0.0092

Table 4.7 Estimated unknown parameters with their standard deviation over 25 identical experiments, obtained in a 7 m $\times$ 7 m square in both Field 1 and Field 2 .

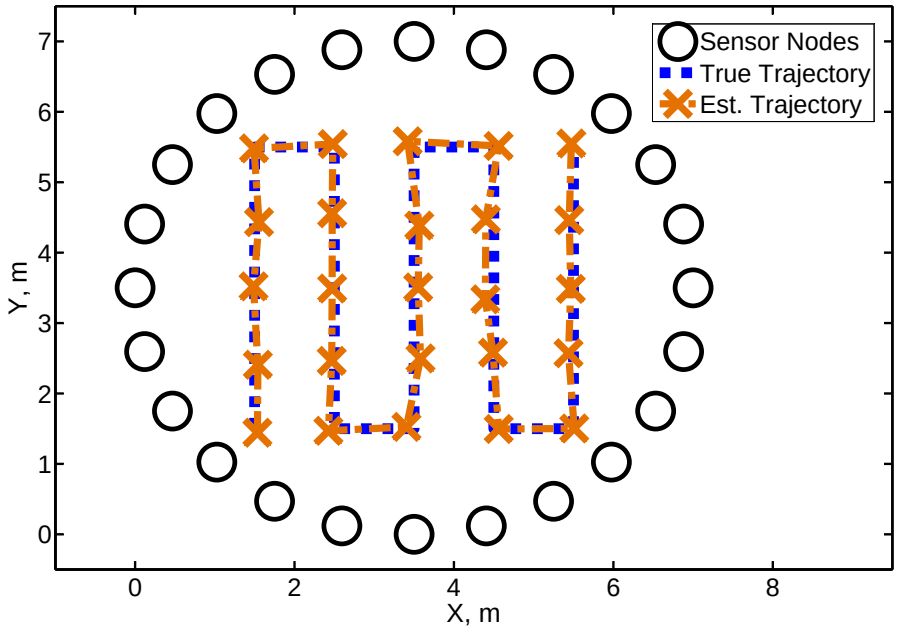


Fig. 4.14 Experimental example of target tracking for a zigzag trajectory in a 7 m diameter circle.

In Table 4.7, we shows estimated values of unknown parameter set θ in both Field

1 and Field 2. Standard deviation values of the estimated parameters are also reported over 25 identical experiments. We can see that the average attenuation level parameter of the pixel-free model ϕ takes a higher value when there is a tree within the network area. This makes sense as the presence of this obstruction leads to smaller changes in the received signal strength when a person moves across a link. We also observe that with an approximate walking speed of $0.5m/sec$, the average transition noise standard deviation σ_v is 0.2 in Field 2 and 0.35 in Field 1. The average measurement noise σ_w is approximately 2.

In addition to the square trajectory scenario, we also ran an experiment where the target moved along a zigzag trajectory with the sensors arranged in a circular pattern with a 7 m diameter, as shown in Figure 4.14. In this case, the average tracking RMSE is 0.2112 m when using on-line sequential Monte Carlo and 0.4670 m when using the imaging with Kalman filtering method. Both Figure 4.12 and Figure 4.14 demonstrate that the proposed on-line sequential Monte Carlo algorithm using the pixel-free model achieves good trajectory estimates in different scenarios with experimental data and that this algorithm is capable of handling different sensor deployments.

Chapter 5

Conclusion

5.1 Summary and Discussion

In this thesis, we have reviewed and analyzed tracking techniques that can be used to perform Radio-Frequency tomographic tracking in wireless sensor networks. The work focused on the design of an efficient on-line tracking algorithm for device-free passive tracking systems with the aim of achieving high estimation accuracy with low computational cost in practical deployments.

In Chapter 2, we first described wireless sensor networks, networks that consists of numerous wireless sensor nodes used to monitor environmental conditions. We then introduced the different architectures in which a wireless sensor network can be deployed, i.e., centralized, distributed and hierarchical architectures. (Recall that in our field experiments, we built a wireless sensor network using a centralized architecture, in which a 'master' node was responsible for collecting data from other nodes and broadcasting control messages.) We also described some application scenarios for wireless networks, including battlefield surveillance, medical care in hospitals and habitat monitoring. Afterwards, we emphasized a specific hardware platform—Crossbow TelosB sensor nodes–which we used in our real experiments.

We then proceeded to an introduction of tracking systems. These were classified as device-based systems and device-free systems depending on whether a tracked object needed to carry devices or not. Recall that device-based systems use various communication techniques to detect and track targets including infrared, ultrasound and Radio-Frequency techniques. All of these methods require a target to carry electronic devices that can co-

operate with the systems, such as an RFID tag, a cell phone or a mobile sensor node. Device-based systems provide relatively high estimation accuracy since they can adjust the estimation based on reference information from the attached devices. However, their initial deployment is time-consuming since all the components of a device-based system need to be pre-installed at fixed locations. In addition, device-based tracking fails in some emergency applications such as earthquake-survivor rescue and in privacy-protection-oriented applications such as medical monitoring for elderly people. Device-free systems have the advantage in these applications. Some of these systems require a training phase before tracking, during which time they record and store signal patterns that are captured by the system when a target is present in different positions within the sensed area. They then perform tracking by comparing real-time signals with the stored data and estimating target locations by determining which of those stored signals share similar features with the current one. Although these systems are device-free, it takes time to train fingerprinting systems and the stored data needs to be calibrated again when the environment changes. Other systems track targets in a signature-free manner so that no training phases are required during the deployment period. The RF tomographic tracking systems which we explored in this thesis belong to this category.

The second half of Chapter 2 focused on the basic sequential Monte Carlo (SMC) algorithms (particle filtering) and static parameter estimation methods that we adopted in our approach. Particle filtering refers to several different simulation-based approaches that can be used to estimate a hidden state in a Bayesian model (e.g., the location of an object which is being tracked) given observations of that state. It outperforms other approaches such as the Kalman filter when the problem is highly nonlinear/non-Gaussian. We then presented the sequential importance sampling (SIS) particle filter and introduced the degeneracy problem from which it suffers. A resampling procedure was proposed to address this problem, creating the sampling importance resampling (SIR) method. Eventually, we introduced an auxiliary particle filter (APF) based on the sampling importance resampling method. The auxiliary particle filter performs better in environments with small process noise, which was suitable for our tracking scenario. Afterwards, we introduced static parameter estimation which can be used to estimate the unknown model parameters in particle filtering. Both Bayesian and maximum likelihood approaches were introduced, and we focused on the on-line expectation maximization (EM) algorithm, which belongs to the family of maximum likelihood approaches. Among the different static parameter

estimation approaches, on-line EM was more stable and computationally efficient for a high-dimensional unknown model parameter set. The on-line EM also suffered from a degeneracy problem, so a pseudo-likelihood function was introduced and incorporated into on-line EM to address this problem and to improve computational efficiency.

