MONTE CARLO METHODS FOR SIGNAL PROCESSING IN WIRELESS SENSOR NETWORKS

A Dissertation Presented

by

Mahesh Vemula

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

August 2007

Stony Brook University

The Graduate School

Mahesh Vemula

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree,

hereby recommend acceptance of this dissertation.

Petar M. Djurić, Advisor of Dissertation
Professor, Department of Electrical & Computer Engineering

Mónica F. Bugallo, Chairperson of Defence
Assistant Professor, Department of Electrical & Computer Engineering

Sangjin Hong,
Professor, Department of Electrical & Computer Engineering

Samir R. Das,
Professor, Department of Computer Science

This dissertation is accepted by the Graduate School

Lawrence Martin,
Dean of the Graduate School

Abstract of the Dissertation

MONTE CARLO METHODS FOR SIGNAL PROCESSING IN WIRELESS SENSOR NETWORKS

by

Mahesh Vemula

Doctor of Philosophy

in

Electrical Engineering

Stony Brook University

2007

Advances in the manufacture of low power and inexpensive micro-sensors, coupled with progress in distributed signal processing and networking are enabling sensor networks to be the computing paradigm of the 21st century. These networks provide us immense opportunities through monitoring and sensing of various phenomena. Some recent areas of research in sensor networks from a signal processing perspective include distributed detection, estimation, localization, target tracking, sensor selection, sensor data fusion, and sensor network management and organization. In this dissertation, we address three major problems in sensor networks

- Sensor self localization with beacon position uncertainty
- Target tracking using quantized data
- Fusion of random measures for target tracking.

These problems are addressed from a Bayesian standpoint, where the underlying principle is that all information about any phenomenon can be obtained through its posterior distribution. Analytical expressions for these posterior distributions, is in many scenarios unattainable, and therefore we resort to Monte Carlo methods.

Consider a scenario where sensors are randomly strewn over a surveillance region. These sensor networks are deployed for the purpose of tracking objects and people. To extract meaningful information about the location and dynamics of people and objects from signals received by the sensors precise knowledge of the sensors' location is essential. We address these tasks of sensor localization and target tracking independently since the joint problem is very computationally intensive.

(i) Sensor self localization with beacon position uncertainty

Along with these sensors are also beacon nodes which have the capability of obtaining their positions. In our work, we characterize the uncertainty in the beacon's location using probabilistic descriptions. These beacon nodes broadcast these descriptions so that the sensor nodes with unknown location information can utilize these descriptions along with the characteristics of the received signals to

obtain estimates of their positions. This process is repeated periodically to ensure that all the nodes of the network are localized. At each sensor node we employ Bayesian methods for incorporating this uncertainty and combining data from several beacons. Due to the non-linearity of the measured data closed form solutions for obtaining the sensor location cannot be obtained, therefore we utilize Monte Carlo-based Importance Sampling for obtaining the distribution of the sensor's location.

(ii) Target tracking using quantized data

An important constraint these networks present are limited communication and power resources. To address these challenges we propose to track and localize the objects using the target emitted signal characteristics obtained by the sensor. Traditionally these characteristics are forwarded by the sensor to a fusion node. In view of the above constraints, we propose methods which only require minimal quantized information transmitted by the sensor. To this end, we consider target tracking only using binary ($'1'$, and $'0'$) levels of data. Clearly this highly quantized nature of the data presents a high degree of non-linearity in the measurement data which cannot be handled using classical target tracking methods. We pose the problem in a Bayesian framework and attempt to obatin recursively the posterior distribution of the target dynamics This posterior distribution is obtained using a class of Monte-Carlo algorithms known as Particle Filters which approximate these evolving target posterior densities in a sequential manner using a random measure which is a weighted set of samples. Thus, inferences about the targets dynamics can be made using these random measures.

(iii) Fusion of random measures for target tracking

In a distributed architecture for target tracking, sensors form clusters and transmit their measurements to a specialized node known as a leader node. Upon obtaining the measurements from the sensors, the leader nodes estimate the posterior density of the target's dynamics and collaborate with the neighboring sensor, leader and fusion nodes. Using Monte Carlo methods, these leader nodes represent this posterior density as a random measure. The summaries of these random measures are then transmitted to a fusion node which combines them to obtain a global summary.

To my elder brother Satish Vemula

Contents

List of Figures

List of Tables

Acknowledgments

The Japanese equivalent word for a teacher “Sensei”, translates “to one who has gone before”. Professor Djurić was our Sensei. It is with a deep sense of gratitude that I thank him for providing me this great opportunity of being his student. Under him, I have learnt to first walk and then run among the deep and unknown jungles of research in signal processing. I have no words to express my thanks to him for making me the researcher I am today.

I take the opportunity to thank my parents who have gone through several hardships to educate me. I also thank my brothers for those constant support and encouragement at several stages of my Ph.D. career.

I thank my co-advisor Ms. Mónica F. Bugallo who apart from co-advising me on my research, has taught me the art of effective writing and presentation. I thank Prof Sanjin Hong and Prof Samir Das for kindly consenting to be members of my Ph.D. committee.

Through the seminar courses on sensor networks provided at the Computer Science Department at Stony Brook, I have been able to pick up some good research habits. I would like to thank Prof Samir Das and Prof Himanshu Gupta for giving me the opportunity to let the frog in me see beyond the well. Special thanks to

Constantinos Papadias and Dan Avidor of Bell Labs, Crawford Hill, for instilling in me the confidence to pursue research.

I also take the opportunity of thanking all my colleagues past and present at the COSINE lab, Yufei Huang, Tadesse Ghirmai, Miodrag Bolic, Katrien De Cock, Xueying Zhang, Jae-Chan Lim, Akshay Athelye, Yao Li, Zejie Zhang, Ting Lu, Mingyi Hong and Vibha Mane, for providing me with a congenial atmosphere. It has been a really a good learning experience when they would share some of their research experiences in our regular journal club meetings.

I would like to thank Scott Tierno, Computer Engineer, who would encouraged me to lookout for the light at the end of the tunnel. I also am very thankful to the secretaries of our Department, Ms Deborah Kloppenburg, Carolyn Huggins, and Judy Eimer for their cooperation and support.

My years at Stony Brook have been extremely pleasant. This has largely been due to my friends Meher Mahesh and Rajesh Elisetty. I thank them both for teaching me many aspects of life. I also take the opportunity to thank Dr. Manoj Muthukuru who has been in many ways an inspiration to many of us through his hardwork and perseverance. I also thank Satprem Reddy and Rohan Kulkarni for their camaraderie.

Chapter 1

Overview of the Dissertation

"As we know, there are known knowns. There are things we know we know. We also know there are known unknowns. That is to say,we know there are some things we do not know. But there are also unknown unknowns - the ones we don't know we don't know."

- Donald Rumsfeld

Wireless sensor networks provide us with a plethora of challenges and opportunities to identify and estimate various physical phenomena such as the dynamics of a target or a temperature map of a particular region. The inferences about a phenomenon is made using the sensor observations which are often corrupted or embedded in noise. Very often the observations are non-linear functions of the unknown state of the phenomenon. These non-linearities and random environments often lead to potential inaccuracies with classical estimation methods. While these networks provide us with many opportunities, they are also constrained in power and bandwidth resources. To this end, we investigate the usage of Monte Carlo methods under such resource constrained scenarios for estimating unknown static and dynamic state parameters. Some examples of these unknown parameters are the position and velocity of an object or the relative location of sensor itself.

In Chapter 2, we provide a brief overview of some of the areas of research of signal processing in sensor networks. Most signal processing methods can be classified under two distinct areas detection (or classification) and estimation. The focus of this dissertation has been towards the estimation of unknown parameters. We motivate the use of Monte Carlo methods for signal processing in Chapter 3, and briefly summarize procedures for obtaining Monte Carlo-based representations of distributions that describe the unknown parameters using the observations.

The numerical procedures discussed in Chapter 3, require knowledge of the distributions of the uncertainties in the system. In Chapter 4, we consider novel procedures termed as Cost Reference Particle Filtering for estimating time varying parameters under unknown probabilistic distributions of the state and measurement noise processes.

In Chapter 5, we introduce a distributed framework for sensor localization using reference nodes also termed as beacons. These beacons are sensor nodes which have some descriptions of their locations. In typical localization frameworks, the true locations of the beacons is often assumed; here we describe the positions of the beacons using probabilistic distributions. The location description of the other sensor nodes is obtained by fusing the signals and locations distributions of the beacons.

As discussed above these networks are constrained in bandwidth and power resources. Therefore in Chapter 6, we study particle filtering-based methods for obtaining distribution of the unknown target state when the sensors transmit two level quantized data instead of the entire sampled raw data. We also identify the loss in accuracy with such quantization.

In Chapter 7, we discuss distributed frameworks for combining non-parametric distributions of unknown parameters under known and unknown noise probability distribution scenarios. In that chapter, we investigate the fusion of distributions for tracking targets where the sensors have relatively large communication and computation power. Fusion of distributions for target tracking in a resource constrained framework is analyzed in Chapter 8. Here we look at specific distributed hierarchical architectures which are amenable to handle the resource constraints.

In Chapter 9, we consider an asynchronous sensor network, in which the local clocks of the sensors are misaligned and the corresponding offsets are unknown. We design recursive algorithms for tracking the unknown target dynamics and the sensor timing offsets.

In Chapter 10, we propose metrics to quantify the performance of a certain class of these procedures which approximate the above distribution with a Gaussian. The Gaussian approximated distributions are compared with distribution obtained with standard Monte Carlo method.

Chapter 2

Sensor Networks : A Signal Processing Perspective

"The senses are the organs by which man places himself in connexion with exterior objects."

- Jean Anthelme Brillat-Savarin

2.1 Introduction

Networking a large number of varied sensors to monitor, collect, disseminate and intelligently combine information about a specific task is the emerging paradigm of this century. The presence of sensors in our day to day lives is not new, and these processing elements were relatively expensive and in most cases 'dumb'. However, this century is witnessing an upsurge in technology towards the manufacture of low cost microelectromechanical systems which include components like sensors, actuators, transducers and many other such RF components. These devices are not ‘dumb’, but they have the added capability of collaborating and exchanging information with its neighbors, and intelligently analyze this data. Being inexpensive these sensors are deployed in large numbers. For instance, microsensors are densely deployed on a skyscraper to monitor the vibrations due to seismic waves or wind gusts.

The number of data received from these sensors can be immense and very rich with information. This huge amount of data can be processed to estimate, detect and predict various phenomena. One example is the prediction of onset of earthquakes, in earthquake prone areas thereby providing a great service to mankind. Futuristically speaking such smart sensors will find their ways in almost all aspects of our lives. The potential of such sensors is thus undoubtedly huge and may be considered as the first few examples of ‘ubiquitous computing’. This chapter dwells into some aspects of these sensor networks with some motivating applications.

In Section 2.2 the basic elements of a sensor network and architectures are described followed with some real-life applications of smart sensor networks. In Section 2.3 we provide a brief introduction to some active areas of research of signal processing in sensor networks.

2.2 Sensor Networks: Elements, Challenges and Applications

2.2.1 Sensors and Sensor Networks

“A sensor is a transducer that receives an input signal or stimulus and responds

with an electrical signal which bears a known relationship with the input" [1]. Alternatively a sensor can be thought of as a device that maps a physical phenomenon to a qunatitative measurement. Mathematically we have

$$S(E,t) \longrightarrow \{V(t), \epsilon(t)\} \tag{2.1}$$

where $S(\cdot)$ maps the environment E at time t to a numerical value V, which is surrounded with a certain uncertainty ϵ [2]. Sensors are of two kinds: passive and active sensors. Passive sensors directly generate an electrical signal in response to stimulus and do not require any additional power source. Typical examples of passive sensors include thermocouples, pyroelectric and piezoelectric detectors. Active sensors require external power for their operation. This external signal is modified by the sensor to produce the output signal. Examples of active sensors include thermistor (temperature sensitive sensors), acoustic and seismic sensors. Thermistors comprise of materials whose resistance varies with temperature which can be detected by passing an external current across the thermistor, hence an external power is essential. Sensors are classified depending upon their sensing applications, dynamic ranges of operation, accuracy, resolution and other characteristics. An exhaustive treatment of sensors is beyond the scope of this report however a good reference to understand the physics and the design involved in these sensors is [1].

Modern sensor nodes consist of a sensing unit, a processing unit, a transceiver and a power unit [3]. The sensing units are composed of sensors and an analog-to-digital converter (ADC). The processing unit is responsible for the actual processing of the raw data.The raw data it processes is obtained either from the ADC unit or from the transceiver when it is in a collaborative mode[1]. The transceiver transmits the data from the sensor node to other nodes or fusion centers. In a collaborative mode it also serves to receive data from neighboring sensor nodes. The power unit supports these units for performing these tasks of sensing, computation and communication. The power unit consists of batteries or other power replenishing units like solar cells. Depending upon the context the term 'sensor' refers to either the whole node or the sensing unit.

By integrating these sensors with the technology to communicate and collaborate we have a Smart Wireless Sensor Network. Sensor networks can be broadly classified according to their architectures as centralized, distributed and hierarchical.

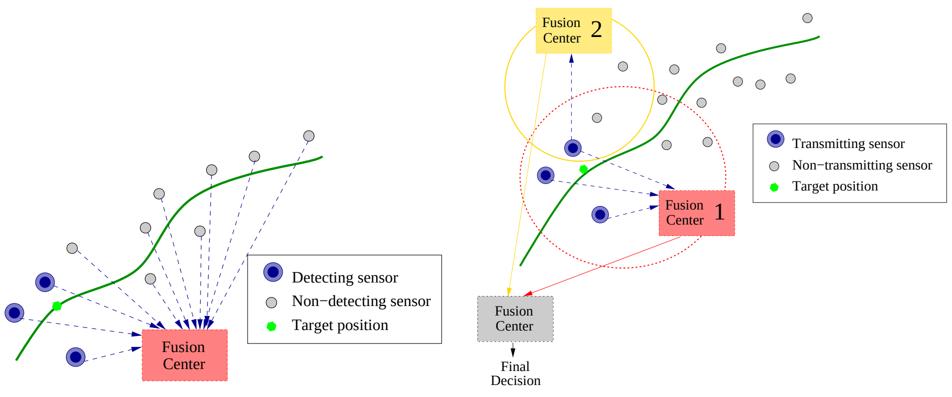
- Centralized Architectures: In this configuration the sensors communicate only with a centralized unit known as the root node (RN) or the fusion center(FC) as shown in 2.1(a). The sensor nodes provide the FC with the data about the sensed event and the FC processes the data. The FC processes the data, combines it and provides a meaningful interpretation to the varied and disparate sensor data. In such architectures the computational load can be shared between the sensor node and the FC with the FC being a high performance computing unit. The disadvantages of such network are the requirements of huge communication bandwidths and robustness of the FC. Such architectures are also not easily scalable.

- Distributed Architectures: In these architectures the sensors sense a given event process the data and exchange information locally among its neighbors thereby

[1]The mode of operation wherein sensors exchange information with its neighbors and collaborate to obtain useful information about the physical world.

requiring no centralized units. Such architectures are more robust, however these sensors need to be more sophisticated in terms of exchanging information among it neighbors in an optimal manner.

- Hierarchical Architectures: In these architecture there are several tiers/clusters of sensor nodes and each of these clusters has a cluster head (CH) or leader node (LN). The sensors communicate with the CHs and the CHs can forward data to and from their LNs as shown in 2.1(b). These clustering architectures are very useful for purposes of fast querying and are easily scalable [4], [5].



(a) Centralized (b) Hierarchical

Figure 2.1: A target tracking scenario in (a) a centralized sensor network and (b) hierarchical sensor network .

2.2.2 Challenges

Conventional wireless networks are driven with the sole motive of providing increased data throughputs without any wires. The purpose of a sensor network is however very different. Sensor networks are deployed in environments for the purpose of providing distributed monitoring and sensing. In many wireless sensing scenarios a major problem is the lack of a clear line of sight (LOS). By deploying sensors in dense numbers and spanning over large areas, problems associated with lack of LOS path and low signal to noise ratios (SNRs) are eliminated. These sensors then transmit information about the sensed event to its leader node or to a centralized processing unit. However in wireless environments the signal attenuates with distance as $\frac{1}{r^\alpha}$ where α is the attenuation constant of the environment. Typical values of α lie bewteen 2 to 4. Therefore a major amount of power is consumed in communication between the sensors and FC [6].

In general, sensor networks are composed of hundreds of microsensors which are deployed for instance along battle lines to track enemy movements. In such situations these sensor networks consist of a variety of sensors namely acoustic sensors, seismic sensors, and magnetometers. General characteristics and challenges involved in the operation of these sensor networks are

- *Limited in power, computational and memory resources:* Sensors are unfortunately highly resource constrained. They are limited in power, storage, communication and computation. Therefore algorithms and protocols are to be built which take into consideration these factors.

- *Of different modalities:* These sensors may sense the same phenomenon however the raw data of these sensors may refer to different aspects of the phenomenon. As an example, the movement of troops would cause the acoustic sensor to record changes in the sound patterns due to movement of troops, thermal sensors would record abrupt changes in temperature patterns due to the presence of troops and so on with the other kinds of sensors. Some interesting challenges involve the synergistic use of multiple sensors which provide varied information to achieve maximum performance.