Chapter 3 first formulated the device-free tracking problem we were going to solve and presented an introduction to our measurement models. Since the previously-proposed pixelized model suffers from a quantizing problem whereby it sacrifices part of estimation accuracy, we proposed a novel pixel-free model that succeeded in addressing this problem by establishing a scale relationship that connects attenuation level and distance between a target and a link. The rest of this chapter described an on-line sequential Monte Carlo algorithm that uses an auxiliary particle filter to approximate the target's location while simultaneously estimating unknown parameters. During the tracking period, the particle filter estimates the target's location at every time step while on-line EM is executed every L time steps. Unknown parameter estimation is based on both observation data and on the estimated location given by particle filter; new updated parameters are then used in the next particle filtering iteration to iteratively improve the tracking performance.

We validated our algorithm in Chapter 4 through both simulation and field experiments. In simulation, we first introduced a square trajectory in a square sensor layout scenario, which we used as a "basic example" throughout our simulations and experiments. The root mean square error (RMSE) of the location estimation, the lost track ratio, and the unknown parameter behavior were reported for more than 100 repeated realizations. We than tested performance of both the on-line sequential Monte Carlo algorithm and the previously proposed imaging plus Kalman filter algorithm by varying measurement noise levels and sensor network dimensions. Under the same "basic example" scenario, our approach achieved a much better RMSE performance than that of the imaging algorithm. For example, when the noise level was set at $\sigma_s = 2$, the RMSE of our algorithm was $0.0988m$ while the RMSE of the imaging algorithm was $1.1451m$. We observed that the RMSE increased when either the noise level or the network dimensions increase. We also evaluated the performance of our algorithm under different combinations of target trajectories and network layouts. The estimated distance error was significantly higher in the zigzag trajectory compared to the square trajectory, e.g., over 100 realizations for each trajectory, the average zigzag RMSE was $0.8m$ while the average square trajectory RMSE was $0.12m$ under a $7m \times 7m$ square layout. Finally, we presented tracking performance in

scenarios where the node locations were uncertain. Under the same basic example setting, the RMSE the and number of lost tracks increased rapidly when σ_p was larger than 0.7 (where σ_p is the standard deviation of the noise added to the node locations). Over 1000 realizations, the RMSE was $0.6263m$ and the lost track ratio was 0.9% when $\sigma_p = 0.3$, while the RMSE was $1.6897m$ and the lost track ratio was 52.6% when $\sigma_p = 1$. As a comparison, we also reported tracking performance under the same setting when our algorithm was incorporated with a node localization algorithm. The sequential Monte Carlo approach then achieved a reasonable RMSE performance under different settings. For example, with a network dimension of $21m \times 21m$ in a square sensor layout and with $\sigma_p = \sqrt{3}$, the RMSE of our particle filtering approach with node localization algorithm was $0.6774m$ over 100 realizations. The second part of Chapter 4 focused on outdoor field experiments we did in the McGill lower campus. We compared the performance for both the Field 1 and the Field 2 experimental scenarios over 25 experiments. The on-line sequential Monte Carlo algorithm outperformed the imaging plus Kalman filter algorithm in both outdoor settings for single target tracking. Based on data collected from an experiment in Field 2 which followed the "basic example" scenario, our approach achieved a RMSE of $0.3214m$ while the RMSE of the imaging approach was $0.6404m$. Both simulation and experimental results demonstrated that our on-line sequential Monte Carlo approach achieves reasonably good performance under a single target tracking scenario.

5.2 Future Work

Although we achieved expected good tracking performance using the proposed on-line sequential Monte Carlo algorithm, we only focused on single target tracking in an outdoor environment. Practical implementation may require a system to track multiple targets in a complex scenario or inside a building (through-wall tracking). Therefore, three main issues need to be addressed in our research: testing the tracking performance when other static objects (such as rocks, metal pieces, woods) are placed within the sensor network, establishing an indoor/through-wall measurement model and realizing multiple targets tracking.

The experiments with different static objects will firstly be done in outdoor fields, objects will be placed on a link between a single pair of nodes to examine their effect to the link. Afterwards, objects will be placed within a sensor network with 24 nodes; a person will follow markers in the sensor network to check the behaviors of each link. We will

also repeat the experiments in indoor areas to obtain features of RF signals in complex scenarios.

Indoor/through-wall environments exhibit a much more complex Radio-Frequency signal pattern. Compared with outdoor environments, multipath effects in indoor scenarios are significantly higher, while obstructions such as walls, tables and computers will greatly increase the background noise level. Thus the proposed pixel-free measurement model needs to be modified. We plan to first explore features of a single link in indoor environments. Different types of obstructions, such as a wooden box and a metal plate will be placed on the link or nearby the link while a person stands in different positions to test how obstructions interact with the presence of a person in indoor environments. Afterwards we will examine multiple links in at least 24 nodes deployment and try to obtain a new, computationally efficient indoor measurement model.

There are two difficulties of multiple target tracking: the first is to determine the number of moving targets within the sensed area and the second is to distinguish targets using a single data set. Targets may happen to stand close to each other, stand still at one position, or leave the network sensing area, so it is a challenging task for a tracking system to detect these movements and track them. The computational cost may also increase rapidly with an increasing number of moving targets.

To reiterate, we will examine human spatial characteristics in indoor wireless sensor networks to achieve higher tracking accuracy and robustness, and we will also augment our tracking algorithm so that it can distinguish different targets. We will also further examine the Radio-Frequency signals transmitted by the TelosB sensor nodes as well as the features of the adaptive environmental parameters, and we will work on improving the sensing mechanisms of wireless sensor networks in field experiments in order to make our system more robust in different environments.

Appendix A

A.1 Software Description

The on-line SMC tracking system operates in Matlab and consists of 7 Matlab files. We now briefly describe each function and present in Figure A.1, a flowchart depicting their interactions. The source code is provided in Section A.2.