- *Prone to failures:* Sensor networks are deployed in hostile and noisy environments. Also these sensors are very prone to failures, hence the networks should be self-learning and self-healing. Due to the failure of sensors the topology of the sensor network no longer remains same. So in such situations the sensors should adapt and reconfigure themselves. As an example, in sensor networks deployed for monitoring the temperature over a region, if a certain set of nodes fail then the remaining sensor nodes should offset the information loss due to the impaired sensors by reconfiguring themselves and providing the temperature information of those regions previously monitored by the impaired sensors.

- *Dense Deployment:* Such dense deployment of sensors not only provides a wide range of information but also has the usual problems of congestion in networks. An equally interesting problem in such dense networks is to collect 'informative data', which depends on selecting sensors which would provide maximum information.

- *Deployed in Time Varying Environments:* The sensor networks are typically deployed in areas where the environment is rapidly changing. Therefore the sensors should also adapt to these rapidly varying environments.

Thus wireless sensor networks provide with interesting challenges and applications.

2.2.3 Applications

Some interesting case studies of sensor networks are as follows:

- *Habitat Monitoring at Great Duck Island (coast of Maine):* The Great Duck Island off the coast of the island of Maine is a breeding ground for a certain species of sea birds known as "Leach's' Storm Petrels". Seabird researchers have been interesting in studying how the ecology influences the breeding patterns of these seabirds. The Intel Research Laboratory at Berkeley in collaboration with the Universities of California deployed a wireless sensor network on this island around and in the burrows of these seabirds. Several kinds of sensors (light, temperature, humidity, and thermopile sensors) are used to allow a non-disruptive monitoring of the habitat surrounding the burrows. These sensors then transmit the data to a sensor gateway which has an external antenna (a Yagi Uda antenna). The gateway provides communication over long distances from the areas around the burrow to a central units situated far away from the island. This data is then later transmitted to the respective research labs for analysis [7, 8].

- *Glacier Monitoring:* Environmental phenomena like global warming and climatic changes is studied by monitoring the glacial fluctuations i.e., changes in the glacier mass, volume, area and length. The sea-level changes, and these deformation patterns of the glaciers are among the key inputs to such studies. Traditional methods involve seismic and radar techniques which are low resolution techniques. To combat some of these drawbacks in these methods, a team from the University of Southampton U.K. had deployed a network of wireless sensor network inside a Norwegian glacier. These sensors are deployed in the sedimentary base of the glacier. Recordings of temperature, pressure and motion parameters of the ice and the sediment is transmitted to a base station which in turn transmits this information to other central units and labs. [9].

- *Oregon Vineyard:* The Intel Research Laboratory at Berkeley deployed a sensor network, in a vineyard, but with a difference. The sensors measure the temperature, moisture content in soils and other environmental factors. Typically the data in sensor networks is transmitted to research labs where scientists and researchers analyze the data. However the end users of the sensor networks are vineyard owners to whom such raw data is of very little use if any. Therefore a proactive sensor system based on ethnographic research methods was implemented where the end output of these systems is the detection of the presence or absence of parasites, optimal water rationing to plants and other such useful tasks. Another interesting aspect of these networks is their reconfigurability. The sensor networks at day times monitor the presence of birds and in the nights reconfigure to monitor the amount of frost in the vineyards [10].

- *Smart Bridge:* The Rion-Antirion Bridge near the Peloponneus region and the Western part of Greece is equipped with 300 sensors. These sensors include stress gauges on gussets that monitor the fatigue of the bridge framework, displacement transducers on stay cables that monitor the motion of the bridge with wind gusts, and also accelerometers on the roadways to measure impact of earthquakes. The bridge has four piers and each pier has a data acquisition center to which the sensors transmit their data. These data units then transfer this information to a central office near the bridge and from the central unit to the operating offices using the Internet [11].

The list of application of sensor networks in day to day life is endless.

2.3 Signal Processing in Sensor Networks

Following the sensing of the physical phenomenon, one is plainly left with mere numbers and only an intelligent processing of these numbers would make any sense about the phenomenon. These numbers are the signal representations of the parameters of the event and these procedures can be broadly classsifed as signal processing.

Signal Processing activities in sensor networks fall in the following areas

- ⋆ Source localization in space and time.
- ⋆ Collaborative signal processing.
- ⋆ Distributed signal estimation, detection, and classification.
- ⋆ Distributed calibration in sensor networks e.g., time synchronization.
- ⋆ Applications of distributed sensor networks.
- ⋆ Multisensor data fusion.

2.3.1 Localization

Localization is the mechanism by which spatial relations between target and the sensor nodes is established. Specifically this refers to the estimation of the target's location with respect to the sensor nodes by measuring the acoustic, seismic, infrared or thermal signatures. Popular techniques involve measurement of the time of arrival (TOA), direction of arrival (DOA) and received signal strength indices (RSSI). Using standard methods of triangulation, trilateration, and method of least squares one can easily obtain the location of the target. Of great interest are other issues like the minimum number of sensors required to attain a certain amount of precision in the location of the target and the arrangement of sensors (constellation patterns) to provide adequate coverage of the sensing area. Other important aspects of these problems involve obtaining the Cramer Rao Lower Bound of these estimators [12], [13], [14], [15], [16].

2.3.2 Collaborative Signal Processing

To obtain an extensive information of an event in a noisy environment a dense network is necessary. The challenges in such dense network involve collaborative sensor activites for efficient utilization of resources. The sensors spatially sample the phenomena of interest and this raw data are combined, depending upon the sensor network architecture, at the nodes, leader nodes or at the FC in such a manner that the maximum benefits are achieved. Another interesting aspect in sensor networks is to dynamically query sensors and route data, under power and bandwidth constraints to obtain the maximum information gain. In [17], an information utility measure is introduced to select sensors for purpose of querying and dynamical routing of data in a sensor network.

2.3.3 Distributed Signal Estimation, Detection and Classification

Distributed methods have many advantages over centralized algorithms in terms of reducing the communication bandwidth, reduced computational load and increased robustness. However, in order for the sensor networks to achieve an optimal performance, specialized multisensor fusion techniques need to be developed.

Distributed estimation algorithms have been studied in the context of target tracking. However most of these algorithms restrict to linear systems and Kalman Filtering. Recent works in distributed estimation are [18] and [19]. In [18], algorithms for distributed particle filtering are outlined, while in [19] maximum likelihood estimation based on based on data collected by a sensor network under power and bandwidth constraints is considered.

Distributed Detection methods have been quite extensively researched [20]. Typically a local decision at each sensor is performed by comparing the signal received with a threshold and these local decisions are fused based upon the sensor network topology to obtain a global decision. The problems involved in this area involve finding the optimal decision-fusion rules in uncertain environments and optimal sequential and non-sequential decision making methods. The methods employed involve Bayesian, Neyman-Pearson criterion, Dempster-Shafer theory and other parametric and nonparametric approaches.

In the light of multiple target tracking, the target is first detected and then classified based upon the sensor node measurements. If the classification of the target is of the desired kind then tracking of the target is initiated. Thus, it is evident that distributed methods for detection, estimation, classification are closely knitted in a multi-sensor networking scenario where a decision rule based on data from multi-sensor data is made and using this decision a classification is performed. If the classification is of the desired kind, then estimation in the light of this prior knowledge will enhance our results [21].

We have seen in this chapter, some of the basic elements of sensor networks. The challenges of combining data from several scenarios and the constraints of power and bandwidth pose many interesting problems of signal processing in the context of sensor networks.

Chapter 3

Sequential Monte Carlo Methods

"Anyone using Monte Carlo is in a state of sin."

- John von Neumann

In this chapter we motivate the need for Monte Carlo methods in parameter estimation. For the case of time-varying parameters classical methods fail to provide an accurate solution when the state space equations are non-linear and involve non-Gaussian noise processes. Sequential Monte Carlo (SMC) algorithms are a class of MonteCarlo algorithms that provide an elegant numerical solution to the filtering problem.

We motivate the study of SMC methods through an initial understanding of Importance Sampling, a Monte Carlo-based density estimation method in Section 3.2. In Section 3.3, we introduce the necessary terminology and the dynamic state space model and in Section 3.4, we define the filtering problem and the various filters. We also briefly introduce the extended Kalman filter, the Unscented Kalman filter, the standard particle filter, the auxiliary particle filter, and the gaussian particle filter.

3.1 Introduction

In many fields of science and engineering, one is often beset with the problem of finding or estimating unknown parameters using some measurements. For this purpose a statistical model is assumed that describes the dependence of the measurements and the parameters and is justified through rigorous experiments within reasonable limits of accuracy. This model is typically a function of the parameter of the system and other random processes which characterize the uncertainty in the system or the uncertainty in the measurement process itself.

Consider a simple model of the form

$$\mathbf{y} = \mathbf{f}(\boldsymbol{x}) + \boldsymbol{\epsilon}$$

where $\mathbf{y} = [y_1, y_2, \cdots, y_N]$ represents the data, $\boldsymbol{x}$ represents the state of the system and $\boldsymbol{\epsilon}$, the uncertainties in the system. The state parameter $\boldsymbol{x}$ is not directly observable, but is partially observable through the measurements. The problem then is the estimation of $\boldsymbol{x}$ using the observed data $\mathbf{y}$. This problem is addressed by a plethora of methods which depend upon (a) the nature of the function $\mathbf{f}(\boldsymbol{x})$ i.e., whether it is linear or non-linear, (b) the knowledge of the probability distributions of $\boldsymbol{\epsilon}$, e.g., whether it is Gaussian or non-Gaussian or in general whether its distribution is known or unknown (c) the arrival of the data $\mathbf{y}$ is sequential or batch-wise.

In statistics, this problem is typically addressed under two frameworks namely Maximum-Likelihood and Bayesian.

Maximum-Likelihood: With this method, one obtains the joint likelihood distribution of the observed data conditioned upon the state $\boldsymbol{x}$ as

$$p(\boldsymbol{y}\,|\,\boldsymbol{x}) = p(y_1, y_2, \cdots, y_N\,|\,\boldsymbol{x}).$$

The likelihood function is then maximized to obtain our parameter of interest $\boldsymbol{x}$. The maximum likelihood estimator (MLE) has many interesting asymptotic properties of being unbiased, achieving the Cramer-Rao lower bound (CRLB) [1] and asymptotically it also possesses a Gaussian probability density function. When analytical expressions for the MLE are not easily available, the maximization step in the MLE is obtained by using standard optimization techniques such as Newton-Raphson, and gradient descent methods. However these numerical methods suffer from typical problems of being stuck at local maxima and inability to explore the system space efficiently .

Bayesian Methods: In the classical likelihood approach the parameter $\boldsymbol{x}$ is assumed to be a deterministic or fixed parameter, however in the Bayesian framework a different philosophy is adopted. Here the parameter of estimation is itself inherently modeled as a random parameter and also any aprior knowledge of the parameter is incorporated. By modeling $\boldsymbol{x}$ as a random parameter, the estimate is one particular realization of the random quantity. Thus, under this setting we define $p(\boldsymbol{x}\,|\,\boldsymbol{y}) = p(\boldsymbol{x}\,|\,y_1, y_2, \cdots, y_N)$, as the posterior distribution which contains all the information about $\boldsymbol{x}$. This distribution can be obtained using Bayes' theorem as

$$p(\boldsymbol{x}\,|\,\boldsymbol{y}) = \frac{p(\boldsymbol{y}\,|\,\boldsymbol{x})p(\boldsymbol{x})}{p(\boldsymbol{y})} = \frac{p(\boldsymbol{y}\,|\,\boldsymbol{x})p(\boldsymbol{x})}{\int p(\boldsymbol{y}\,|\,\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}}.$$

Simple inference such as $\mathbb{E}_p[g(\boldsymbol{x})]$, where $\mathbb{E}_p$ denotes the expectation over the

[1]The CRLB is the lower bound on the variance of unbiased estimators.

random variable $\boldsymbol{x}$ whose distribution is given by p and $g(\boldsymbol{x})$ is any function depending upon the parameter $\boldsymbol{x}$

$$\mathbb{E}_p[g(\boldsymbol{x})] = \int_{\boldsymbol{x}} g(\boldsymbol{x})p(\boldsymbol{x} \mid \boldsymbol{y})d\boldsymbol{x} = \int_{\boldsymbol{x}} g(\boldsymbol{x})\frac{p(\boldsymbol{y} \mid \boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}}{\int p(\boldsymbol{y} \mid \boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}}. \tag{3.1}$$

The computation of the integration is often carried out using numerical quadrature methods. However, a major drawback associated is efficient spanning of the parameter space especially in higher dimensions.

Monte Carlo methods are known to alleviate some of the problems associated with some these numerical methods.

3.2 Monte Carlo Integration

Consider a simple variation of the previous inference problem (3.1), i.e., the evaluation the integral

$$\mathbb{E}_q[g(\boldsymbol{x})] = \int_{\boldsymbol{x}} g(\boldsymbol{x})q(\boldsymbol{x})d\boldsymbol{x} \tag{3.2}$$

under the distribution $q(\boldsymbol{x})$. With a simple Monte-Carlo strategy, a straightforward procedure is to draw samples $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(n)}\}$ from the density $q(\boldsymbol{x})$ to numerically evaluate the integral (3.2) as

$$\mathbb{E}_q[g(\boldsymbol{x})] \approx \hat{g}_n = \frac{1}{n}\sum_{i=1}^{n} g(\boldsymbol{x}^{(i)}). \tag{3.3}$$

Under the strong law of large numbers $\hat{g}_n$ converges almost surely to $\mathbb{E}_q[g(\boldsymbol{x})]$. The variance of $\hat{g}_n$ is given as $\frac{1}{n}Var_q[g(\boldsymbol{x})]$ [22]. As noted earlier, traditional numerical integration methods have serious drawbacks in situations when $\boldsymbol{x}$ is a high dimensional vector. In such cases, the convergence rate of the methods such as Simpson's rules is very poor as the dimension of $\boldsymbol{x}$ increases. Intuitively, numerical methods basically cover the entire space with a grid and as the number of dimension increases the grid space also increases exponentially, thus requiring many number of points for computation. However, Monte Carlo methods do not rely on such deterministic grids. A drawback of Monte Carlo methods is that the above results are asymptotic, i.e., the variance of the estimator goes to zero when the number of points approaches infinity. There exist several Monte Carlo methods which reduce the variance of the estimator, of which importance sampling is a popular sampling technique.

3.2.1 Importance Sampling

An alternative representation to (3.2) is given as

$$\mathbb{E}_q[g(\boldsymbol{x})] = \int_{\boldsymbol{x}} g(\boldsymbol{x}) \frac{q(\boldsymbol{x})}{\pi(\boldsymbol{x})} \pi(\boldsymbol{x}) dx \tag{3.4}$$

By drawing samples $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(n)}$ are from a distribution $\pi(\boldsymbol{x})$, (3.4) can be approximated as

$$\begin{aligned} \mathbb{E}_q[g(\boldsymbol{x})] &\approx \frac{1}{n} \sum_{i=1}^{n} g(\boldsymbol{x}^{(i)}) \frac{q(\boldsymbol{x}^{(i)})}{\pi(\boldsymbol{x}^{i})} \\ \hat{g}_n &= \frac{1}{n} \sum_{i=1}^{n} g(\boldsymbol{x}^{(i)}) w^{(i)} \qquad (3.5) \end{aligned}$$

where $\pi(\cdot)$ is called the proposal or importance function from which samples are drawn and $w^{(i)}$ is the associated importance weight. The density function $q(\cdot)$ can be therefore approximated as

$$q(x) \approx \sum_{i=1}^{n} w_i \boldsymbol{\delta} \left(\boldsymbol{x} - \boldsymbol{x}^{(i)} \right). \tag{3.6}$$

Thus in Importance Sampling, any probabilistic distribution is represented with samples and corresponding weights. This representation can then be used easily in computation of various estimates. The variance of the estimator $\hat{g}_n$ is then given as

$$\begin{aligned} Var(\hat{g}_n) &= \frac{1}{n}\left(\mathbb{E}_\pi \left[\frac{g(\boldsymbol{x})q(\boldsymbol{x})}{\pi(\boldsymbol{x})} \right]^2 - \hat{g}_n^2 \right) \\ &= \frac{1}{n}\left(\mathbb{E}_\pi \left[\frac{g^2(\boldsymbol{x})q^2(\boldsymbol{x})}{\pi^2(\boldsymbol{x})} \right] - \hat{g}_n^2 \right) \\ &\geq \frac{1}{n}\left(\left(\mathbb{E}_\pi \left[\frac{|g(\boldsymbol{x})|q(\boldsymbol{x})}{\pi(\boldsymbol{x})} \right] \right)^2 - \hat{g}_n^2 \right) \qquad \{\text{Using Jensen's Inequality}\} \\ &= \frac{1}{n}\left(\left(\int_{\boldsymbol{x}} |g(\boldsymbol{x})|q(\boldsymbol{x})d\boldsymbol{x} \right)^2 - \hat{g}_n^2 \right) \end{aligned} \tag{3.7}$$

which provides a lower bound on the variance of estimator and also suggests possible choices of the proposal function π. Typically choices of π are distributions which is nearly proportional to q [22].

In the previous sections we have considered $\boldsymbol{x}$ as a static parameter. In some scenarios, the parameter of interest is also often a time varying parameter and observations of this time varying quantity are recorded. In the following sections, we consider the sequential estimation of the $\boldsymbol{x}_t$ using the observations.