- ***Main.m***, is the m-file that execute the complete algorithm, it calls function $getTrueTrajectory$ to generate target trajectory and function $simData$ to generate RSS data. Afterwards it calls function $particleFilter$ to track target position and function $onlineEM$ to estimate unknown model parameters.
- ***getTrueTrajectory.m***, is the function to generate simulated target trajectory.
- ***simData.m***, is the function to generate simulated RSS values.
- ***particleFilter.m***, is the function to track target locations using RSS values, it also draws the tracked trajectory every loop. It also calls function $multinomialR$ to do resampling.
- ***onlineEM.m***, is the function to estimate unknown model parameters, it executes every L particle filtering runs.
- ***multinomialR.m***, is the function to finish resampling procedure in particle filtering.
- ***L2distance.m***, is the function to calculate link distance between two nodes.

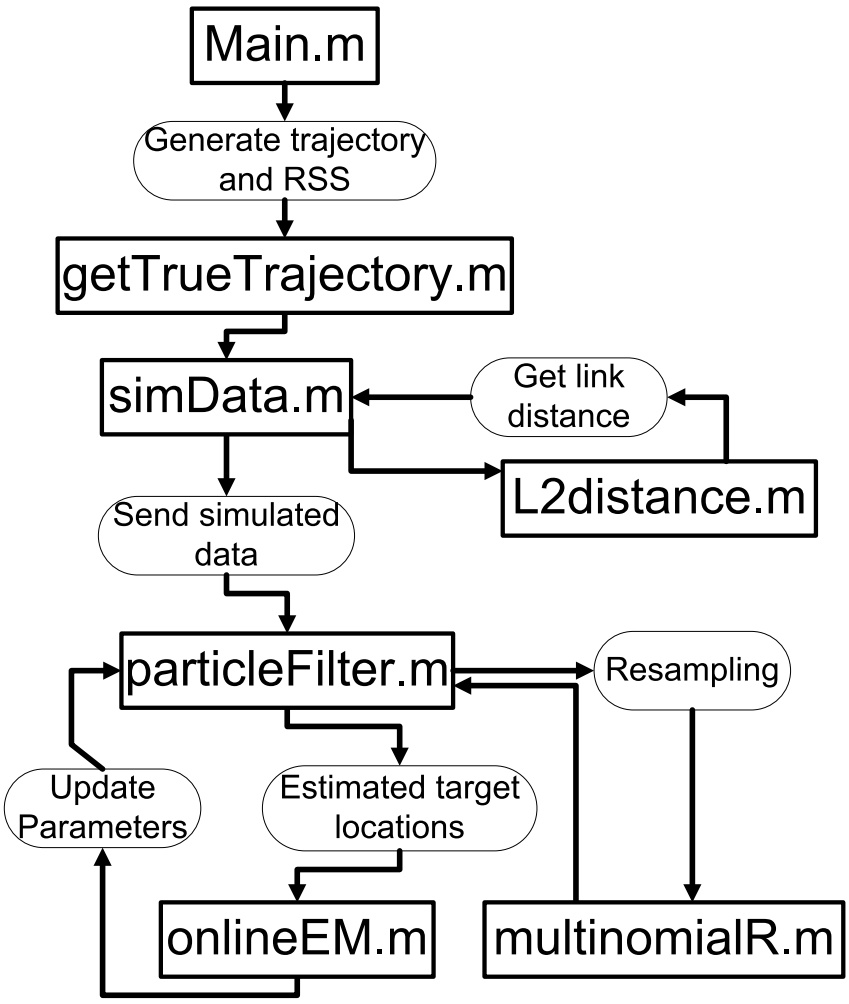


Fig. A.1 Matlab simulation software flowchart

A.2 Matlab Source Code

A.2.1 Main.m

```
clc
clear all;
close all;

%% Basic setting
```

```
numNodes = 24;      %Number of nodes
linkNum = numNodes * (numNodes-1)/2;  %Number of links
sideLength = 7;   %Length of the square is 21 feet.
deltaP = 0.1;      %Pixel width of image
numRealization = 1;   %Number of identical simulations
stateDimension = 2;    %State dimension for dynamic model
sigma = 0.02; % \sigma_lambda to control the curve

% Set ground-truth parameters' values, generate RSS
phi_true = 5;
sigma_v_true = 0.3;
sigma_w_true = 0.5;

T_interval = 0.24;    %Simulated time interval per sensing loop
blockLength = T_interval * 50; %Data block to process in a run

% Circle layout, link order is node 1 to node 2,node 1 to 3,...
% node 1 to node 24, node 2 to node 3,...
actualLocs(1,:) = sideLength / 2 *(1 + ...
                    cos(2*pi/numNodes*[0:numNodes-1]));
actualLocs(2,:) = sideLength / 2 *(1 + ...
                   sin(2*pi/numNodes*[0:numNodes-1]));
temp = actualLocs(:,19:24);
actualLocs(:,7:24) = actualLocs(:,1:18);
actualLocs(:,1:6) = temp;

% Store the sensor locations correspond to this link
linkLocs       = zeros(4,linkNum);
linksDistances = zeros(linkNum,1);

linkIndex = 1;
for i = 1:numNodes
   for j = i+1:numNodes
```

```
        linkLocs(1:2,linkIndex) = actualLocs(:,i);
        linkLocs(3:4,linkIndex) = actualLocs(:,j);
        linksDistances(linkIndex,:) = ...
            L2distance(actualLocs(:,i),actualLocs(:,j));
        linkIndex = linkIndex + 1;
    end
end

% Generate the zigzag trajectory
targetTruePos = getTrueTrajectory(sideLength, numNodes);

% Generate the attenuated RSS data
attenuationCausedByMoving = phi_true * simData(...
    targetTruePos, linkLocs, linksDistances, sigma);

% Store particles for later use, store ground-truth locations
estimatedPositions = zeros(2,size(targetTruePos,2),numRealization);
truePositions = zeros(size(targetTruePos,1),...
    size(targetTruePos,2),numRealization);
for i = 1:numRealization
    truePositions(:,:,i) = targetTruePos;
end

%% Main algorithm
for realizationForOneTrack = 1:numRealization

    % Initialize the unknown parameters
    phi = normrnd(0,5);    % \phi in measurement model
    sigma_v = normrnd(0,1); % noise \sigma_v in dynamic model
    sigma_w = normrnd(0,1);% noise \sigma_s in measurement model

    deltaYMat(:,:) = attenuationCausedByMoving +...
        sigma_w_true * randn(size(attenuationCausedByMoving));
```

```
counter = 0;
N = 1000;  % Number of particles
L = 10;    % Length of one online EM run

% Algorithm parameter setting
psiVec = zeros(1,4);  % Store the unknown parameters
deltaYBuffer = zeros(L, size(deltaYMat,1));
xParticlesInOneEMRun = zeros(stateDimension,N,L);
xParticlesPreviousRound = zeros(stateDimension,N);
xParticlesPreviousRound(1:2,:) = sideLength * rand(2,N);
weightsCurrent = ones(1,N);
diffVec = [];
mseResultVec = [];
pathDistanceFromSide = 1;
targetPosVec = zeros(2,22);
phiVec = [phi];
sigmavVec = [sigma_v];
sigmawVec = [sigma_w];
phiLength = floor(size(deltaYMat,2) / L);


while true
    counter = counter + 1;

    if counter > size(deltaYMat,2)
        break
    end

    % Particle filtering
    deltaY = deltaYMat(:,counter);

    [xParticlesThisRound, weightsCurrent] = ...
```

```
        particleFilter(xParticlesPreviousRound,deltaY,...
        phi,sigma_v,sigma_w,N,sideLength,...
        weightsCurrent,linkLocs,linksDistances,sigma);

    % Get the estimated target position
    targetPos = mean(xParticlesThisRound,2);
    estimatedPositions(:,counter,realizationForOneTrack)...
        = targetPos(:);

    %Buffer the number of particle filtering runs
    indexWithinOneEMRun = mod(counter,L);

    %Online EM when buffer L is full
    if indexWithinOneEMRun
        deltaYBuffer(indexWithinOneEMRun,:) = deltaY;
        xParticlesInOneEMRun(1:2,:,indexWithinOneEMRun)...
            = xParticlesThisRound;
    else
        deltaYBuffer(L,:) = deltaY;
        xParticlesInOneEMRun(1:2,:,L) = xParticlesThisRound;

        %Compute the sufficient statistics,
        %update the unknown model parameter.
        [phi, sigma_v, sigma_w, psiVec, phiVec,sigmavVec, ...
            sigmawVec] = onlineEM(phi,...
            psiVec, xParticlesInOneEMRun, deltaYBuffer,...
            linkLocs, linksDistances, counter, ...
            phiVec, sigmavVec, sigmawVec, sigma);

    end
    %Update the particles
    xParticlesPreviousRound = xParticlesThisRound;
end
```

```
    phis(:,realizationForOneTrack) = phiVec(:);
    sigmavs(:,realizationForOneTrack) = sigmavVec;
    sigmaws(:,realizationForOneTrack) = sigmawVec;
end

    %% Save the tracked results

    save('estimatedPositions.mat', 'estimatedPositions');
    save('truePositions.mat', 'truePositions');
    save('phis.mat', 'phis');
    save('sigmavs.mat', 'sigmavs');
    save('sigmaws.mat', 'sigmaws');
```

A.2.2 getTrueTrajectory.m

```
%% Generate ground-truth zigzag trajectory
function targetTruePos = getTrueTrajectory(sideLength,numNodes)

sepa = sideLength/(numNodes/4+1);
targetTruePos = ...
        [1.5*sepa 1.5*sepa; 1.5*sepa 2.5*sepa; 1.5*sepa 3.5*sepa;
         1.5*sepa 4.5*sepa; 1.5*sepa 5.5*sepa;
         2.5*sepa 5.5*sepa; 2.5*sepa 4.5*sepa; 2.5*sepa 3.5*sepa;
         2.5*sepa 2.5*sepa; 2.5*sepa 1.5*sepa;
         3.5*sepa 1.5*sepa; 3.5*sepa 2.5*sepa; 3.5*sepa 3.5*sepa;
         3.5*sepa 4.5*sepa; 3.5*sepa 5.5*sepa;
         4.5*sepa 5.5*sepa; 4.5*sepa 4.5*sepa; 4.5*sepa 3.5*sepa;
         4.5*sepa 2.5*sepa; 4.5*sepa 1.5*sepa;
         5.5*sepa 1.5*sepa; 5.5*sepa 2.5*sepa; 5.5*sepa 3.5*sepa;
         5.5*sepa 4.5*sepa; 5.5*sepa 5.5*sepa;
         4.5*sepa 5.5*sepa; 4.5*sepa 4.5*sepa; 4.5*sepa 3.5*sepa;
         4.5*sepa 2.5*sepa; 4.5*sepa 1.5*sepa;
         3.5*sepa 1.5*sepa; 3.5*sepa 2.5*sepa; 3.5*sepa 3.5*sepa;
```

```
        3.5*sepa 4.5*sepa; 3.5*sepa 5.5*sepa;
        2.5*sepa 5.5*sepa; 2.5*sepa 4.5*sepa; 2.5*sepa 3.5*sepa;
        2.5*sepa 2.5*sepa; 2.5*sepa 1.5*sepa]';
end
```

A.2.3 simData.m

```
%% Generate the attenuated RSS data
function YParticlesWithoutPhi = simData(...
    xParticles, linksLocations, linksDistance, sigma)
    % Calculate the link distance
    distanceBetweenParticlesToA = L2distance(...
        linksLocations(1:2,:), xParticles);
    distanceBetweenParticlesToB = L2distance(...
        linksLocations(3:4,:), xParticles);
    lambdaVector = distanceBetweenParticlesToA +...
        distanceBetweenParticlesToB - linksDistance *...
        ones(1,size(xParticles,2));
    % Generate RSS data through measurement model
    YParticlesWithoutPhi = exp(-lambdaVector / (2*sigma));
end
```