3.3 Dynamic State Space Model

In scenarios such as target tracking using radar , the dynamics of the target, i.e., its position, velocity and acceleration are to be obtained using the radar measurements. In wireless communications, the channel conditions are to be estimated using the received symbols. Clearly one can see that a common model exists that explains the evolvement of the states of the system and the acquisition of observations.

The discrete state space (DSS) model is one such indispensable mathematical model that describes the evolution and the measurement function of the state [23]. A standard formulation of the model is as follows:

$$\boldsymbol{x}_t = g(\boldsymbol{x}_{t-1}) + \mathbf{u}_t \tag{3.8}$$

$$\boldsymbol{y}_t = h(\boldsymbol{x}_t) + \mathbf{v}_t \tag{3.9}$$

where $\boldsymbol{x}_t$ is the unknown state at time t, $g(\cdot)$ is the state transition function, $\boldsymbol{y}_t$ are the measurements, $h(\cdot)$ is the measurement function and $\mathbf{u}_t$ and $\mathbf{v}_t$ are observation and measurement noise processes respectively.

The objective in many scenarios is to estimate and recover $\boldsymbol{x}_t$ which varies with time and is not directly measurable. The measurements $\boldsymbol{y}_t$ are functions of the quantities and are often corrupted by noise. Equations (3.8) and (3.9) are commonly referred to as the state space equations.

3.4 The Filtering Problem

The estimation of $\boldsymbol{x}_t$ using the measurements up to time t i.e., $\{\boldsymbol{y}_0, \boldsymbol{y}_1, \cdots, \boldsymbol{y}_t\}$ is defined as the filtering problem. In a Bayesian framework, the problem is broadly defined as the estimation of the distribution of the random variable, $\boldsymbol{x}_t$, given the sequence $\{\boldsymbol{y}_0, \boldsymbol{y}_1, \cdots, \boldsymbol{y}_t\}$ [23]. We will henceforth denote the sequence $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_t\}$ as $\boldsymbol{x}_{0:t}$ and $\{\boldsymbol{y}_0, \boldsymbol{y}_1, \cdots, \boldsymbol{y}_t\}$ as $\boldsymbol{y}_{0:t}$. The conditional density $p(\boldsymbol{x}_t \,|\, \boldsymbol{y}_{0:t})$, is termed as the filtering density.

We can write

$$\begin{aligned}
p(\boldsymbol{x}_t|\boldsymbol{y}_{0:t}) &= p(\boldsymbol{x}_t|\boldsymbol{y}_t, \boldsymbol{y}_{0:t-1}) && (3.10)\\
p(\boldsymbol{x}_t|\boldsymbol{y}_{0:t}) &= \frac{p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{y}_{0:t-1})p(\boldsymbol{x}_t|\boldsymbol{y}_{0:t-1})}{p(\boldsymbol{y}_t|\boldsymbol{y}_{0:t-1})} && (3.11)\\
&= \frac{p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{y}_{0:t-1})p(\boldsymbol{x}_t|\boldsymbol{y}_{0:t-1})}{\int p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{y}_{0:t-1})p(\boldsymbol{x}_t|\boldsymbol{y}_{0:t-1})d\boldsymbol{x}_t} && (3.12)\\
&\propto p(\boldsymbol{y}_t|\boldsymbol{x}_t)p(\boldsymbol{x}_t|\boldsymbol{y}_{0:t-1}). && (3.13)
\end{aligned}$$

Also we have

$$p(\boldsymbol{x}_t|\boldsymbol{y}_{0:t-1}) = \int p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{0:t-1})d\boldsymbol{x}_{t-1} \qquad (3.14)$$

where $p(\boldsymbol{x}_t|\boldsymbol{y}_{0:t-1})$ is termed as the predictive density, $p(z_t|\boldsymbol{x}_t)$ as the likelihood and $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ as the prior density.

Another important distribution is the posterior distribution $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{0:t})$.

The posterior distribution can be written as

$$p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{0:t}) = \frac{p(\boldsymbol{y}_t|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{0:t-1})p(\boldsymbol{x}_t|\boldsymbol{y}_{0:t-1})}{p(\boldsymbol{y}_t|\boldsymbol{y}_{0:t-1})} \tag{3.15}$$

$$= \frac{p(\boldsymbol{y}_t|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{0:t-1})p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{y}_{0:t-1})p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{0:t-1})}{p(\boldsymbol{y}_t|\boldsymbol{y}_{0:t-1})} \tag{3.16}$$

$$\propto p(\boldsymbol{y}_t|\boldsymbol{x}_{0:t})p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{0:t-1}). \tag{3.17}$$

The filtering density is a marginal of the posterior density. It is worth noting the following points regarding the calculation of the filtering and posterior densities.

- In a Bayesian framework, all information about the states is contained in these densities and thus obtaining this density is complete solution to the filtering problem.

- The application of Baye's rule to (3.10) yields (3.11), and (3.15) yields (3.16). The combining of the likelihoods and the prior densities leads to calculation of the posterior densities which is the essence of the Bayesian Methodology.

- These densities can be recursively calculated i.e., the filtering density $p(\boldsymbol{x}_t|\boldsymbol{y}_{0:t})$ can be obtained when one has complete knowledge of $p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{0:t-1})$, the likelihood $p(\boldsymbol{y}_t|\boldsymbol{x}_t)$, and the prior distribution $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$. The situation is the same with the posterior density. Thus, a framework for an online and realtime processing of these densities is available.

- There are essentially two stages in the computation of these densities, a predictive and an update stage. In the predictive stage, the density $p(\boldsymbol{x}_t|\boldsymbol{y}_{0:t-1})$ is obtained from the filtering density at time instant $t-1$, i.e., $p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{0:t-1})$.

Using the latest measurements, these predictive densities are updated to obtain the filtering densities, $p(\boldsymbol{x}_t|\boldsymbol{y}_{0:t})$ and $p(\boldsymbol{x}_t|\boldsymbol{y}_{0:t})$.

- Estimates of $\boldsymbol{x}_t$, which maximize these densities are known as maximum a posteriori estimates. If the estimate is given by

$$\hat{\boldsymbol{x}}_t = E[\boldsymbol{x}_t|\boldsymbol{y}_{0:t}] = \int \boldsymbol{x}_t p(\boldsymbol{x}_t|\boldsymbol{y}_{0:t}) d\boldsymbol{x}_t \tag{3.18}$$

  it is known as minimum mean square error(MMSE).

Thus, one is motivated to obtain these filtering densities sequentially in real-time. The prior and the likelihood densities can be obtained using equations (3.8) and (3.9). In situations where $\{\mathbf{v}_t\}$ and $\{\boldsymbol{u}_t\}$ are Gaussian processes and the functions $f(\cdot)$ and $g(\cdot)$ are linear, the Kalman filter presents the optimal solution. However when the noise processes $\{\mathbf{v}_t\}, \{\boldsymbol{u}_t\}$ are non-Gaussian and the functions $g(\cdot)$ and $h(\cdot)$ are non-linear, analytically tractable results are often hard to obtain. In the remainder of this chapter and the thesis we will only consider filters which address these later scenarios. Also we classify the various filters as Monte Carlo-based and non-Monte Carlo-based filters.

3.5 Non-Monte Carlo-based filters

As stated earlier, most recursive solutions to the filtering problem involve two key operations at each time instant: (a) propagation of the state estimate from the previous time instant to the current time instant and (b) updating of the state estimate using the current measurements. Two popular non-Monte Carlo filters that address the filtering problem when the state space models are non-linear are

- the Extended Kalman filter (EKF) and
- the Unscented Kalman filter (UKF)

We now briefly summarize these filters. They provide approximate solutions to the filtering problem when the functions in the DSS model are nonlinear. In these filters the following approximations are made: the predictive density of the state is approximated by a Gaussian, $p(\boldsymbol{x}_t \,|\, \boldsymbol{y}_{1:t-1}) \approx \mathcal{N}(\bar{\boldsymbol{x}}_{t|t-1}, \mathbf{P}_{t|t-1})$, where $\bar{\boldsymbol{x}}_{t|t-1}$ and $\mathbf{P}_{t|t-1}$ are the predictive mean and covariance matrix of $\boldsymbol{x}_t$ given $\boldsymbol{y}_{1:t-1}$, and the filtering density is approximated by another Gaussian, $p(\boldsymbol{x}_t \,|\, \boldsymbol{y}_{1:t}) \approx \mathcal{N}(\bar{\boldsymbol{x}}_{t|t}, \mathbf{P}_{t|t})$ where $\bar{\boldsymbol{x}}_{t|t}$ and $\mathbf{P}_{t|t}$ are the mean and covariance matrix of $\boldsymbol{x}_t$ given $\boldsymbol{y}_{1:t}$.

3.5.1 Extended Kalman Filter (EKF)

In the EKF, the transition and measurement functions of the DSS model are linearized through a Taylor series expansion, and the DSS equations are transformed into a model with linear state and measurement functions for which the Kalman filter presents an optimal solution [23]. Given $\mathbf{P}_{0|0} = E(\boldsymbol{x}_0 \boldsymbol{x}_0^\top)$ and using the Gaussian approximation of the predictive and filtering distributions at time instant $t-1$, the two steps in the EKF at time instant t are

1. Time update step:

$$\begin{aligned}\bar{\boldsymbol{x}}_{t|t-1} &= g(\bar{\boldsymbol{x}}_{t|t-1}) \\ \mathbf{P}_{t|t-1} &= \mathbf{C}_{u,t} + \mathbf{G}_{t-1}\mathbf{P}_{t-1|t-1}\mathbf{G}_{t-1}^\top.\end{aligned} \tag{3.19}$$

2. Measurement update:

$$\begin{aligned}\bar{\boldsymbol{x}}_{t|t} &= \bar{\boldsymbol{x}}_{t|t-1} + \mathbf{K}_t(\mathbf{y}_t - h(\bar{\boldsymbol{x}}_{t|t-1})) \\ \mathbf{P}_{t|t} &= \mathbf{P}_{t|t-1} - \mathbf{K}_t\mathbf{H}_t\mathbf{P}_{t|t-1} \end{aligned} \tag{3.20}$$

where $\mathbf{K}_t = \mathbf{P}_{t|t-1}\mathbf{H}_t^\top\mathbf{S}_t^{-1}$, $\mathbf{S}_t = \mathbf{H}_t\mathbf{P}_{t|t-1}\mathbf{H}_t^\top + \mathbf{C}_{v,t}$, $\mathbf{H}_t = \frac{\partial h(\boldsymbol{x})}{\partial \boldsymbol{x}}_{\boldsymbol{x}=\bar{\boldsymbol{x}}_{t|t-1}}$, and $\mathbf{G}_t = \frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}}_{\boldsymbol{x}=\bar{\boldsymbol{x}}_{t|t}}$.

3.5.2 Unscented Kalman Filter (UKF)

In the UKF, a set of deterministically chosen points known as sigma points $\boldsymbol{\Omega}_t^s$ are used in approximating the filtering and predictive distributions by Gaussian distributions. In this filter, the state of the system $\boldsymbol{x}_t$ is represented by concatenating it with the process and measurement noise states, i.e., by forming $\boldsymbol{x}_t^s = [\boldsymbol{x}_t^\top, \mathbf{u}_t^\top, \mathbf{v}_t^\top]^\top$. The main steps in the UKF are

1. Calculation of sigma points: $\boldsymbol{\Omega}_{t-1}^s = [\boldsymbol{x}_t^s, \boldsymbol{x}_t^s \pm \sqrt{(n_s+\lambda)\mathbf{P}_{t-1}^s)}]$ where n_s is the dimension of $\boldsymbol{x}_t^s$ and λ is a scaling parameter, and $\mathbf{P}_{t-1}^s$ is the covariance matrix of $\boldsymbol{x}_{t-1}^s$.

2. Time update step: Using (3.8), the sigma points $\boldsymbol{\Omega}_{t-1}^s$ are propagated to obtain $\boldsymbol{\Omega}_{t\,|\,t-1}^s$ which are then appropriately weighted. Using $\boldsymbol{\Omega}_{t\,|\,t-1}^s$ and the weights, the mean $\boldsymbol{x}_{t|t-1}$ and covariance matrix $\mathbf{P}_{t|t-1}$ of the predictive distribution are easily obtained. The predictive sigma points $\boldsymbol{\Omega}_{t\,|\,t-1}^s$ are transformed through the observation model (3.9) to obtain a new set of measurement points $\mathcal{Y}_{t-1}$ and their predictive measurement mean $\bar{\boldsymbol{y}}_{t|t-1}$.

3. Measurement update step: Using the points computed in the time update step, the covariance matrix $\mathbf{P}_{\boldsymbol{y}_t \boldsymbol{y}_t}$ of $\mathcal{Y}_{t-1}$ and the cross covariance matrix $\mathbf{P}_{\boldsymbol{\Omega}_t \boldsymbol{y}_t}$ between $\boldsymbol{\Omega}^s_{t\,|\,t-1}$ and $\mathcal{Y}_{t-1}$ are computed. The parameters of the filtering density are obtained as follows:

$$
\begin{aligned}
\mathbf{K}_t &= \mathbf{P}_{\boldsymbol{\Omega}_t \boldsymbol{y}_t} \mathbf{P}^{-1}_{\boldsymbol{y}_t \boldsymbol{y}_t} \\
\bar{\boldsymbol{x}}_{t|t} &= \bar{\boldsymbol{x}}_{t|t-1} + \mathbf{K}_t(\mathbf{y}_t - \bar{\boldsymbol{y}}_{t|t-1}) \\
\mathbf{P}_{t|t} &= \mathbf{P}_{t|t-1} - \mathbf{K}_t \mathbf{P}_{\boldsymbol{y}_t \boldsymbol{y}_t} \mathbf{K}_t^\top
\end{aligned}
\qquad (3.21)
$$

More details about the expressions of the UKF can be found in [24], [25].

3.6 Sequential Monte Carlo-based Filters

In a Monte Carlo framework the posterior density $p(\boldsymbol{x}_t|\boldsymbol{y}_{0:t})$ is represented by a random measure, $\Xi_t = \{\boldsymbol{x}_t^{(m)}, w_t^{(m)}\}$, which is a set of samples or support points $\{\boldsymbol{x}_t^{(m)}\}$ with weights $\{w_t^{(m)}\}$ with $m = 1, 2, \cdots M$ and M is the total number of samples. These support points or sample points are also referred to as particles and hence the method particle filtering [26], [27].

3.6.1 Sequential Importance Sampling (SIS)

These filters exploit the Importance Sampling principle of Monte Carlo methods. A Monte Carlo approximation of the posterior distribution is

$$\begin{aligned} p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{0:t}) &\approx \sum_{m=1}^{M} w_{0:t}^{(m)} \delta\Big(\boldsymbol{x}_{0:t} - \boldsymbol{x}_{0:t}^{(m)}\Big) \\ w_{0:t}^{(m)} &\propto \frac{p(\boldsymbol{x}_{0:t}^{(m)}|\boldsymbol{y}_{0:t})}{\pi(\boldsymbol{x}_{0:t}^{(m)}|\boldsymbol{y}_{0:t})} \end{aligned} \tag{3.22}$$

where $\pi(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{0:t})$ is the proposal density function. When the proposal function has the following recursive form

$$\pi(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{0:t}) \propto \pi(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{y}_{0:t})\pi(\boldsymbol{x}_{0:t-1}|\boldsymbol{y}_{0:t-1}) \tag{3.23}$$

then

$$w_t^{(m)} = w_{t-1}^{(m)} \frac{p(\boldsymbol{y}_t|\mathbf{x}_t^{(m)})p(\boldsymbol{x}_t^{(m)}|\boldsymbol{x}_{t-1}^{(m)})}{\pi(\boldsymbol{x}_t^{(m)}|\boldsymbol{x}_{t-1}^{(m)}, \boldsymbol{y}_{0:t})} \tag{3.24}$$

through equations (3.17) and (3.22). Therefore equation (3.24) is also known as the weight update equation and the filtering density is thereby approximated as

$$p(\boldsymbol{x}_t|\boldsymbol{y}_{0:t}) = \sum_{m=1}^{M} w_t^{(m)} \delta\Big(\boldsymbol{x}_t - \boldsymbol{x}_t^{(m)}\Big). \tag{3.25}$$

3.6.2 Issues Related to SIS

- *Degeneracy of the algorithm:* A commonly observed phenomenon is that the variance of the weights increases with time, which means that all but a few particles are left which have significant weights. Thus, we are left with few particles that have important weights. This is commonly known as the *degeneracy phenomenon.* A significant amount of computation time is wasted with these particles whose contribution is almost zero. To avoid these problems the particles are resampled i.e., the mth particle is copied N_m times where N_m is distributed according to a multinomial distribution with parameters $\{w^{(1)})_t, w_t^{(2)}, \cdots, w_t^{(M)}\}$. This ensures that particles with low importance weights are eliminated and particles with sufficient weights are allowed to propagate and multiply. There are several schemes for resampling of which the popular ones are the ones introduced in [26] and [28]. The other resampling methods are systematic resampling, stratified resampling, random resampling and residual resampling [29].