A.2.4 particleFilter.m

```
%% Auxiliary particle filter
function [xAPFParticles, weightsCurrent] = particleFilter(...
    xParticlesPreviousRound, deltaY, phi, sigma_v, sigma_w, N,...
    sideLength,weightsPrevious,linksLocations,linksDistances,sigma)
%%
%Target motion dynamics
xParticlesThisRound = xParticlesPreviousRound + sigma_v *...
    randn(size(xParticlesPreviousRound));
%Generate approximat measurement \y
yParticles = phi * simData(xParticlesThisRound,...
    linksLocations,linksDistances,sigma);
```

```
%Calculate weights
weightings = normpdf(yParticles, deltaY*ones(1,N), sigma_w) + 1e-8;
logWeights= sum(log(weightings));
logWeights = logWeights - max(logWeights);
weightsCurrent = weightsPrevious .* exp(logWeights);
%Normalized the weights
weightsCurrent = weightsCurrent/sum(weightsCurrent);
%Multinomial Resampling Process
outIndex = multinomialR(1:N, weightsCurrent');

%% Auxiliary part
xAPFParticles(:,:) = xParticlesThisRound(:,outIndex);
yNewAPFParticles(:,:) = yParticles(:,outIndex);

yAPFParticles = phi * simData(xAPFParticles,...
    linksLocations,linksDistances,sigma);

% Calculate the final weights
valueMat = normpdf(yAPFParticles, deltaY * ones(1,N), sigma_w);
logValueVec = sum(log(valueMat));
maxLogValueVec = max(logValueVec);
logValueVec = logValueVec - maxLogValueVec;
valueMatNew = normpdf(yNewAPFParticles, deltaY * ones(1,N), sigma_w);
logValueVecNew = sum(log(valueMatNew));
maxLogValueVecNew = max(logValueVecNew);
logValueVecNew = logValueVecNew - maxLogValueVecNew;
likelihood = exp(logValueVec);
likelihoodNew = exp(logValueVecNew);
weightsCurrent = likelihood ./ likelihoodNew;
%Normalize the weights
weightsCurrent = weightsCurrent / sum(weightsCurrent);

%% Draw the dynamic tracking results
```

```
numPixelRow = 100;
numPixelCol = 100;
xShow = zeros(numPixelRow, numPixelCol);

for i = 1:N
    xPos = round(xAPFFParticles(1,i)/sideLength * numPixelCol);
    yPos = round(xAPFFParticles(2,i)/sideLength * numPixelRow);
    if xPos > numPixelCol
        xPos = numPixelCol;
    end
    if xPos < 1
        xPos = 1;
    end
    if yPos > numPixelRow
        yPos = numPixelCol;
    end
    if yPos < 1
        yPos = 1;
    end
    xShow(numPixelRow + 1 - yPos,xPos) = xShow(...
        numPixelRow + 1 - yPos,xPos) + 1;
end

figure(11)
movegui('northwest');
colormap cool;
set(gca,'units','points');
imagesc(xShow);
colorbar;
```

A.2.5 multinomialR.m

```
function outIndex = multinomialR(inIndex,q)
% PURPOSE : Performs the resampling stage of the SIR
```

```
%            in order(number of samples) steps.
% INPUTS  : - inIndex = Input particle indices.
%           - q = Normalised importance ratios.
% OUTPUTS : - outIndex = Resampled indices.
% AUTHORS  : Arnaud Doucet and Nando de Freitas
% DATE     : 08-09-98

if nargin < 2, error('Not enough input arguments.'); end

[S,arb] = size(q);  % S = Number of particles.

% MULTINOMIAL SAMPLING:
% =====================
N_babies= zeros(1,S);
cumDist= cumsum(q');
% generate S ordered random variables uniformly distributed in [0,1]
% high speed Niclas Bergman Procedure
u = fliplr(cumprod(rand(1,S).^(1./(S:-1:1))));
j=1;
for i=1:S
  while (u(1,i)>cumDist(1,j))
    j=j+1;
  end
  N_babies(1,j)=N_babies(1,j)+1;
end;

% COPY RESAMPLED TRAJECTORIES:
% ============================
index=1;
for i=1:S
  if (N_babies(1,i)>0)
    for j=index:index+N_babies(1,i)-1
      outIndex(j) = inIndex(i);
```

```
    end;
  end;
  index= index+N_babies(1,i);
end
```

A.2.6 L2distance.m

```
function d = L2distance(a,b,df)
% L2_DISTANCE - computes Euclidean distance matrix
%
% E = L2_distance(A,B)
%
%    A - (DxM) matrix
%    B - (DxN) matrix
%    df = 1, force diagonals to be zero; 0 (default), do not force
%
% Returns:
%    E - (MxN) Euclidean distances between vectors in A and B
%
%
% Description :
%    This fully vectorized (VERY FAST!) m-file computes the
%    Euclidean distance between two vectors by:
%
%                ||A-B|| = sqrt ( ||A||^2 + ||B||^2 - 2*A.B )
%
% Example :
%    A = rand(400,100); B = rand(400,200);
%    d = distance(A,B);

% Author    : Roland Bunschoten
%             University of Amsterdam
%             Intelligent Autonomous Systems (IAS) group
%             Kruislaan 403  1098 SJ Amsterdam
```

```
%             tel.(+31)20-5257524
%             bunschot@wins.uva.nl
% Last Rev : Wed Oct 20 08:58:08 MET DST 1999
% Tested   : PC Matlab v5.2 and Solaris Matlab v5.3

% Copyright notice: You are free to modify, extend and distribute
%    this code granted that the author of the original code is
%    mentioned as the original author of the code.

% Fixed by JBT (3/18/00) to work for 1-dimensional vectors
% and to warn for imaginary numbers.  Also ensures that
% output is all real, and allows the option of forcing diagonals to
% be zero.

if (nargin < 2)
   error('Not enough input arguments');
end

if (nargin < 3)
   df = 0;     % by default, do not force 0 on the diagonal
end

if (size(a,1) ~= size(b,1))
   error('A and B should be of same dimensionality');
end

if ~(isreal(a)*isreal(b))
   disp('Warning: Results may be off.');
end

if (size(a,1) == 1)
  a = [a; zeros(1,size(a,2))];
  b = [b; zeros(1,size(b,2))];
```

```
end

aa=sum(a.*a); bb=sum(b.*b); ab=a'*b;
d = sqrt(repmat(aa',[1 size(bb,2)]) +...
    repmat(bb,[size(aa,2) 1]) - 2*ab);

% make sure result is all real
d = real(d);

% force 0 on the diagonal?
if (df==1)
  d = d.*(1-eye(size(d)));
end
```