- *Choice of Importance Function* The optimal choice of the importance function is that which minimizes the variance of the weights which is given as

$$\pi(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{y}_{0:t}) = p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{y}_t). \tag{3.26}$$

However the computation of the optimal importance is analytically possible only for a class of DSS models. Even if there exists an analytical model, obtaining samples from this distribution is in general an intensive task. A simple and elegant choice of importance function is the prior distribution which is readily available from the state equation (3.8). Some of the advantages in using this importance function is the ease of drawing samples and the ease in updating the weights.

Although resampling is an important step in removing the degeneracy phenomenon, repeated stages of resampling reduce the diversity in the particles. This is termed as *sample impoverishment*. A major cause of this sample impoverishment is often observed to be the choice of importance function. When the importance function (in many cases the predictive or prior distribution) differs considerably from the target distribution then many of the particles drawn using this importance function get eliminated under resampling. There have been schemes to enhance particle diversity [30] which after the resampling stage introduce a Markov chain move which drifts the particle to a new random position in the parameter space with the constraint that the transition kernel preserves stationarity with respect to the target posterior density. Other schemes which address this problem of sample impoverishment exist which impose a criterion for resampling at a particular time instant t .

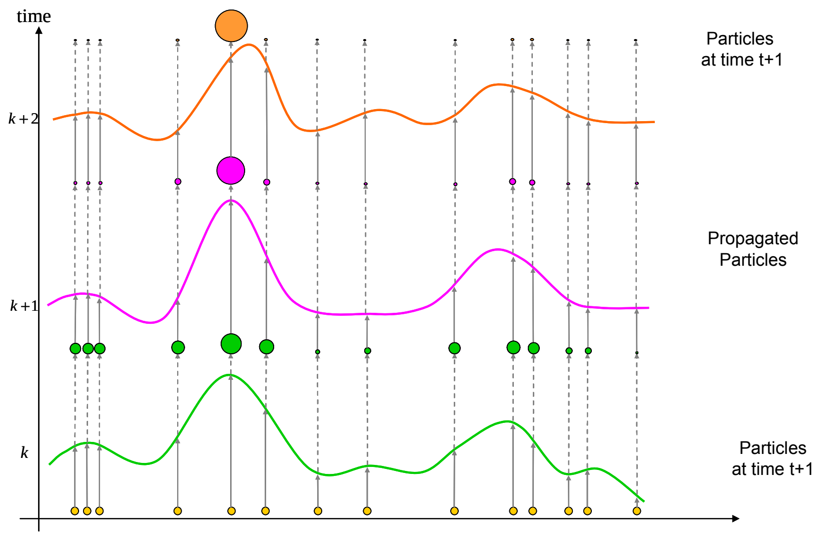


Figure 3.1: Evolution of the Filtering Density without any resampling.

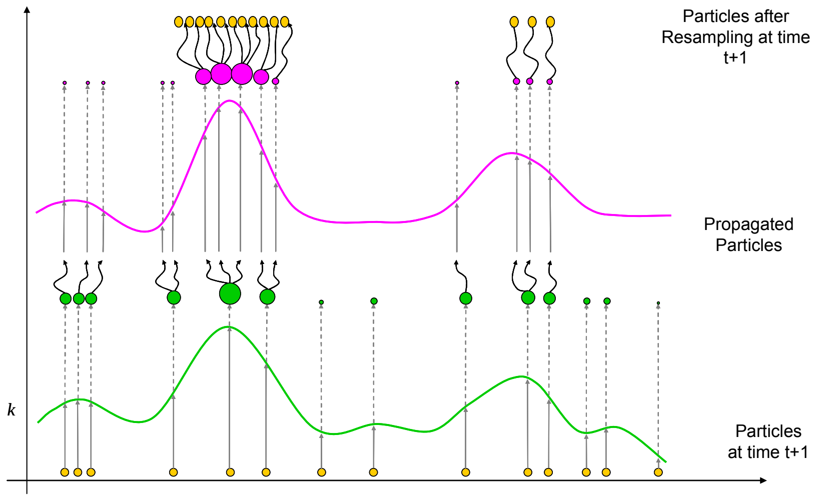


Figure 3.2: Evolution of the Filtering Density with resampling.

3.6.3 Standard particle filtering (SPF)

SPF, methods are an extension of SIS schemes with a resampling step which allow for the recursive approximation of the posterior density of the unknown state by $p(\boldsymbol{x}_{0:t}, |\ \boldsymbol{y}_{1:t}) \approx \sum_{m=1}^{M} \delta\left(\boldsymbol{x}_{0:t} - \boldsymbol{x}_{0:t}^{(m)}\right) w_t^{(m)}$, where $\delta(\cdot)$ denotes the Dirac delta function [30]. At time instant t, the algorithm updates the random measure $bfXi_{t-1} = \{\boldsymbol{x}_{0:t-1}^{(m)}, w_{t-1}^{(m)}\}_{m=1}^{M}$ to $\boldsymbol{\Xi}_t = \{\boldsymbol{x}_{0:t}^{(m)}, w_t^{(m)}\}_{m=1}^{M}$ by three main steps:

1. *Particle generation:* New particles are drawn from a proposal distribution function $\pi(\cdot)$ i.e., $\boldsymbol{x}_t^{(m)} \sim \pi(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}^{(m)}, \mathbf{y}_{1:t})$.

2. *Weight update:* Upon reception of the measurement $\mathbf{y}_t$, the weights are updated as

$$\tilde{w}_t^{(m)} = w_{t-1}^{(m)} \frac{p(\mathbf{y}_t \mid \boldsymbol{x}_{0:t}^{(m)}, \mathbf{y}_{1:t-1}) p(\boldsymbol{x}_t^{(m)} \mid \boldsymbol{x}_{t-1}^{(m)})}{\pi(\boldsymbol{x}_t^{(m)} \mid \boldsymbol{x}_{t-1}^{(m)}, \mathbf{y}_{1:t})}$$

   and normalized such that $\sum_{m=1}^{M} w_t^{(m)} = 1$.

3. *Resampling:* Drawing of samples from the multinomial distribution with parameters $\{w_t^{(1)}, \cdots, w_t^{(M)}\}$.

Example 1

We now consider the following non-linear time series model [31]:

$$\begin{aligned} x_t &= \frac{1}{2}x_{t-1} + 25\frac{x_{t-1}}{1+x_{t-1}^2} + 8\cos(1.2t) + v_t \\ y_t &= \frac{x_t^2}{20} + w_t \end{aligned}$$

where $x_0, \sim \mathcal{N}(0,5)$ and $v_t \sim \mathcal{N}(0,\sigma_v^2)$ and w_t $,\sim \mathcal{N}(0,\sigma_w^2)$ with $\sigma_v^2 = 10$ and $\sigma_w^2 = 1$. As can be seen, the state space equations are highly non-linear and therefore the Kalman filter and its variants are not suitable for accurate results. A particle filtering based solution is as follows:

1. Initialization: The particles $x_0^{(m)}$ are initially drawn from an initial distribution $\mathcal{N}(0,5)$ and the weights of the particles are set to $\frac{1}{M}$.

2. New particle generation:

The new particles are drawn from $\mathcal{N}(\mu_t^{(m)}, \sigma_v^2)$ with

$$\begin{aligned} x_t^{(m)} &\sim \mathcal{N}(\mu_t^{(m)}, \sigma_v^2) \\ \mu_t^{(m)} &= \frac{1}{2}x_{t-1}^{(m)} + 25\frac{x_{t-1}^{(m)}}{1 + x_{t-1}^{(m)\,2}} + 8cos(1.2t). \end{aligned}$$

3. Weight computation: The weight update proceeds according to

$$w_t^{(m)} \propto w_{t-1}^{(m)}\, p(y_t|x_t^{(m)}).$$

When these weights are normalized, one can use $\{x_t^{(m)}, w_t^{(m)}\}$ to compute estimates of the unknown states. For example, the MMSE estimate is obtained from

$$\hat{x}_t = \sum_{m=1}^{M} w_t^{(m)} x_t^{(m)}.$$

Upon completion of the estimation, the particles may be resampled. The resampling can be performed at every time instant t or according to a predefined schedule. In Figure 3.3, resampling is not performed while in 3.4, resampling is performed at every step. It can be clearly seen that with resampling the particles with negligible weights are replaced with particles which has larger weights. The sample impoverishment phenomenon can also be observed. For sake of clarity in the figures, the filtering densities are plotted for every 5s only.

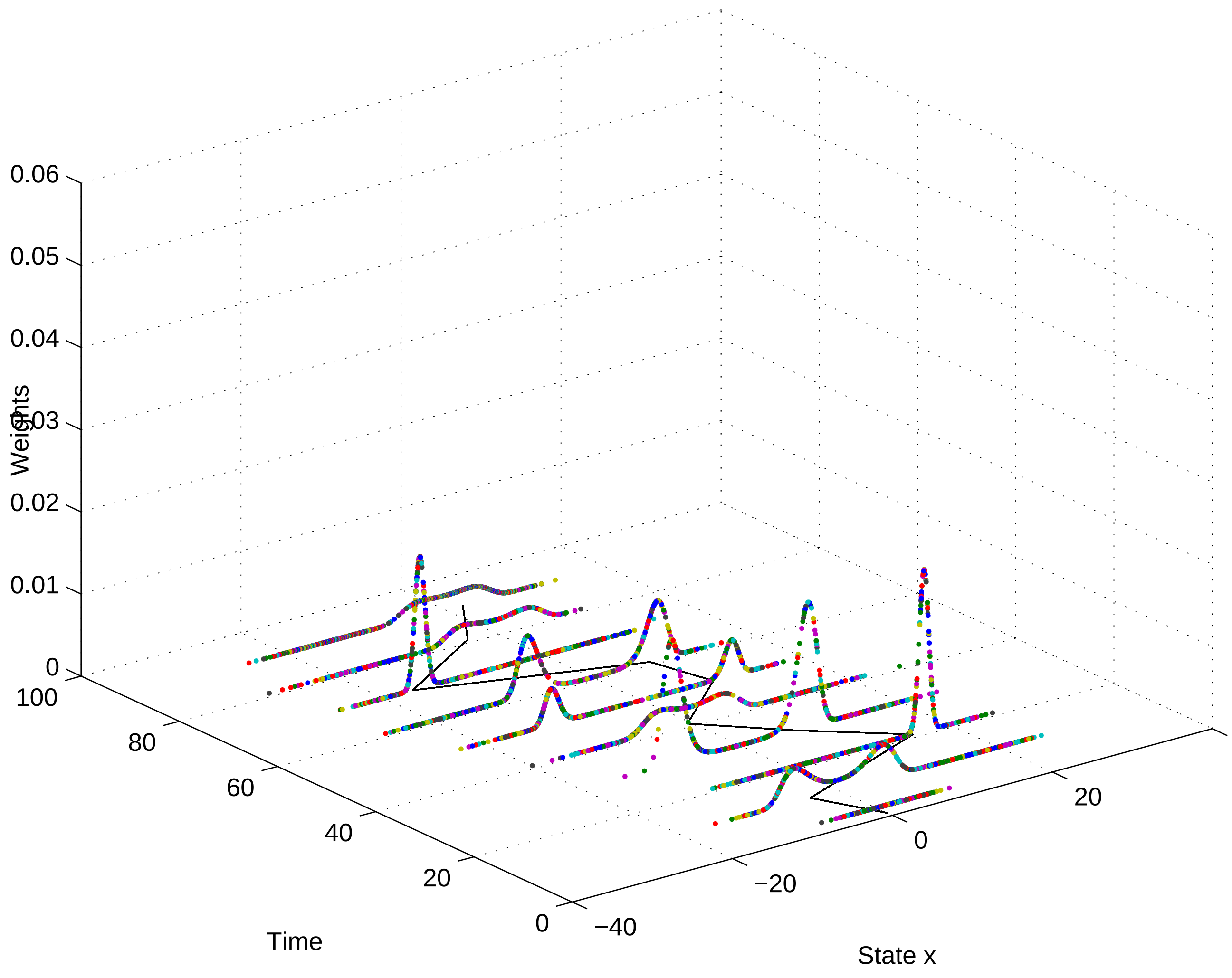


Figure 3.3: Evolution of the filtering density without resampling.The dark line is the true state while the dots represent the filtering density. Note the presence of a large number of particles whose weights are almost 0.

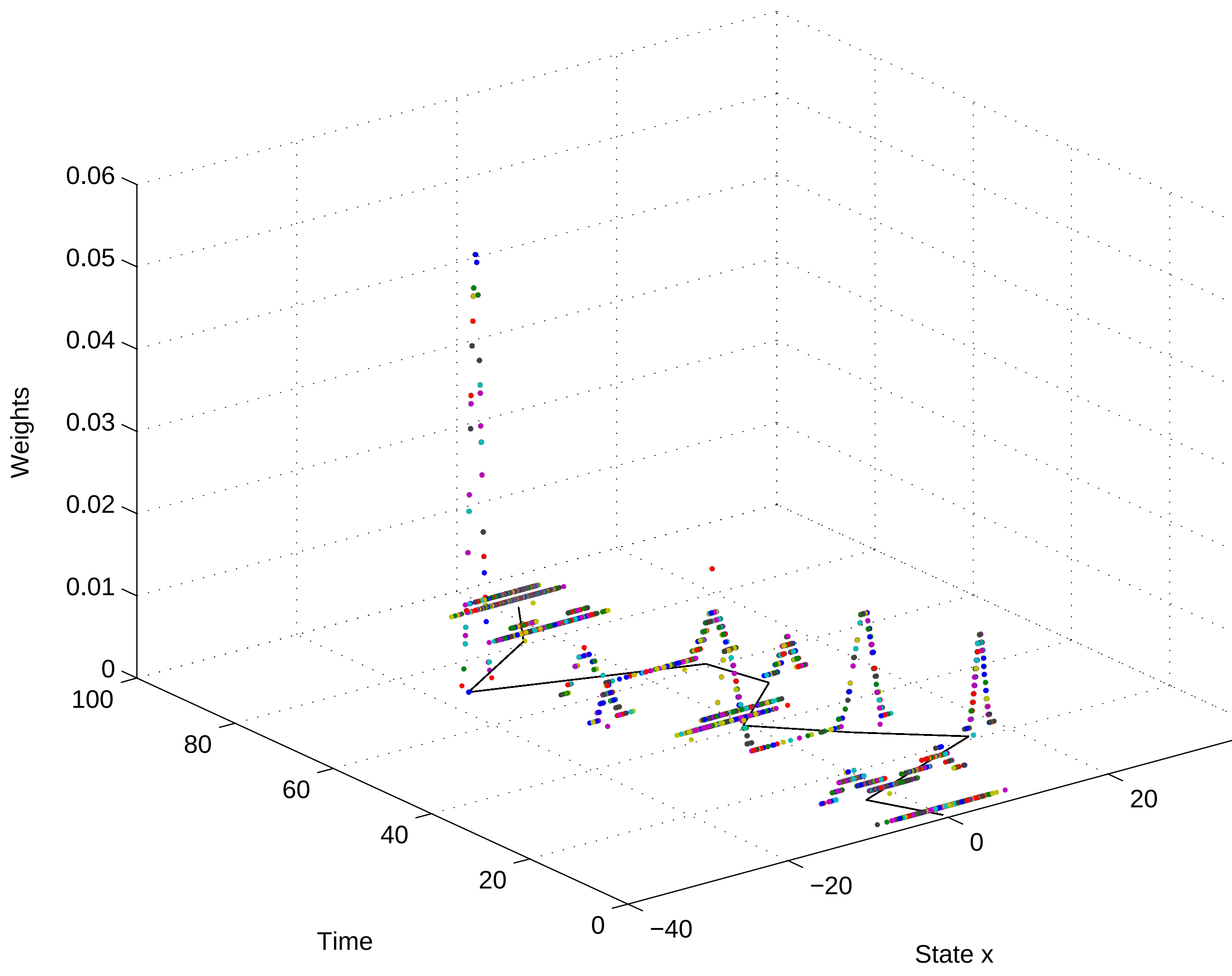


Figure 3.4: Evolution of the filtering density with resampling. The dark line is the true state. Note the absence of particle with negligible weights.

3.6.4 Auxiliary Particle Filtering (APF)

The APF attempts to draw samples from a joint distribution $p(\boldsymbol{x}_t, i \,|\, \boldsymbol{y}_{0:t})$ of the state $\boldsymbol{x}_t$ and an index i. These indices known as auxiliary variables aid in the propagation of particles that have large predictive likelihoods.

To that end, the selection of most promising particles is carried out by sampling from a multinomial distribution where the number of possible outcomes is M and the probabilities of the respective outcomes are $\tilde{w}_t^{(m)}$, $m = 1, 2, \cdots, M$.

At the beginning, the initial set of particles $\boldsymbol{x}_0^{(m)}$, $m = 1, 2, \cdots, M$, are drawn from a prior distribution $\pi(\boldsymbol{x}_0)$, and the weights of the particles are set to $\frac{1}{M}$. At time instant $t-1$, we have the random measure $\chi_{t-1} = \{\boldsymbol{x}_{0:t-1}^{(m)}, w_{t-1}^{(m)}\}_{m=1}^{M}$. The steps of the APF are then implemented as follows:

1. *Selection of most promising particle streams:* For selection of the most promising particles, we choose the conditional mean of $\boldsymbol{x}_t^{(m)}$ given $\boldsymbol{x}_{t-1}^{(m)}$ as a characterizing parameter of every stream, i.e.,

$$\boldsymbol{\mu}_t^{(m)} = E\big(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}^{(m)}\big). \tag{3.27}$$

The conditional means are computed readily from the state equation (3.8)

$$\boldsymbol{\mu}_t^{(m)} = g(\boldsymbol{x}_{t-1}^{(m)}) \tag{3.28}$$

and the weights associated with these means are calculated as

$$\tilde{w}_t^{(m)} \propto p(\boldsymbol{y}_t \mid \boldsymbol{\mu}_t^{(m)}) w_{t-1}^{(m)}. \tag{3.29}$$

A probability mass function (pmf) represented by these normalized weights is then obtained and a set of indices, the auxiliary variables $\{i_m\}$ are drawn from this pmf.

2. *New particle generation*: Using the indices and the prior state equation (**??**), one method for generating new particles is as

$$\boldsymbol{x}_t^{(m)} \sim p(\boldsymbol{x}_t^{(m)} \,|\, \boldsymbol{x}_{t-1}^{(i_m)}).$$

3. *Weight computation*:

   The newly generated particles are assigned weights according to

$$w_t^{(m)} \propto \frac{p(\boldsymbol{y}_t \mid \boldsymbol{x}_t^{(m)})}{p(\boldsymbol{y}_t \mid \boldsymbol{\mu}_t^{(i_m)})}.$$

   The likelihood terms of the numerator and denominator are calculated using the likelihood model of the state equation (3.9).

4. *State estimation*: Once the weights are normalized, one can use χ_t to compute estimates of the unknown states. The MMSE estimate is obtained from

$$\hat{\boldsymbol{x}}_t = \sum_{m=1}^{M} w_t^{(m)} \boldsymbol{x}_t^{(m)}. \tag{3.30}$$

3.6.5 Gaussian Particle Filtering (GPF)

The GPF is a SMC filtering algorithm that approximates the predictive and posterior distributions with Gaussian distributions [32]. The main steps of this filter are

1. *Generation of particles*: Samples $\boldsymbol{x}_{t-1}^{(m)}$ are drawn from $\mathcal{N}(\bar{\boldsymbol{x}}_{t-1|t-1}, \mathbf{P}_{t-1|t-1})$ and the samples $\boldsymbol{x}_t^{(m)}$ are drawn from $p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}^{(m)})$.

2. *Computation of the mean and covariance of the predictive density*: Using the samples of the previous step, the sample mean $\bar{\boldsymbol{x}}_{t|t-1}$ and covariance matrix $\mathbf{P}_{t|t-1}$ are obtained.

$$\bar{\boldsymbol{x}}_{t|t-1} = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{x}_t^{(m)} \tag{3.31}$$

$$\mathbf{P}_{t|t-1} = \frac{1}{M}\sum_{m=1}^{M} (\boldsymbol{x}_t^{(m)} - \bar{\boldsymbol{x}}_{t|t-1})(\boldsymbol{x}_t^{(m)} - \bar{\boldsymbol{x}}_{t|t-1})^\top \tag{3.32}$$

3. *Computation of the weights*: Using the measurement $\mathbf{y}_t$, the weights are calculated by

$$\tilde{w}_t^{(m)} = \frac{p(\mathbf{y}_t \mid \boldsymbol{x}_t^{(m)}) p(\boldsymbol{x}_t^{(m)} | \mathbf{y}_{1:t-1})}{\pi(\boldsymbol{x}_t^{(m)} \mid \boldsymbol{x}_{t-1}^{(m)}, \mathbf{y}_t)} = \frac{p(\mathbf{y}_t \mid \boldsymbol{x}_t^{(m)}) \mathcal{N}(\bar{\boldsymbol{x}}_{t|t-1}, \mathbf{P}_{t|t-1})}{p(\boldsymbol{x}_t^{(m)} \mid \boldsymbol{x}_{t-1}^{(m)}, \mathbf{y}_t)}$$

where $p(\boldsymbol{x}_t | \mathbf{y}_{1:t-1})$ is a Gaussian with a mean $\bar{\boldsymbol{x}}_{t|t-1}$ and a covariance matrix $\mathbf{P}_{t|t-1}$. The weights are subsequently normalized.

4. *Computation of the mean and covariance of the filtering density*: The sample mean $\bar{\boldsymbol{x}}_{t|t}$ and covariance $\mathbf{P}_{t|t}$ of the filtering density are obtained using the

weighted set of samples.

$$\bar{\boldsymbol{x}}_{t|t} = \frac{1}{M}\sum_{m=1}^{M} w_t^{(m)} \boldsymbol{x}_t^{(m)} \tag{3.33}$$

$$\mathbf{P}_{t|t} = \frac{1}{M}\sum_{m=1}^{M} w_t^{(m)} (\boldsymbol{x}_t^{(m)} - \bar{\boldsymbol{x}}_{t|t-1})(\boldsymbol{x}_t^{(m)} - \bar{\boldsymbol{x}}_{t|t-1})^\top \tag{3.34}$$

Summary: In this chapter we had reviewed some Sequential Monte Carlo algorithms and have noted their efficacy in dealing with non-linear filtering problems. The reference example clearly shows some of the important issues of these algorithms namely resampling and sample impoverishment. The Gaussian Particle filter, a variant of these sequential Monte Carlo methods has been introduced. Some applications of this algorithm will be seen in Chapters 5 and 6.

Chapter 4

Cost-Reference Particle Filtering

Standard particle filtering (SPF) schemes rely on the availability of statistics of the state and observation noises involved in the dynamic state space model. Cost reference particle filtering (CRPF) techniques have proven to be a viable and robust alternative in situations when distributions of these noise processes are unknown. In Section 4.2, we study the orginally proposed CRPF algorithms and in Section 4.3, we propose two novel CRPF methods which are simpler to use and less computationally intensive. The proposed algorithms are applied to tracking a single target. Computer simulations are provided in Section 4.4 which show a good performance of the proposed algorithms when compared to the original CRPF methods and a more robust behavior than the SPF algorithms.

4.1 Introduction

4.1.1 Problem Statement

Recall that the standard DSS model, defined by equations (3.8) and (3.9), is a useful tool for describing a wide variety of phenomena ranging from the dynamics of a moving target to time-varying wireless channels in communications. The objective is the online estimation of $\boldsymbol{x}_{0:t}$ given the sequence of observations $\boldsymbol{y}_{1:t}$.

When the distribution of the noise processes are known, the filtering problem reduces to the estimation of the posterior density, $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})$, or statistics of the system state given the measurements. It is well known that the Kalman filter is optimal when the noise processes are Gaussian and the functions $g(\cdot)$ and $h(\cdot)$ are linear [23]. When these conditions are not satisfied, the Kalman filter and its extensions are often suboptimal. During the last decade, SMC methods, have been intensively studied and have gained wide acceptance in addressing situations where the noise processes are not strictly Gaussian and the state functions are non-linear [30, 33].

The filtering methods, Kalman filter or SMC algorithms, rely on the assumption that the distributions of the noise processes in the state space model are available. However, in many situations this may not be the case. When the distributions of the noise processes are unknown, propagation of the state particles and calculation of the weights is not straightforward. Recently a new class of particle filters known as CRPF methods has been proposed to address the filtering problem is such scenarios [34, 35].

In [34], the issue of sequential estimation in the context of "blind" particle filtering was first addressed. The authors introduced notions of costs and risks to determine the quality of the proposed particles. Using these functions, particles were resampled and propagated in time thereby allowing recursive state estimation without any knowledge of the noise distributions of the state space models. In [35], connections between CRPF and SPF were discussed and efficient variants to the original CRPF scheme were proposed. In this chapter we propose two new CRPF techniques and study their sensitivity to various proposal functions.

4.2 Cost Reference Particle Filtering (CRPF)

The random measures are constructed by applying user-defined costs. If the random measure at time instant $t-1$ is $\boldsymbol{\zeta}_{t-1} = \left\{\boldsymbol{x}_{0:t-1}^{(m)}, c_{t-1}^{(m)}\right\}_{m=1}^{M}$, where $c_{t-1}^{(m)}$ are the costs assigned to the particles $\boldsymbol{x}_{t-1}^{(m)}$, then upon the reception of the measurement $\boldsymbol{y}_t$, ζ_{t-1} is updated to $\boldsymbol{\zeta}_t = \left\{\boldsymbol{x}_{0:t}^{(m)}, c_t^{(m)}\right\}_{m=1}^{M}$ following the APF structure and the principle of *survival of the fittest*. The main steps of this scheme are:

1. **Selection of the most promising paths:** This step resembles the resampling procedure in SPF schemes. The most promising paths are selected using risk functions defined as $r_t^{(m)} = \lambda c_{t-1}^{(m)} + \Delta r(\boldsymbol{x}_{t-1}^{(m)}|\mathbf{y}_t)$ where

$$\Delta r(\boldsymbol{x}_{t-1}^{(m)}|\mathbf{y}_t) = \left|\left|\mathbf{y}_t - h\left(g(\boldsymbol{x}_{t-1}^{(m)})\right)\right|\right|^q, \tag{4.1}$$

with λ being a forgetting factor to avoid attributing excessive importance to the past, $q > 0$ and $||\cdot||$ denoting norm of a vector. These risk functions measure the adequacy of the particles at time instant $t-1$ given the new measurement $\mathbf{y}_t$ [34]. For resampling, a probability mass function (pmf), $\hat{\pi}_{r,t}^{(m)}$, is created to allow for assignment of weights to each particle, $\hat{\pi}_{r,t}^{(m)} \propto \mu_r(r_t^{(m)})$, where $\mu_r : \mathbb{R} \rightarrow [0, +\infty)$ is a monotonically decreasing function[1]. A simple formulation of this pmf is given by

$$\mu_r(r_t^{(m)}) \propto \frac{1}{r_t^{(m)}} \tag{4.2}$$

which is modified into a proper pmf through normalization. Following resampling, a new stream $\hat{\zeta}_{t-1} = \left\{ \hat{\boldsymbol{x}}_{0:t-1}^{(m)}, \hat{c}_{t-1}^{(m)} \right\}_{m=1}^{M}$ is obtained.

2. **Particle generation:** New particles are proposed using a proposal density, $p_t(\boldsymbol{x}_t|\hat{\boldsymbol{x}}_{t-1}^{(m)})$. In [34], the authors approximate $p_t(\boldsymbol{x}_t|\hat{\boldsymbol{x}}_{t-1}^{(m)})$ as a Gaussian kernel with statistics

$$\mathbb{E}(\boldsymbol{x}_t) = f(\hat{\boldsymbol{x}}_{t-1}^{(m)}), \quad \text{and} \quad \mathrm{Cov}(\boldsymbol{x}_t) = \sigma_t^{2,(m)} \mathbf{I}_d,$$

where d is the dimension of the state and the variance $\sigma_t^{2,(m)}$ is recursively updated as

$$\begin{aligned} t \leq \tau_0 & \qquad \sigma_t^{2,(m)} = \sigma_{t-1}^{2,(m)}, \\ t > \tau_0 & \qquad \sigma_t^{2,(m)} = \tfrac{t-1}{t}\sigma_{t-1}^{2,(m)} + \tfrac{\left\| \boldsymbol{x}_t^{(m)} - f(\hat{\boldsymbol{x}}_{t-1}^{(m)}) \right\|^2}{td}, \end{aligned}$$

with τ_0 being the time instant until which the filter obtains adequate measurements for learning the statistics of the state process.

[1]High risks indicate poor predictions of the state and lower risks indicate good predictions of the state.

3. **Costs update:** Costs measure the quality of the estimated state and are recursively updated as

$$c_t^{(m)} = \lambda c_{t-1}^{(m)} + \triangle c(\boldsymbol{x}_t^{(m)}|\boldsymbol{y}_t). \tag{4.3}$$

   A possible choice of the incremental cost function is

$$\triangle c(\boldsymbol{x}_t|\mathbf{y}_t) = ||\boldsymbol{y}_t - h(\boldsymbol{x}_t)||^q. \tag{4.4}$$

   Depending upon system requirements such as robustness, other cost functions such as the Huber loss function or the fair function can be incorporated [36].

4. **State estimation:** A simple estimation scheme consists of choosing the particle with the minimum cost as the state estimate. Alternatively, one can construct another artificial pmf $\tilde{\pi}_{c,t}^{(m)} \propto \mu_c(c_t^{(m)})$, $m = 1, \ldots, M$, and obtain estimates such as the weighted mean of the particles.

4.3 New CRPF Methods

In this section we outline two new cost reference-based particle filtering schemes. The main focus of these methods is to obtain good proposal densities and thereby robust performance by adequately exploring the state space.

4.3.1 New CRPF method Type -I (CRPF-I)

As in CRPF schemes, at each time instant t, a new stream of particles and their associated costs is generated. The main steps of the algorithm are maintained but the order is changed to allow for construction of a better proposal.

1. **Propagation of new particles :** Using the particles at time instant t, we construct a proposal mixture density,

$$\begin{aligned} \hat{\boldsymbol{x}}_{t+1}^{(m)} &= f_x(\boldsymbol{x}_t^{(m)}), \\ \pi_{t+1}(\boldsymbol{x}_{t+1}) &= \frac{1}{M}\sum_{m=1}^{M} \mathcal{K}(\boldsymbol{x}_{t+1}; \hat{\boldsymbol{x}}_{t+1}^{(m)}, h^2\hat{\boldsymbol{\Sigma}}_{x,t+1}), \end{aligned}$$

   where $\mathcal{K}(\cdot)$ is a density function centered at $\hat{\boldsymbol{x}}_{t+1}$, $\hat{\boldsymbol{\Sigma}}_{x,t+1}$ is a scale parameter and h is a smoothing parameter which is chosen as a slowly decreasing function of the sample size. For Gaussian densities this component is chosen as $h = c/M^{1/(1+4d)}$ and $c = 4/(1+2d)^{1/(1+4d)}$ [37] . Particles, $\{\boldsymbol{x}_{t+1}^{(m)}\}, m = 1 \cdots M$, are thus drawn from this kernel. In our simulations we use a Gaussian kernel and thus the proposal density is approximated by a mixture Gaussian density with each density centered at $\hat{\boldsymbol{x}}_{t+1}^{(m)}$.

2. **Computation of costs:** We use quadratic ($q = 2$) functions (see equations (4.3) and (4.4)) to determine the effectiveness and the robustness of the particles given the current measurements.

3. **Updating the proposal density and resampling:** As in the original CRPF, we construct a pmf, $\hat{\pi}_{c,t+1}^{(m)}$, and assign weights using the current costs of the particles such that particles with lower costs have larger weights and vice-versa.

The covariance matrix, $\hat{\boldsymbol{\Sigma}}_{x,t+2}$, is updated as

$$\hat{\boldsymbol{\Sigma}}_{x,t+2} \approx \sum_{m=1}^{M} \hat{\pi}_{c,t+1}^{(m)} (\boldsymbol{x}_{t+1}^{(m)} - \hat{\boldsymbol{x}}_{t+1})(\boldsymbol{x}_{t+1}^{(m)} - \hat{\boldsymbol{x}}_{t+1})^{\top}, \tag{4.5}$$

where $\hat{\boldsymbol{x}}_{t+1} = \sum \pi_{c,t+1}^{(m)} \boldsymbol{x}_{t+1}^{(m)}$ is the sample mean. We now perform the resampling step and eliminate particles with poorer weights (i.e., particles with larger costs).

4. **State estimation**: The state estimation is analogous to the original CRPF schemes.

The above algorithm does not use any risk functions and that consitutes a major difference with respect to the original CRPF proposed in [34]. There, the risk function indicates the effectiveness of the state at time instant t given the new measurement $\mathbf{y}_{t+1}$ and is computed using the costs at the previous time instant. With this new method, we argue that a good proposal density accompanied by a resampling scheme can lead to a similar performance than the original CRPF with risk calculations. Therefore, the step of risk calculation can be eliminated and the computational complexity reduced. The proposed methods re pictorially represented in 4.1.