A.2.7 onlineEM.m

```
function [phi, sigma_v, sigma_w, psiVec, phiNewVec,...
    sigmavNewVec, sigmawNewVec] = onlineEM(...
    phi, psiVec, xBuffer, deltaYBuffer, linksLocations,...
    linksDistances, counter, phiVec, sigmavVec,sigmawVec,sigma)
%Online EM to calculate the sufficient statistics and update
%the unknown model parameters

%1st dimension denotes different tests, 2nd dimension
%is different targets/particles, in single target's case,
%3rd dimension is the cartesion coordinates)
L = size(xBuffer,3);
N = size(xBuffer,2);
gamma = sqrt(L/counter);
psiNewVec = zeros(N,4);

%loop on different instance, 1:N
for i = 1:N
    %compute sufficient statsitcs
```

```
    xBufferOneInstance(:,:) = xBuffer(1:2,i,:);

    xDiff = xBufferOneInstance(:,2:end) - xBufferOneInstance(:,1:end-1);

    YParticlesWithoutPhi = simData(...
        xBufferOneInstance,linksLocations,linksDistances,sigma);
    %Calculate sufficient statistics
    psiNewVec(i,1) = (norm(xDiff,2))^2;
    psiNewVec(i,2) = (norm(deltaYBuffer' - phi *...
        YParticlesWithoutPhi, 'fro'))^2;
    psiNewVec(i,3) = sum(sum((YParticlesWithoutPhi.*(deltaYBuffer'))));
    psiNewVec(i,4) = (norm(YParticlesWithoutPhi,'fro'))^2;
end

if nnz(psiNewVec(:,4)) ~= 0
    psiVec = (1-gamma) * psiVec + gamma * 1/N * sum(psiNewVec);
    phi = psiVec(3) / psiVec(4);
else
    psiVec(:,1:2) = (1-gamma) * psiVec(:,1:2) +...
        gamma * 1/N * sum(psiNewVec(:,1:2));
end

sigma_v = sqrt(psiVec(1) / (2*(L-1)));
if sigma_v == 0
    sigma_v = eps;
end
numLink = size(deltaYBuffer, 2);
sigma_w = sqrt(psiVec(2) / (numLink * L));

phiNewVec = zeros(1,size(phiVec,2)+1);
phiNewVec(1:end-1) = phiVec;
phiNewVec(end) = phi;
```

```
sigmavNewVec = zeros(1,size(sigmavVec,2)+1);
sigmavNewVec(1:end-1) = sigmavVec;
sigmavNewVec(end) = sigma_v;

sigmawNewVec = zeros(1,size(sigmawVec,2)+1);
sigmawNewVec(1:end-1) = sigmawVec;
sigmawNewVec(end) = sigma_w;
iteration = 1/gamma^2
true;
```

References

[1] N. Ahmed, M. Rutten, T. Bessell, S.S. Kanhere, N. Gordon, and S. Jha. Detection and tracking using particle filter based wireless sensor networks. *IEEE Trans. Mobile Computing*, 9(9):1332–1345, Sep. 2010.

[2] E. Aitenbichler and M. Muhlhauser. An IR local positioning system for smart items and devices. In *Proc. Intl. Conf. Distributed Computing Systems*, pages 334–339, Providence, RI, USA, May 2003.

[3] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Computer Networks*, 38(4):393–422, Mar. 2002.

[4] C. Alippi and G. Vanini. Wireless sensor networks and radio localization: a metrological analysis of the Mica2 received signal strength indicator. In *Proc. IEEE Intl. Conf. Local Computer Networks*, pages 579–582, Tampa, FL, USA, Nov. 2004.

[5] X. An, J. Wang, R.V. Prasad, and I. Niemegeers. OPT: On-line person tracking system for context-awareness in wireless personal network. In *Proc. Intl. Workshop on Multi-hop Ad Hoc Networks: From Theory to Reality*, pages 47–54, Florence, Italy, May 2006.

[6] C. Andrieu, J. De Freitas, and A. Doucet. Sequential MCMC for Bayesian model selection. In *Proc. IEEE Signal Processing Workshop on Higher-Order Statistics*, pages 130–134, Ceasarea, Israel, Jun. 1999.

[7] C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

[8] C. Andrieu, A. Doucet, and V.B. Tadic. On-line parameter estimation in general state-space models. In *Proc. IEEE Conf. Decision and Control/European Control Conf.*, pages 332–337, Seville, Spain, Dec. 2005.

[9] T. Arampatzis, J. Lygeros, and S. Manesism. A survey of applications of wireless sensors and wireless sensor networks. In *Proc. IEEE Intl. Symp. Control and Automation*, pages 719–724, Limassol, Syprus, Jun. 2005.

[10] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Trans. Signal Processing*, 50(2):174–188, Aug. 2002.

[11] P. Bahl and V. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proc. IEEE Intl. Conf. Computer Communications (INFOCOM)*, pages 775–784, Tel-Aviv, Israel, Mar. 2000.

[12] J. Carpenter, P. Clifford, and P. Fearnhead. Improved particle filter for nonlinear problems. *IEE Proc. Radar, Sonar and Navigation*, 146(1):2–7, 1999.

[13] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology. In *Proc. ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, pages 20–41, San Jose, Costa Rica, Apr. 2001.

[14] X. Chen, A. Edelstein, Y. Li, M. Coates, M. Rabbat, and A. Men. Sequential Monte Carlo for simultaneous passive device-free tracking and sensor localization using received signal strength measurements. In *Proc. ACM/IEEE Intl. Conf. Information Processing in Sensor Networks*, Chicago, IL, USA, Apr. 2011. to appear.

[15] Chipcon. *MicaZ Data Sheet*, 2006. available at `http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf`.

[16] H.D. Chon, S. Jun, H. Jung, and S.W. An. Using RFID for accurate positioning. *Journal of Global Positioning Systems*, 3(2):32–39, 2004.

[17] CISCO. *Cisco Aironet 1130AG Series IEEE 802.11A/B/G Access Point*, 2008. available at `http://www.cisco.com/en/US/prod/collateral/wireless/ps5678/ps6087/product_data_sheet0900aecd801b9058.pdf`.

[18] Crossbow Technology Inc. *2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*, 2004. available at `http://focus.ti.com/lit/ds/symlink/cc2420.pdf`.

[19] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[20] R. Douc and O. Cappe. Comparison of resampling schemes for particle filtering. In *Proc. Intl. Symp. Image and Signal Processing and Analysis*, pages 64–69, Zagreb, Croatia, Sep. 2005.

[21] A. Doucet, S. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[22] A. Doucet and N.J. Gordon. Simulation-based optimal filter for maneuvering target tracking. In *Proc. Signal and Data Processing of Small Targets*, pages 241–255, Denver, CO, USA, Jul. 1999.

[23] A. Doucet, N.J. Gordon, and V. Krishnamurthy. Particle filters for state estimation of jump Markov linear systems. *IEEE Trans. Signal Processing*, 49(3):613–624, Mar. 2001.

[24] A. Doucet and M. Johansen. A tutorial on particle filtering and smoothing: fifteen years later. In *Oxford Handbook of Nonlinear Filtering*, pages 656–704. Oxford University Press, 2011.

[25] A. Edelstein, X. Chen, Y. Li, and M. Rabbat. RSS-Based node localization in the presence of attenuating objects. In *Proc. Intl. Conf. Acoustics, Speech, and Signal Processing*, Prague, Czech Republic, May 2011. to appear.

[26] X.N. Fernando, S. Krishnan, H. Sun, and K. Kazemi Moud. Adaptive denoising at infrared wireless receivers. In *Proc. SPIE Annual Aerosense Symp. Infrared Technology and Applications*, pages 199–207, Bellingham, WA, USA, Sep. 2003.