4.3.2 New CRPF method Type-II (CRPF-II)

This method is constructed in a similar way as the CRPF Type-I method. It primarily differs in the propagation of the particles and the construction of the proposal density. It is based on a learning process that leads to the acquisition of the statistics of the noise process in the state equation. Consider a stream of trajectories at time instant t which includes state particles, noise particles, and incremental and accumulated costs $\tilde{\boldsymbol{\Xi}}_t = \left\{\boldsymbol{x}_{0:t}^{(m)}, \boldsymbol{\epsilon}_{0:t}^{(m)}, \mathcal{C}_t^{(m)}, \Delta\mathcal{C}_t^{(m)}\right\}_{m=1}^{M}$. With a slight abuse of notation, we obtain a set of noise particles and associated incremental costs as $\dot{\boldsymbol{\Xi}}_t=$ $\left\{(\boldsymbol{\epsilon}_0^{(m)}, \Delta\mathcal{C}_0^{(m)}), (\boldsymbol{\epsilon}_1^{(m)}, \Delta\mathcal{C}_1^{(m)}), \cdots (\boldsymbol{\epsilon}_t^{(m)}, \Delta\mathcal{C}_t^{(m)})\right\}$. The weighted sample noise mean of this trajectory is given by

$$\boldsymbol{\mu}_{\boldsymbol{\epsilon}_t^{(m)}} = \frac{\sum_{\tau=1}^{t} \alpha_\tau^{(m)} \boldsymbol{\epsilon}_\tau^{(m)}}{\sum_{\tau=1}^{t} \alpha_\tau^{(m)}}, \tag{4.6}$$

and the weighted sample noise covariance matrix is computed by

$$\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}_t^{(m)}} = \frac{\sum_{\tau=1}^{t} \alpha_\tau^{(m)} \boldsymbol{\epsilon}_\tau^{(m)} \boldsymbol{\epsilon}_\tau^{(m)\top}}{\sum_{\tau=1}^{t} \alpha_\tau^{(m)}}, \tag{4.7}$$

with $\alpha_t^{(m)} = \frac{1}{\Delta\mathcal{C}_t^{(m)}}$. Recursive expressions for obtaining these terms are as follows

$$\begin{aligned}
\boldsymbol{\mu}_{\boldsymbol{\epsilon}_{t+1}^{(m)}} &= \left(\frac{\beta_t^{(m)}}{\beta_{t+1}^{(m)}}\right) \boldsymbol{\mu}_{\boldsymbol{\epsilon}_t^{(m)}} + \frac{\boldsymbol{\epsilon}_\tau^{(m)} \alpha_{t+1}^{(m)}}{\beta_{t+1}^{(m)}}, \\
\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}_{t+1}^{(m)}} &= \left(\frac{\beta_t^{(m)}}{\beta_{t+1}^{(m)}}\right) \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}_t^{(m)}} + \frac{\alpha_{t+1}^{(m)} \boldsymbol{\epsilon}_t^{(m)} \boldsymbol{\epsilon}_{t+1}^{(m)\top}}{\beta_{t+1}^{(m)}},
\end{aligned} \tag{4.8}$$

with $\beta_t^{(m)} = \sum_{\tau=1}^{t} \alpha_\tau^{(m)}$. The filter is displayed in 4.2

The main steps of the algorithm are:

1. **Propagation of particles:** For each stream of particles we construct an individual Gaussian proposal density $\pi_{\boldsymbol{\epsilon}_{t+1}^{(m)}} = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\epsilon}_t^{(m)}}, \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}_t^{(m)}})$. Using this proposal we draw noise samples $\boldsymbol{\epsilon}_{t+1}^{(m)} \sim \pi_{\boldsymbol{\epsilon}_{t+1}^{(m)}}$ and propagate the particles,

$$\boldsymbol{x}_{t+1}^{(m)} = f_x(\boldsymbol{x}_t^{(m)}) + \boldsymbol{\epsilon}_{t+1}^{(m)}.$$

2. **Computation of costs:** The costs are calculated as in the original CRPF (see Section 4.2.)

3. **Updating the proposal density and resampling:** As in the CRPF we construct a pmf and assign weights using the current costs of the particles. The statistics of the noise trajectories are updated using (4.8) after which we perform the resampling step.

4. **State estimation**: The state estimation is similar to the original CRPF schemes.

The new algorithms are summarized in the table in the next page.

New Cost Reference Particle Filters

- Initialization, at $t = 0$.
  - For $m = 1 \cdots M$ draw samples $\boldsymbol{x}_0^{(m)} \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$.
  - For $m = 1 \cdots M$ set $\mathcal{C}_0^{(m)} = 0$.

  For $t = 1 \cdots T$

- Propagation of Particles
  - CRPF Type -I

    $\pi_{t+1}(\boldsymbol{x}_{t+1}) = \frac{1}{M} \sum_{m=1}^{M} \mathcal{K}(\boldsymbol{x}_{t+1}; \hat{\boldsymbol{x}}_{t+1}^{(m)}, h^2 \hat{\boldsymbol{\Sigma}}_{x,t+1})$
  - CRPF Type -II

    $\pi_{t+1}(\boldsymbol{x}_{t+1}) = \mathcal{N}(f_x(\boldsymbol{x}_t^{(m)}) + \boldsymbol{\mu}_{\boldsymbol{\epsilon}_t^{(m)}}, \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}_t^{(m)}})$.

  For $m = \{1 \cdots M\}$ $\quad \boldsymbol{x}_{t+1}^{(m)} \sim \pi_{t+1}(\boldsymbol{x}_{t+1})$

- Computation of Costs For $m = \{1 \cdots M\}$
  $\mathcal{C}_{t+1}^{(m)} = \lambda \mathcal{C}_t^{(m)} + ||\boldsymbol{y}_{t+1} - f_y(\boldsymbol{x}_{t+1}^{(m)})||^q$

- Updating Proposal Density
  - CRPF Type -I

    $\hat{\boldsymbol{\Sigma}}_{x,t+1} \approx \sum_{m=1}^{M} \pi^{(m)} (\boldsymbol{x}_t^m - \hat{\boldsymbol{x}}_t)(\boldsymbol{x}_t^m - \hat{\boldsymbol{x}}_t)^\top$
  - CRPF Type -II

    $\boldsymbol{\mu}_{\boldsymbol{\epsilon}_{t+1}^{(m)}} = \left(\frac{\beta_t^{(m)}}{\beta_{t+1}^{(m)}} \right) \boldsymbol{\mu}_{\boldsymbol{\epsilon}_t^{(m)}} + \frac{\boldsymbol{\epsilon}_\tau^{(m)} \alpha_{t+1}^{(m)}}{\beta_{t+1}^{(m)}}$

    $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}_{t+1}^{(m)}} = \left(\frac{\beta_t^{(m)}}{\beta_{t+1}^{(m)}} \right) \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}_t^{(m)}} + \frac{\alpha_{t+1}^{(m)} \boldsymbol{\epsilon}_t^{(m)} \boldsymbol{\epsilon}_{t+1}^{(m)\top}}{\beta_{t+1}^{(m)}}$.

- State Estimation
  $\pi_{t+1}^{(m)} \propto \mu(\mathcal{C}_{t+1}^{(m)})$ $\qquad$ and $\hat{x}_{t+1} = \sum_m \boldsymbol{x}_{t+1}^m \pi_{t+1}^{(m)}$

- Resampling

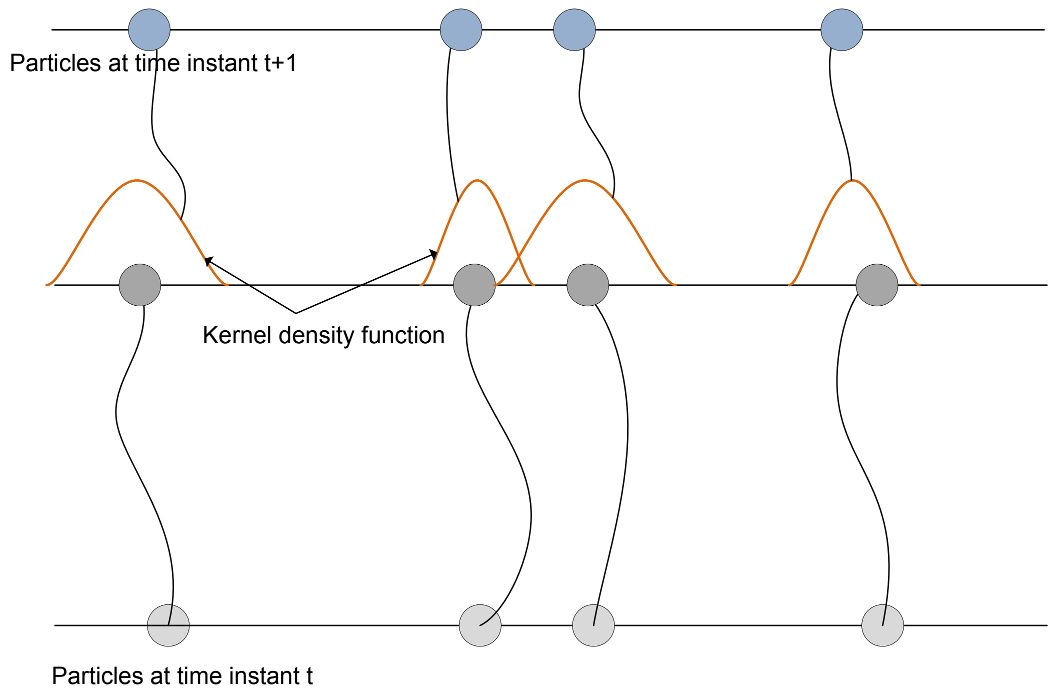


Figure 4.1: A new cost reference particle filter: CRPF Type -I

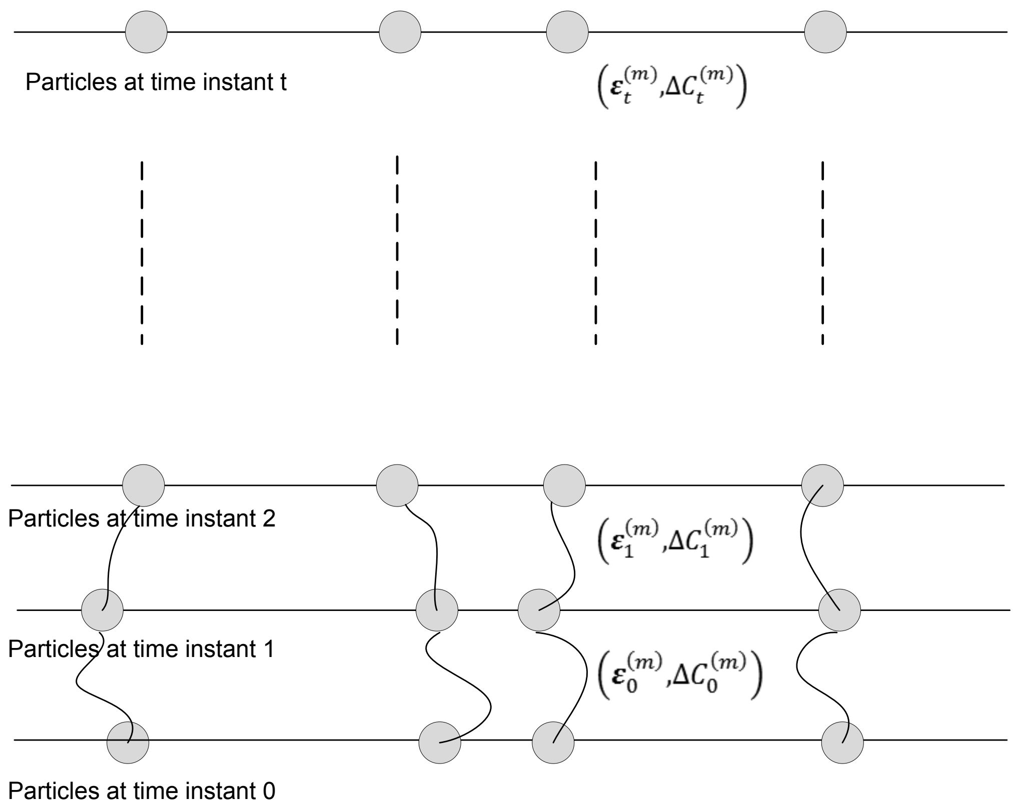

Particles at time instant t
$\left(\boldsymbol{\varepsilon}_t^{(m)}, \Delta C_t^{(m)}\right)$
Particles at time instant 2
$\left(\boldsymbol{\varepsilon}_1^{(m)}, \Delta C_1^{(m)}\right)$
Particles at time instant 1
$\left(\boldsymbol{\varepsilon}_0^{(m)}, \Delta C_0^{(m)}\right)$
Particles at time instant 0

Figure 4.2: A new cost reference particle filter: CRPF Type -II

4.4 Computer Simulations

In this section we present some simulation results to illustrate the performance of the proposed algorithms for target tracking in a wireless sensor network.

4.4.1 The target tracking problem

Consider a two-dimensional sensing field with one moving target. The target motion model under constant acceleration is given by

$$\boldsymbol{x}_t = \mathbf{G}_x \boldsymbol{x}_{t-1} + \boldsymbol{\Gamma}_u \mathbf{u}_t, \tag{4.9}$$

where $\boldsymbol{x}_t = [x_{1,t}, x_{2,t},\ \dot{x_{1,t}},\ \dot{x_{2,t}}]^\top \in \mathbb{R}^4$ represents the position and velocity of the target, $\mathbf{u}_t$ represents a Gaussian noise with covariance matrix $\mathbf{C}_u$ that accounts for acceleration perturbances in the system, and $\mathbf{G}_x$ and $\boldsymbol{\Gamma}_u$ are known transition matrices of the form

$$\mathbf{G}_\mathbf{x} = \begin{pmatrix} 1 & 0 & T_s & 0 \\ 0 & 1 & 0 & T_s \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \boldsymbol{\Gamma}_u = \begin{pmatrix} \frac{T_s^2}{2} & 0 \\ 0 & \frac{T_s^2}{2} \\ T_s & 0 \\ 0 & T_s \end{pmatrix}.$$

The sensors measure the signal emitted by the target every T_s seconds. Following [38] we model the signal received by the n-th sensor as

$$y_t^n = \Psi_0 - 10\gamma \log_{10}\left(\frac{|\mathbf{l}_t - \mathbf{s}^n|}{d_0}\right) + v_t^n \tag{4.10}$$

where Ψ_0 is an unknown parameter that represents the signal strength at a known reference distance d_0, $\mathbf{s}^n \in \mathbb{R}^2$ is the position of the n-th sensor, $\mathbf{l}_t \in \mathbb{R}^2$ denotes the location of the target at time t, γ is an attenuation parameter due to the communication channel, and v_t^n is observation noise. The problem of estimating the target trajectory at every time instant is now posed as a filtering problem, i.e., estimation of the state $\boldsymbol{x}_{0:t}$ given the measurements $y_{1:t}$. Note that in this context, the state of the system includes not only the target dynamic parameters but also the static parameter Ψ_0[2].

4.4.2 Simulation parameters and algorithms

PARAMETER	VALUE
Sensing field dimensions	$650m \times 750m$
Number of sensors	4
Reference power Ψ_0	-50dB
Attenuation parameter γ	2.5
Number of particles M	2000
Sampling period T_s	1s
Total observation period T	60s
Forgetting factor λ	0.95

Table 4.1: Parameters of the system and algorithms

We considered an experiment with a target moving along the sensor field for one minute with samples taken every $T_s = 1$ second. The initial state, $\boldsymbol{x}_0$, was drawn from $\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ with $\boldsymbol{\mu}_0 = [0.0\ 0.0\ 5.0\ 5.0]^\top$ and $\boldsymbol{\Sigma}_0 = diag\ (50\ 50\ 5\ 5)$. The state and observation noises were modeled as $\mathbf{u}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_u)$ with $\mathbf{C}_u = diag(0.1, 0.2)$ and $v_t^n \sim \mathcal{N}(0, 1)$, respectively.

We placed four static sensors in the sensing field at locations $(100, 400)$, $(300, 300)$, $(200, 0)$ and $(0, 250)$. The sensors collected the measurements according

[2]Estimation of the static parameter Ψ_0 was addressed in [39].

to equation (4.10) and transmitted them to the fusion center, which performed the tracking according to the previously explained algorithms (labeled in the Figures as CRPF-I and CRPF-II). For comparision purposes we also considered the original CRPF algorithm proposed in [34] (labeled CRPF). All the CRPF methods were initialized by drawing particles from the initial distribution $\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$. Other parameter' values of the system and the algorithms are displayed in Table 4.1.

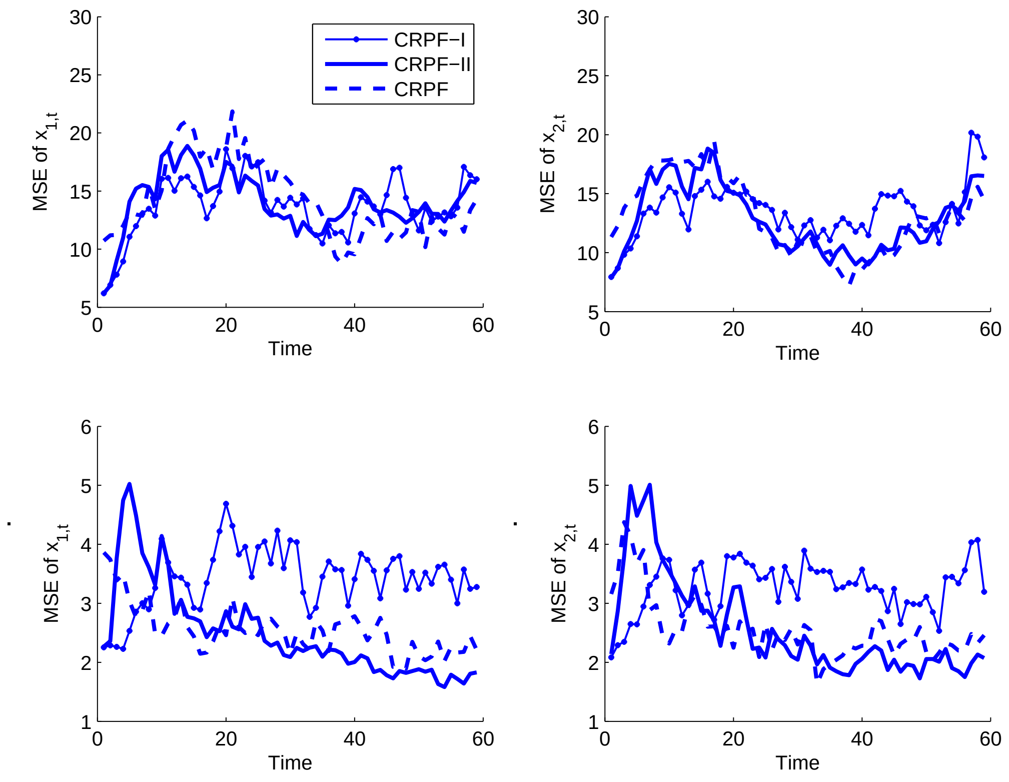


Figure 4.3: MSE of the target dynamics for various CRPF algorithms

4.4.3 Results

Figures 4.3 and 4.4 show the mean square errors (MSEs) of the targets dynamics and the unknown static parameter Ψ_0.The MSE plots were obtained by averaging over 50 trajectory runs. It can be seen that the performance of the three algorithms is very similar for the considered scenario. Figure 4.5 depicts one realization of the target trajectory and the obtained estimates using the original CRPF algorithms and the CRPF methods Type-I and II.