[27] N. Furstenau, M. Schmidt, H. Horack, W. Goetze, and W. Schmidt. Extrinsic Fabry-Perot interferometer vibration and acoustic sensor systems for airport ground traffic monitoring. *IEE Proc. Optoelectronics*, 144(3):134–144, Jun. 1997.

[28] D. Gay, P. Levis, R. Von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC language: A holistic approach to networked embedded systems. In *Proc. ACM Conf. Programming Language Design and Implementation*, pages 1–11, San Diego, CA, USA, Jun. 2003.

[29] J. Geweke. Bayesian inference in econometric models using Monte Carlo integration. *Econometrica*, 57(6):1317–1339, 1989.

[30] N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proc. Part F: Radar and Signal Processing*, 140(2):107–113, Apr. 1993.

[31] Y. Gu, A. Lo, and I. Niemegeers. A survey of indoor positioning systems for wireless personal networks. *IEEE Communications Surveys and Tutorials*, 11(1):13–32, First Quarter 2009.

[32] M. Hazas and A. Ward. A novel broadband ultrasonic location system. In *Proc. Intl. Conf. Ubiquitous Computing*, pages 299–305, Goetborg, Sweden, Sep. 2002.

[33] T. He, P. Vicaire, T. Yan, L. Luo, L. Gu, G. Zhou, R. Stoleru, Q. Cao, J.A. Stankovic, and T. Abdelzaher. Achieving real-time target tracking using wireless sensor networks. In *Proc. IEEE Real-Time Embedded Technology and Application Symp.*, pages 37–48, San Jose, CA, USA, Apr. 2006.

[34] J. Hightower and G. Borriello. Location sensing techniques. *IEEE Computer Magazine*, 34(8):57–66, Aug. 2001.

[35] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning System: Theory and Practice*. Springer, 1993.

[36] T.T. Hsieh. Using sensor networks for highway and traffic applications. *IEEE Potentials*, 23(2):13–16, May 2004.

[37] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Trans. Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 34(3):334–352, Jul. 2004.

[38] A.M. Johansen and A. Doucet. A note on auxiliary particle filters. *Statistics and Probability Letters*, 78(12):1498–1504, 2008.

[39] D.B. Jourdan, J.J. Deyst Jr, M.Z. Win, and N. Roy. Monte Carlo localization in dense multipath environments using UWB ranging. In *Proc. IEEE Intl. Conf. Ultra-Wideband*, pages 314–319, Zurich, Switzerland, Sep. 2005.

[40] S.J. Julier and J.K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Proc. Intl. Symp. Aerospace/Defense Sensing, Simulation and Controls*, pages 182–193, Orlando, FL, USA, Apr. 1997.

[41] R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[42] M. Kanso and M. Rabbat. Compressed RF tomography for wireless sensor networks: centralized and decentralized approaches. In *Proc. IEEE Distributed Computing in Sensor Systems*, pages 173–186, Santa Barbara, CA, USA, Jun. 2010.

[43] N. Kantas, A. Doucet, S.S. Singh, and J.M. Maciejowski. An overview of sequential Monte Carlo methods for parameter estimation in general state-space models. In *Proc. IFAC Symp. System Identification*, pages 774–785, Saint-Malo, France, July 2009.

[44] T. King, S. Kopf, T. Haenselmann, C. Lubberger, and W. Effelsberg. COMPASS: A probabilistic indoor positioning system based on 802.11 and digital compasses. In *Proc. Intl. Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*, pages 34–40, Los Angeles, CA, USA, Sep. 2006.

[45] P. Kinney. *Zigbee technology: Wireless control that simply works*, 2003. available at `http://www.zigbee.org/resources`.

[46] G. Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

[47] G. Kitagawa. A self-organizing state-space model. *Journal of the American Statistical Association*, 93(443):1203–1215, Sep. 1998.

[48] D. Koller, G. Klinker, E. Rose, D. Breen, R. Whitaker, and M. Tuceryan. Real-time vision-based camera tracking for augmented reality applications. In *Proc. ACM*

Symp. Virtual Reality Software and Technology, pages 87–94, Lausanne, Switzerland, Sep. 1997.

[49] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer. Multi-camera multi-person tracking for easyliving. In *Proc. IEEE Intl. Workshop on Visual Surveillance*, pages 3–10, Dublin, Ireland, Jul. 2000.

[50] P.W.Q. Lee, W.K.G. Seah, H.P. Tan, and Z.X. Yao. Wireless sensing without sensors - An experimental approach. In *Proc. IEEE Intl. Symp. Personal, Indoor and Mobile Radio Communications*, pages 62–66, Tokyo, Japan, Sep. 2009.

[51] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, and E. Brewer. TinyOS: An operating system for sensor networks. In *Ambient Intelligence*, pages 115–148. Springer, 2005.

[52] Y. Li, X. Chen, M. Coates, and B. Yang. Sequential Monte Carlo Radio-Frequency tomographic tracking. In *Proc. Intl. Conf. Acoustics, Speech, and Signal Processing*, Prague, Czech Republic, May 2011. to appear.

[53] Y. Li, M. Thai, and W. Wu. *Wireless Sensor Networks and Applications (Signals and Communication Technology)*. Springer, 2007.

[54] A. Lin and H. Ling. Three-dimensional tracking of humans using very low-complexity radar. *Electronics Letters*, 42:1062–1063, Sep. 2006.

[55] A.J. Lipton, H. Fujiyoshi, and R.S. Patil. Moving target classification and tracking from real-time video. In *Proc. IEEE Workshop on Applications of Computer Vision*, pages 8–14, Princeton, NJ, USA, Oct. 1998.

[56] J. Liu, R. Chen, and T. Logvinenko. A theoretical framework for sequential importance sampling and resampling. In *Sequential Monte Carlo Methods in Practice*, pages 225–246. Springer, 2001.

[57] J.S. Liu and R. Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, 1998.

[58] K. Lorincz, D.J. Malan, T.R.F. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, M. Welsh, and S. Moulton. Sensor networks for emergency response: Challenges and opportunities. *IEEE Pervasive Computing*, 3(4):16–23, Dec. 2005.

[59] M. Moussa and M. Youssef. Smart devices for smart environments: Device-free passive detection in real environments. In *Proc. IEEE Intl. Conf. Pervasive Computing and Communications*, pages 53–58, Galveston, TX, USA, May 2009.

[60] L.M. Ni, Y. Liu, Y.C. Lau, and A.P. Patil. LANDMARC: Indoor location sensing using active RFID. *Wireless Networks*, 10(6):701–710, 2004.

[61] M.K. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, Jun. 1999.

[62] G. Poyiadjis, A. Doucet, and S.S. Singh. Particle methods for optimal filter derivative: application to parameter estimation. In *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing*, pages 925–928, Philadelphia, PA, USA, Mar. 2005.

[63] N.B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *Proc. Intl. Conf. Mobile Computing and Networking*, pages 32–43, Boston, MA, USA, Aug. 2000.

[64] N.B. Priyantha, A.K.L. Miu, H. Balakrishnan, and S. Teller. The cricket compass for context-aware mobile applications. In *Proc. Intl. Conf. Mobile Computing and Networking*, pages 1–14, Rome, Italy, Jul. 2001.

[65] T. Rydén. On recursive estimation for hidden Markov models. *Stochastic Processes and their Applications*, 66(1):79–96, 1997.

[66] M. Seifeldin and M. Youssef. Nuzzer: A large-scale device-free passive localization system for wireless environments. Technical report, Wireless Intelligent Networks Center (WINC), Nile University, Aug. 2009. available at `http://arxiv.org/PS_cache/arxiv/pdf/0908/0908.0893v1.pdf`.

[67] M.J. Skibniewski and W. Jang. Localization technique for automated tracking of construction materials utilizing combined RF and ultrasound sensor interfaces. In *Proc. ASCE Intl. Workshop on Computing in Civil Engineering*, pages 657–664, Pittsburgh, PA, USA, Jul. 2007.

[68] R. Steele, C. Secombe, and W. Brookes. Using wireless sensor networks for aged care: the patient's perspective. In *Proc. Pervasive Health Conf.*, pages 1–10, Innsbruck, Austria, Dec. 2006.

[69] P. Steggles and S. Gschwind. The Ubisense smart space platform. In *Advances in Pervasive Computing–Adjunct Proc. Intl. Conf. Pervasive Computing*, pages 73–76, Munich, Germany, May 2005.

[70] G. Storvik. Particle filters for state-space models with the presence of unknown static parameters. *IEEE Trans. Signal Processing*, 50(2):281–289, Aug. 2002.

[71] J.A.K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

[72] A. Taheri, A. Singh, and A. Emmanuel. Location fingerprinting on infrastructure 802.11 wireless local area networks (WLANs) using Locus. In *Proc. IEEE Intl. Conf. Local Computer Networks*, pages 676–683, Tampa, FL, USA, Nov. 2004.

[73] Y.C. Tseng, S.P. Kuo, H.W. Lee, and C.F. Huang. Location tracking in a wireless sensor network by mobile agents and its data fusion strategies. In *Proc. Intl. Conf. Information Processing in Sensor Networks*, pages 625–641, Palo Alto, CA, USA, Apr. 2003.

[74] M. Tubaishat and S. Madria. Sensor networks: an overview. *IEEE Potentials*, 22(2):20–23, May 2003.

[75] R. Van Der Merwe, A. Doucet, N. De Freitas, and E. Wan. The unscented particle filter. In *Advances in Neural Information Processing Systems (NIPS13)*, pages 584–590. MIT Press, 2001.

[76] F. Viani, L. Lizzi, P. Rocca, M. Benedetti, M. Donelli, and A. Massa. Object tracking through RSSI measurements in wireless sensor networks. *IEEE Electronics Letters*, 44(10):653–654, May 2008.

[77] F. Viani, P. Rocca, M. Benedetti, G. Oliveri, and A. Massa. Electromagnetic passive localization and tracking of moving targets in a WSN-infrastructured environment. *Inverse Problems*, 26(7):286–300, Mar. 2010.

[78] A.S. Voulodimos, C.Z. Patrikakis, A.B. Sideridis, V.A. Ntafis, and E.M. Xylouri. A complete farm management system based on animal identification using RFID technology. *Computers and Electronics in Agriculture*, 70(2):380–388, Mar. 2010.

[79] S.W. Wang, W.H. Chen, C.S. Ong, L. Liu, and Y.W. Chuang. RFID application in hospitals: a case study on a demonstration RFID project in a Taiwan hospital. In *Proc. Annual Hawaii Intl. Conf. System Sciences*, pages 184–193, Hawaii, HI, USA, Jan. 2006.

[80] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system. *ACM Trans. Information Systems*, 10(1):91–102, Jan. 1992.

[81] G. Welch and G. Bishop. An introduction to the Kalman filter. In *SIGGRAPH 2001 Course 8. Computer Graphics*. ACM Press/Addison-Wesley, 2001.

[82] J. Wilson. *Device-Free Localization with Received Signal Strength Measurements in Wireless Networks*. PhD thesis, University of Utah, Salt Lake City, UT, USA, Aug. 2010.

[83] J. Wilson and N. Patwari. See through walls: motion tracking using variance-based radio tomography networks. IEEE Trans. Mobile Computing, 2011. to appear.

[84] J. Wilson and N. Patwari. Radio tomographic imaging with wireless networks. *IEEE Trans. Mobile Computing*, 9(5):621–632, Jan. 2010.

[85] K. Woyach, D. Puccinelli, and M. Haenggi. Sensorless sensing in wireless networks: Implementation and measurements. In *Proc. Intl. Symp. Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, pages 1–8, Boston, MA, USA, Apr. 2006.

[86] N. Xu, S. Rangwala, K.K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *Proc. Intl. Conf. Embedded Networked Sensor Systems*, pages 13–24, Baltimore, MD, USA, Nov. 2004.

[87] M. Youssef, M. Mah, and A. Agrawala. Challenges: device-free passive localization for wireless environments. In *Proc. ACM Intl. Conf. Mobile Computing and Networking*, pages 222–229, Montreal, Canada, Sep. 2007.

[88] V.S. Zaritskii, V.B. Svetnik, and L.I. Shimelevich. Monte Carlo technique in problems of optimal data processing. *Automation and Remote Control*, 12:95–103, 1975.

[89] D. Zhang, J. Ma, Q. Chen, and L.M. Ni. An RF-based system for tracking transceiver-free objects. In *Proc. IEEE Intl. Conf. Pervasive Computing and Communications*, pages 135–144, White Plains, NY, USA, Mar. 2007.

[90] D. Zhang and L.M. Ni. Dynamic clustering for tracking multiple transceiver-free objects. In *Proc. IEEE Intl. Conf. Pervasive Computing and Communications*, pages 1–8, Galveston, TX, USA, May 2009.