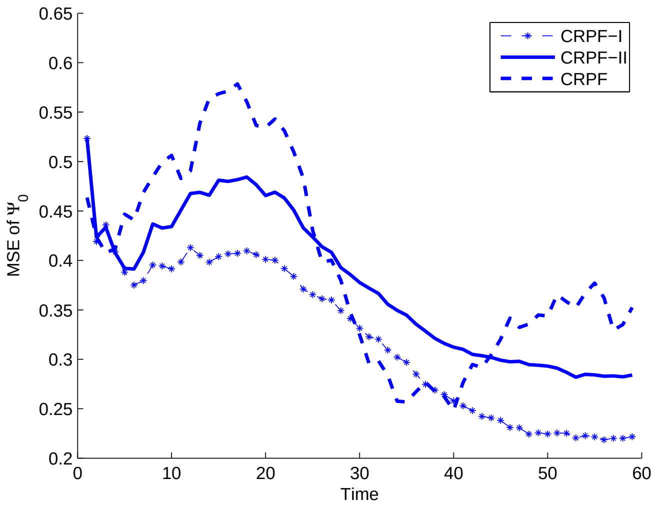


Figure 4.4: MSE of the static parameter Ψ_0 for various CRPF algorithms

4.4.4 Robustness of the methods

For the previous experiment we considered the state noise to be Gaussian. In this second experiment we considered the state noise to be a mixture Gaussian density,

$$\mathbf{u}_t \sim 0.5\mathcal{N}(\mathbf{0}, \mathbf{C}_{u,1}) + 0.5\mathcal{N}(\mathbf{0}, \mathbf{C}_{u,2})$$

$\mathbf{C}_{u,1} = diag(0.10.2)$ and $\mathbf{C}_{u,1} = \mathbf{I}_2$. The sensors were placed at locations $(50, 300), (200, 100), (400, 250),$ and, $(300, 400)$. All the coordinates are expressed in meters. The remaining parameters used in the simulation were the same as the

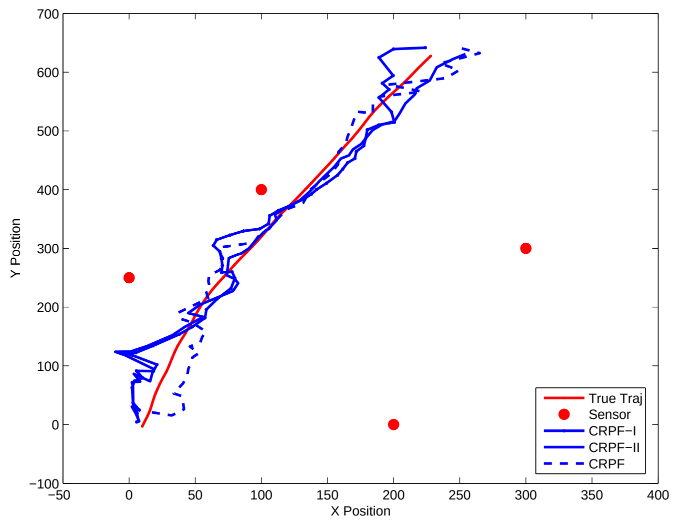


Figure 4.5: A run of the target trajectory for the various implementations of CRPF

ones used in experiment one (see Table 4.1). For comparison purposes we simulated, besides the CRPF algorithms, a SPF [30] that uses perfect knowledge of the statistics of the system (labeled SPR-True), and a SPF with incorrect statistics which assumes the state noise as $\mathbf{u}_t \sim \mathcal{N}(0, \tilde{\mathbf{C}}_u)$ with $\tilde{\mathbf{C}}_u = diag(0.1, 0.1)$ and the standard deviation of the measurement noise $\sigma_v = 3$ (labeled SPR-False). Figures 4.6 and 4.7 show the MSEs and percentage of missed trajectories obtained by the various implementations. It is clear that the performance of the three CRPFs is very similar and more robust than the SPF scheme when the assumed noise statistics deviate from the true values.

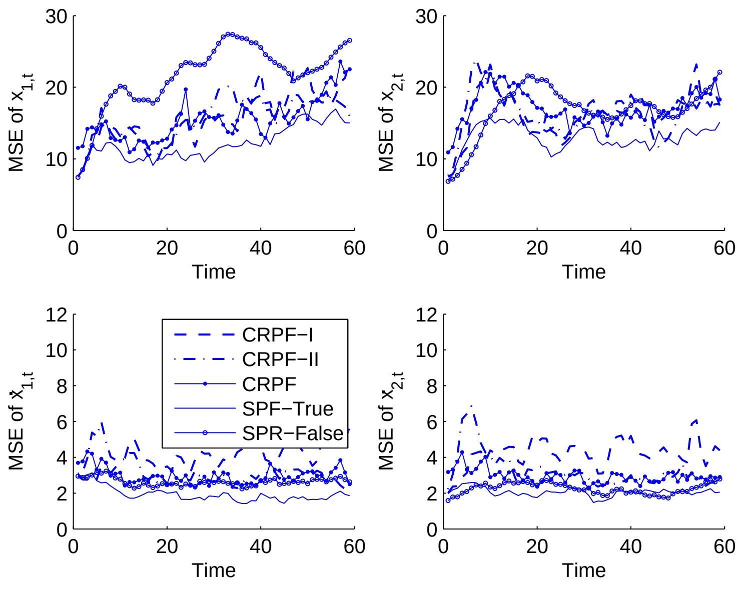


Figure 4.6: MSE of the target dynamics for various CRPF and SPF algorithms

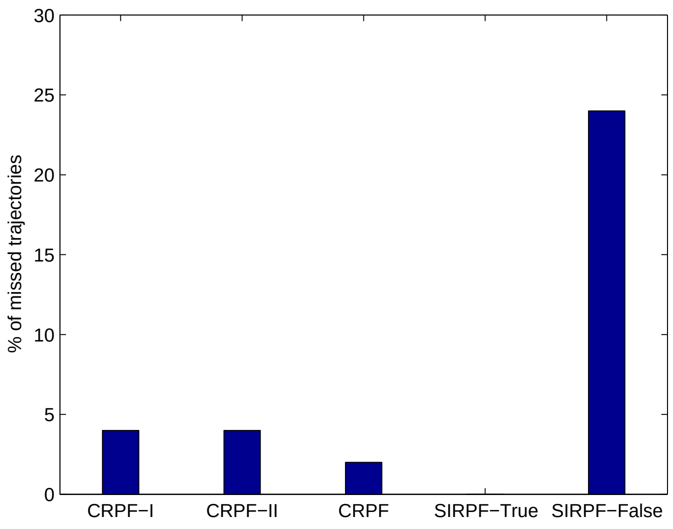


Figure 4.7: Percentage of missed tracks

4.5 Summary

Cost reference particle filtering (CRPF) methods address the problem of filtering for scenarios where the distribution of state and observation noises are unknown. We propose two alternative methods within the family of the original CRPF algorithms, with emphasis on building better proposal functions for a better state space exploration. The obtained algorithms show a good performance and a considerable reduction in computational complexity when compared to the original CRPF methods. We also compared the new algorithms with the standard particle filtering (SPF) approach that relies on the statistics of the system. The results show that the new algorithms are more robust than the SPF methods since they do not rely on any probabilistic assumptions.

Chapter 5

Sensor Self-Localization with Beacon Position Uncertainty

Algorithms for distributed sensor self-localization using beacon nodes are proposed. These beacon nodes broadcast some information which describes their positions. The sensor nodes with unknown location information utilize these descriptions along with the characteristics of received signals to obtain estimates of their positions. Sensors with resolved positions, in the successive stages of the algorithm also broadcast their location information to other sensors so that they can resolve their own positions. Conditional upon the availability of probabilistic distributions of noise processes, we propose iterative Bayesian and least squares methods in Section 5.3 and Monte Carlo sampling-based methods in Section 5.4. We also provide iterative hybrid Cramér-Rao bounds for distributed sensor self-localization in Section 5.5 and compare them with the proposed algorithms. We demonstrate the performance of the proposed algorithms through extensive computer simulations in Section 5.7.

5.1 Introduction and Motivation

In many sensor networking applications such as object tracking, sensors measure signals that are functions of the geometry between the object under surveillance and the sensor. Therefore to extract meaningful information of the object's dynamics, knowledge about the sensors' location is required. Incorporation of technologies like Global Positioning System (GPS) into these networks increases the cost and power requirements of the sensors. Alternatively, a periodic calibration of sensors' positions can be accomplished by establishing collaboration among the sensors. This procedure is known as self-localization.

In centralized sensor networks, the sensor measurements are routed to a central unit which performs the task of obtaining the sensors' locations. In [13, 38], the authors suggest a centralized framework and propose a maximum likelihood (ML) based solution for obtaining sensors' locations. Self-localization is also addressed using beacon nodes, also known as anchor nodes, leader nodes or access points [40,41]. In the remainder of this chapter we use the term beacon nodes when referring to sensors which have some initial information about their positions. In distributed sensor localization algorithms, beacons broadcast their location. Sensors estimate their distances from the beacons and using these distances, a set of equations or geometric constraints are formed. Triangulation, trilateration, least-squares or other optimization methods have been used in the literature to obtain the sensor positions [42], [43]. In these methods, the effects of uncertainty due to the measurement noise and beacon position are not considered. A quantitative comparison of some of these distributed techniques is provided in [44]. Static and mobile beacon-based self-localization procedures have also been explored in the literature (see, for example [41,45,46,47,48]). A more comprehensive survey of localization in sensor networks is provided in [49,50] and the references therein.

Here, we propose a distributed Bayesian framework for solving the sensor-localization problem. Bayesian techniques provide with a principled way of dealing with location uncertainty and multi-sensor fusion [51]. Some related work includes [52], [53], [54] and [55]. In [52,53], the authors propose an iterative multilateration technique for localizing nodes with unknown location. In our work, we consider similar distributive algorithms but we formulate the sensor location estimation from a probabilistic point of view and we include beacon position uncertainty. In [54], the sensor localization problem is addressed from a probabilistic perspective where sensors with unknown positions receive descriptions of beacon positions which is used by the sensors to update their positions. However, it is unclear how the authors compute and store these probability distributions. In [55], the authors present a probabilistic method termed as non-parametric belief propagation for self-localization of sensors. In this work, a sensor node computes a sample-based location description with each neighbor node, and obtains the joint marginal through resampling. In the sequel we provide a novel approach to sensor self-localization that uses a probabilistic formulation of the problem and takes into consideration computational and transmission issues of these probability distributions.

More specifically, we address the problem of distributed self-localization in sensor networks using iterative and Monte Carlo sampling-based methods. Some of the contributions are :

- Scalable distributed algorithms for sensor self-localization
- A statistical framework for incorporating beacon uncertainty
- A novel cost-based distributed algorithm when probability distributions of modeling noise processes are unknown
- Iterative hybrid Cramér-Rao Bounds (HCRBs) for sensor self-localization with beacon position uncertainty.

5.2 Sensor self-localization: Problem statement and notation

Consider the sensor network shown in Figure 5.1. The shaded nodes represent beacons, which broadcast their location details using probabilistic or spatial descriptions. The parameters of these descriptions are transmitted using known reference or pilot signals. We represent this prior information about their locations, $\boldsymbol{\ell}_b \in \mathbb{R}^2, b = 1, 2$ and 3 as $p(\boldsymbol{\ell}_b)$ and the signal received by a sensor as

$$y_{s,b} = f(\boldsymbol{\ell}_s, \boldsymbol{\ell}_b) + v_{s,b} \tag{5.1}$$

where $y_{s,b}$ is the received signal characteristic by sensor s from beacon b; $\boldsymbol{\ell}_s, \boldsymbol{\ell}_b \in \mathbb{R}^2$ are the positions of the nodes s and b in the two dimensional Cartesian coordinate system ($\boldsymbol{\ell}_s = [l_{s,x}, l_{s,y}]^\top$ and $\boldsymbol{\ell}_b = [l_{b,x}, l_{b,y}]^\top$); and $v_{s,b}$ is a Gaussian noise process with mean zero and variance $\sigma^2_{s,b}$. In our simulations we consider the following received

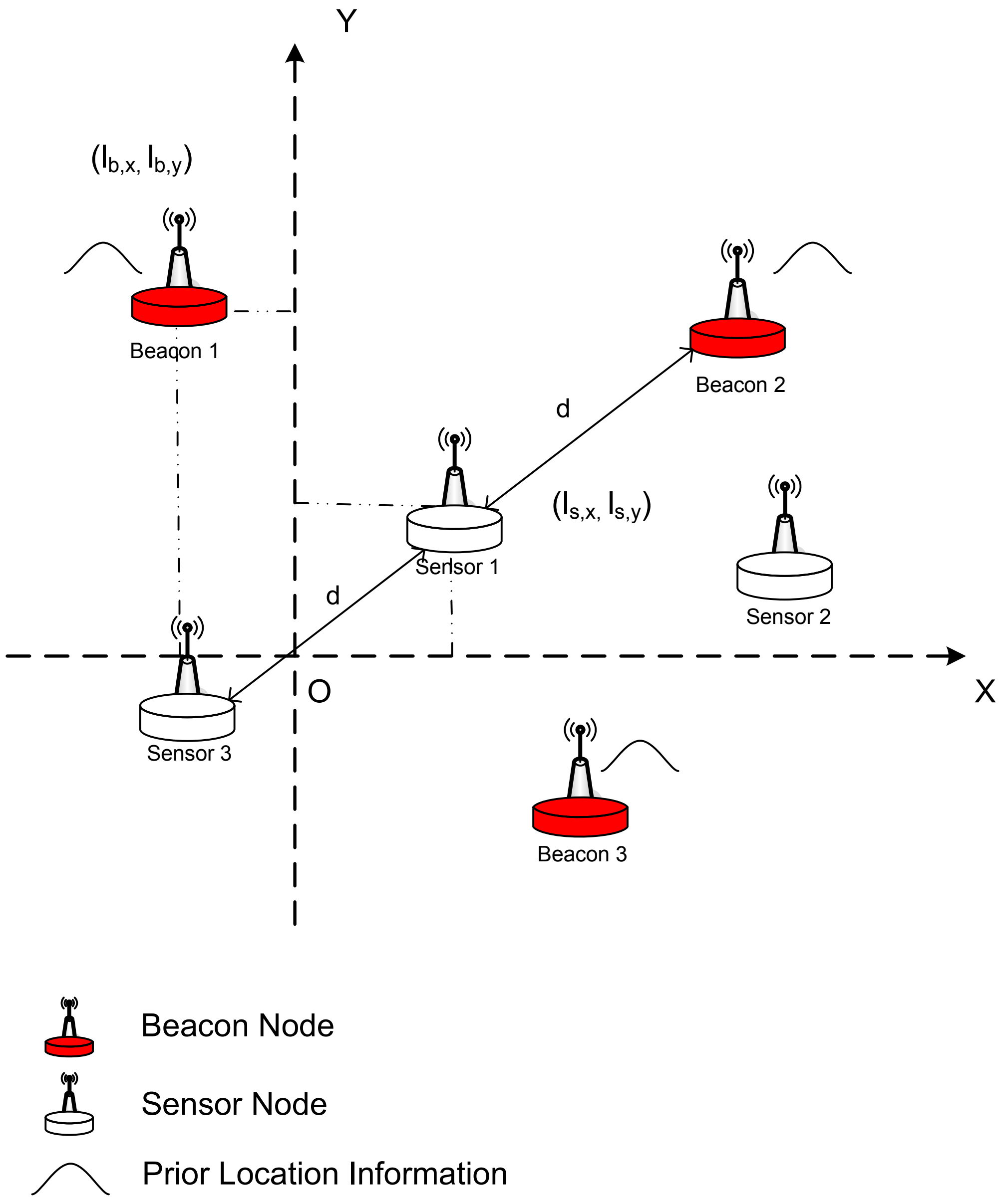


Figure 5.1: Sensor Network with beacon and sensor nodes

signal strength (RSS) model [38]

$$f(\boldsymbol{\ell}_s, \boldsymbol{\ell}_b) = \Psi_0 - 10\alpha \log_{10}\left(|\boldsymbol{\ell}_s - \boldsymbol{\ell}_b|\right) \tag{5.2}$$

where Ψ_0 is the power received at a known reference distance, α is the path-loss attenuation and $|\cdot|$ denotes norm of a vector.

We ask and answer the following question: when the distribution of the model noise processes are known, can we obtain the probability distribution of the sensor's location by combining the measurements and the beacon location descriptions. Furthermore, we also attempt to obtain descriptions of sensor locations when the probability distributions of the noise processes are unknown.

We explain our approach by using Figure 5.1. In each timeslot, the sensors and beacons broadcast their location descriptions which are utilized by the neighboring sensor nodes with unknown locations to obtain estimates of their locations. In the first timeslot, beacons 1, 2 and 3, shown in Figure 5.1, transmit their location descriptions. Sensor node 1, receives signals from all the beacons and combines the received prior descriptions and the measurements to obtain an estimate of its location. However, sensors 2 and 3 which are far away from beacons 1 and 2 respectively, do not receive any signals from these beacons and therefore cannot resolve the ambiguity in their positions. At the end of the first timeslot only the beacons 1, 2, 3 and sensor 1 have position information. Sensors which have knowledge about their position may broadcast their locations which may be used by other sensors to estimate their positions. Therefore in the next timeslot, sensors 2 and 3 utilize the beacon and sensor 1's locations to obtain estimates of their respective locations. The algorithm thus proceeds in successive timeslots with sensors estimating, broadcasting or updating their location estimates.

Clearly, the two important issues of the proposed distributed self-localization framework are (a) combination of measurements and beacon descriptions to obtain sensor estimates and (b) representation of estimates of the sensor location estimate for transmission. In the sequel, we propose methods which address these two problems.

5.3 Iterative sensor localization methods

In this section we propose two iterative methods for sensor localization which incorporate beacon location uncertainty and are based on linearization of the measurement function in (5.1). We consider a generic scenario where a sensor s receives signal measurements from K beacons as well as some information about their locations. We denote the set of measurements as $\{y_{s,1}, \cdots, y_{s,K}\}$ and describe the uncertainty in beacon location with standard multivariate Gaussian distributions. Stacking these set of measurements and writing in vector notation we have

$$\mathbf{y}_s = \mathbf{f}(\boldsymbol{\ell}) + \mathbf{v}_s \tag{5.3}$$

where $\mathbf{y}_s = [y_{s,1} \cdots y_{s,K}]^\top$, $\mathbf{f}(\boldsymbol{\ell}) = [f(\boldsymbol{\ell}_s, \boldsymbol{\ell}_1), \cdots, f(\boldsymbol{\ell}_s, \boldsymbol{\ell}_K)]^\top$, and $\mathbf{v} = [v_{s,1} \cdots v_{s,K}]^\top$ are all vectors of dimension $K \times 1$. Under assumptions of independent zero mean Gaussian, we denote the distribution of $\mathbf{v}_s$ as $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_v)$ where $\boldsymbol{\Sigma}_v = diag(\sigma^2_{v_{s,1}}, \cdots, \sigma^2_{v_{s,K}})$[1]. The prior location description of the beacon nodes, $p(\boldsymbol{\ell}_k) = \mathcal{N}(\hat{\boldsymbol{\ell}}_k, \boldsymbol{\Sigma}_{\boldsymbol{\ell}_k})$ $\forall k = 1 \cdots K$. Assuming independence among the beacon prior distributions, we have $p(\boldsymbol{\ell}_{1:K}) = \mathcal{N}(\hat{\boldsymbol{\mu}}_{\boldsymbol{\ell}}, \mathbf{P}_{\boldsymbol{\ell}_K})$, where $\hat{\boldsymbol{\mu}}_{\boldsymbol{\ell}} = \left[\hat{\boldsymbol{\ell}}_1^\top, \cdots, \hat{\boldsymbol{\ell}}_K^\top\right]^\top$, and $\mathbf{P}_{\boldsymbol{\ell}_K}$ is a block diagonal matrix, i.e., $\mathbf{P}_{\boldsymbol{\ell}} = diag(\boldsymbol{\Sigma}_{\boldsymbol{\ell}_1}, \cdots, \boldsymbol{\Sigma}_{\boldsymbol{\ell}_K})$[2].

[1]The symbol $diag(\boldsymbol{x})$ represents a diagonal matrix formed with the vector $\boldsymbol{x}$ as its diagonal.

[2]The symbol $diag(\mathbf{A}, \mathbf{B}) = \left(\begin{smallmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{smallmatrix}\right)$ where $\mathbf{A}, \mathbf{B}$ are diagonal matrices.

5.3.1 Bayesian method with linearization(BS method)

A direct approach for solving the sensors' positions is via the Maximum Likelihood framework. The optimization criterion can be written as

$$\underset{\boldsymbol{\ell}_s}{\operatorname{argmin}} \left[(\boldsymbol{y}_s - \mathbf{f}(\boldsymbol{\ell}))^\top \boldsymbol{\Sigma}_v^{-1} (\boldsymbol{y}_s - \mathbf{f}(\boldsymbol{\ell})) \right]. \tag{5.4}$$

The Levenberg-Marquardt method is one popular optimization method for finding solutions to problems like (5.4) [56]. However in its straightforward application, the prior information $p(\boldsymbol{\ell}_{1:K})$ of the beacons positions uncertainty is not incorporated. We change the procedure to incorporate this prior knowledge.

We denote the sensor position and the set of beacon locations as $\boldsymbol{\ell} = \left[\boldsymbol{\ell}_s^\top, \boldsymbol{\ell}_{1:K}^\top\right]^\top$, a $2(K+1) \times 1$ vector. Using a Taylor series expansion and neglecting higher order terms, we linearize (5.3), with respect to $\boldsymbol{\ell}$ around $\boldsymbol{\ell}^0$ as follows:

$$\boldsymbol{y}_s - f(\boldsymbol{\ell}^0) \approx \mathbf{H} \times (\boldsymbol{\ell} - \boldsymbol{\ell}^0) + \mathbf{v_s labeleq : lin_t rf} \tag{5.5}$$

$$\mathbf{H} = \begin{bmatrix} \frac{\partial f(\boldsymbol{\ell}_s,\boldsymbol{\ell}_1)}{\partial \boldsymbol{\ell}_s} & \frac{\partial f(\boldsymbol{\ell}_s,\boldsymbol{\ell}_1)}{\partial \boldsymbol{\ell}_1} & \cdots & \frac{\partial f(\boldsymbol{\ell}_s,\boldsymbol{\ell}_1)}{\partial \boldsymbol{\ell}_K} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f(\boldsymbol{\ell}_s,\boldsymbol{\ell}_K)}{\partial \boldsymbol{\ell}_s} & \frac{\partial f(\boldsymbol{\ell}_s,\boldsymbol{\ell}_K)}{\partial \boldsymbol{\ell}_1} & \cdots & \frac{\partial f(\boldsymbol{\ell}_s,\boldsymbol{\ell}_K)}{\partial \boldsymbol{\ell}_K} \end{bmatrix} = \begin{bmatrix} \frac{\partial f(\boldsymbol{\ell}_s,\boldsymbol{\ell}_1)}{\partial \boldsymbol{\ell}_s} & \frac{\partial f(\boldsymbol{\ell}_s,\boldsymbol{\ell}_1)}{\partial \boldsymbol{\ell}_1} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f(\boldsymbol{\ell}_s,\boldsymbol{\ell}_K)}{\partial \boldsymbol{\ell}_s} & \mathbf{0} & \cdots & \frac{\partial f(\boldsymbol{\ell}_s,\boldsymbol{\ell}_K)}{\partial \boldsymbol{\ell}_K} \end{bmatrix} \tag{5.6}$$

with $\frac{\partial f(\boldsymbol{\ell}_s,\boldsymbol{\ell}_i)}{\partial \boldsymbol{\ell}_j} = 0$, $\forall i \neq j$, $i, j \in 1 \cdots K$. Here $\mathbf{H}$ is a $2K \times 2(K+1)$ matrix and $\boldsymbol{\ell}$ and $\boldsymbol{\ell}^0$ are vectors of dimension $2(K+1) \times 1$. A Bayesian solution to this linear problem which incorporates the prior information is as follows [57]:

$$\hat{\boldsymbol{\ell}} - \boldsymbol{\ell}^0 = (\boldsymbol{\Sigma}_l^{-1} + \mathbf{H}^\top \boldsymbol{\Sigma}_v^{-1} \mathbf{H})^{-1} \mathbf{H}^\top \boldsymbol{\Sigma}_v^{-1} (\boldsymbol{y}_s - \mathbf{f}(\boldsymbol{\ell}^0)) \tag{5.7}$$

where $\mathbf{\Sigma}_l^{-1} = diag(\mathbf{0}_2,\ \mathbf{P}_{\boldsymbol{\ell}_K}^{-1})$. For the sensor localization problem, we solve for $\boldsymbol{\ell}$ by iteratively computing (5.7) with a suitable stopping criterion. Therefore, at iteration k we have

$$\hat{\boldsymbol{\ell}}^k = \hat{\boldsymbol{\ell}}^{(k-1)} + (\mathbf{\Sigma}_l^{-1} + \mathbf{H}^{(k)\top}\mathbf{\Sigma}_v^{-1}\mathbf{H}^{(k)})^{-1}\mathbf{H}^{(k)\top}\mathbf{\Sigma}_v^{-1}(\boldsymbol{y}_s - \mathbf{f}(\hat{\boldsymbol{\ell}}^{(k-1)})) \tag{5.8}$$

where $\mathbf{H}^{(k})$ is defined in (5.6) and computed at $\hat{\boldsymbol{\ell}}^{(k-1)}$. The initialization of the beacon parameters by $\hat{\boldsymbol{\ell}}^{(0)}$ is obtained using $\hat{\boldsymbol{\mu}}_{\boldsymbol{\ell}}$ and the sensor location is initialized using the mean of the beacon positions. The covariance matrix of the estimate of $\boldsymbol{\ell}$ is $\left(\mathbf{\Sigma}_l^{-1} + \mathbf{H}^{(k)\top}\mathbf{\Sigma}_v^{-1}\mathbf{H}^{(k)}\right)$.

5.3.2 Least squares method with linearization (LS method)

A least squares criterion that incorporates the prior knowledge of the beacons locations can be written as [58]

$$\underset{\boldsymbol{\ell}}{\text{argmin}} \left[(\boldsymbol{\ell} - \boldsymbol{\ell}^0)^\top \mathbf{\Sigma}_l^{-1}(\boldsymbol{\ell} - \boldsymbol{\ell}^0) + |\boldsymbol{y}_s - \mathbf{H} \times (\boldsymbol{\ell} - \boldsymbol{\ell}^0)|^2\right]$$

and its solution is

$$\hat{\boldsymbol{\ell}} = \boldsymbol{\ell}^0 + \left(\mathbf{\Sigma}_l^{-1} + \mathbf{H}^\top\mathbf{H}\right)^{-1}\mathbf{H}^\top\left(\boldsymbol{y}_s - f(\boldsymbol{\ell}^0)\right).$$

Similarly we can iteratively solve for $\boldsymbol{\ell}$. Thus, at iteration k, we have

$$\hat{\boldsymbol{\ell}}^{(k)} = \hat{\boldsymbol{\ell}}^{(k-1)} + (\mathbf{\Sigma}_l^{-1} + \mathbf{H}^{(k)\top}\mathbf{H}^{(k)})^{-1}\mathbf{H}^{(k)\top}(\boldsymbol{y}_s - \mathbf{f}(\hat{\boldsymbol{\ell}}^{(k-1)})) \tag{5.9}$$

and the corresponding covariance matrix of $\hat{\boldsymbol{\ell}}^{(k)}$ is $\left(\boldsymbol{\Sigma}_l^{-1} + \mathbf{H}^{(k)\top}\mathbf{H}^{(k)}\right)$.

Clearly, we can see that (5.8) differs from (5.9) in that the least squares solution does not take into account the distributions of the noise process $\mathbf{v}$. In the above proposed iterative methods, the iterations are performed in one timeslot and once a sensor resolves its position, then in the consequent timeslots it, too, broadcasts its position description. Using the final estimates and the covariance matrix, the sensor position can be modeled as a Gaussian distribution and these parameters are transmitted in the subsequent timeslots for both the BS and LS methods.

5.4 Monte Carlo-based methods for Sensor localization

In this section we propose Monte Carlo-based methods which approximate the posterior distribution of the sensor location by a set of samples. In these methods the measurement functions are not linearized but the marginal distributions are parameterized and approximated. In this section, too, we consider the earlier scenario where each sensor receives measurements from K beacons and their location descriptions.

5.4.1 Importance sampling-based method (IS method)

In a Bayesian framework, all knowledge about the sensors' and beacons' positions is contained in the posterior distribution $p(\boldsymbol{\ell} \,|\, \boldsymbol{y}_s)$. We therefore have

$$p(\boldsymbol{\ell} \,|\, \boldsymbol{y}_s) = p(\boldsymbol{\ell}_s, \boldsymbol{\ell}_1 \cdots, \boldsymbol{\ell}_K \,|\, \boldsymbol{y}_s) \propto p(\boldsymbol{\ell}_s) \prod_{i=1}^{K} p(y_{s,i} \,|\, \boldsymbol{\ell}_s, \boldsymbol{\ell}_i) p(\boldsymbol{\ell}_i) \tag{5.10}$$

which is obtained using Bayes' theorem and assuming independence among the prior distributions. Closed form analytical solutions to this posterior density cannot be obtained when the function $f(\cdot)$ in (5.1) is nonlinear in the state parameters and the noise process $v_{s,i}$ is non-Gaussian. Therefore we use Monte Carlo (MC) methods for capturing the posterior distribution or some of its statistics. In most MC methods these posterior distributions are represented by sample-based discrete random measures. Importance Sampling (IS) is one such method where one can obtain a discrete representation of the posterior distribution [59].

Very briefly, with IS, the density $p(\boldsymbol{\ell})$ is approximated by a large weighted set of samples $\Xi_l \equiv \{\boldsymbol{\ell}^{(m)}, w^{(m)}\}_{m=1}^{M}$ with M being the total number of drawn samples and m the sample index. This approach is particularly useful when it is infeasible to draw samples directly from the density $p(\boldsymbol{\ell})$ but ti can be evaluated up to a constant. The samples are obtained from another function known as importance function or proposal density, $\pi(\boldsymbol{\ell})$. The importance weights are proportional to $\frac{p(\boldsymbol{\ell}^{(m)})}{\pi(\boldsymbol{\ell}^{(m)})}$ and measure the quality of the generated particles. A more rigorous treatment of the subject can be found in [59].

The posterior density (5.10) is approximated as a weighted set of particles

$\{w^{(m)}, \boldsymbol{\ell}_s^{(m)}, \boldsymbol{\ell}_1^{(m)}, \cdots, \boldsymbol{\ell}_K^{(m)}\}$,

$$p(\boldsymbol{\ell}_s, \boldsymbol{\ell}_1, \cdots, \boldsymbol{\ell}_K \,|\, y_{s,1}, \cdots, y_{s,K}) = \sum_{m=1}^{M} w^{(m)} \delta\left(\boldsymbol{\ell}_s - \boldsymbol{\ell}_s^{(m)}\right) \times \prod_{b=1}^{K} \delta\left(\boldsymbol{\ell}_b - \boldsymbol{\ell}_b^{(m)}\right) \quad (5.11)$$

where weights, $w^{(m)}$, are obtained according to

$$w^{(m)} \propto \frac{p(\boldsymbol{\ell}_s^{(m)})}{\pi(\boldsymbol{\ell}_s^{(m)})} \prod_{b=1}^{K} \frac{p(y_{s,b} \,|\, \boldsymbol{\ell}_s^{(m)}, \boldsymbol{\ell}_b^{(m)}) p(\boldsymbol{\ell}_b^{(m)})}{\pi(\boldsymbol{\ell}_b^{(m)})} \quad (5.12)$$

with samples of the beacon location drawn from their transmitted prior location distributions, i.e., $\boldsymbol{\ell}_b^{(m)} \sim p(\boldsymbol{\ell}_b)$, and samples representing the sensor location drawn from an importance distribution function $\pi(\boldsymbol{\ell}_s)$. In Section 5.4.3, we propose methods for constructing these importance distributions.

The posterior density is then easily marginalized, i.e.,

$$p(\boldsymbol{\ell}_s \,|\, y_{s,1}, \cdots, y_{s,K}) \approx \sum_{m=1}^{M} w^{(m)} \delta\left(\boldsymbol{\ell}_s - \boldsymbol{\ell}_s^{(m)}\right). \quad (5.13)$$

If the sensor obtains information about its location, it is then described by a probability distribution which is broadcast in the next timeslot to other sensors so that they estimate their positions. Transmission of these sample-based distributions would require large amounts of communication resources, and therefore we approximate them with standard probability distributions and broadcast their parameters only. As an example, we approximate the sample-based distribution by a Gaussian $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\ell}_s}, \boldsymbol{\Sigma}_{\boldsymbol{\ell}_s})$ where the mean and covariance matrices $\boldsymbol{\mu}_{\boldsymbol{\ell}_s}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\ell}_s}$ are obtained as

$$\begin{aligned}
\boldsymbol{\mu}_{\boldsymbol{\ell}_s} &= \sum_{m=1}^{M} w^{(m)} \boldsymbol{\ell}_s^{(m)} \\
\boldsymbol{\Sigma}_{\boldsymbol{\ell}_s} &= \sum_{m=1}^{M} w^{(m)} \left({\boldsymbol{\ell}_s}^{(m)} - \boldsymbol{\mu}_{\boldsymbol{\ell}_s} \right) \left({\boldsymbol{\ell}_s}^{(m)} - \boldsymbol{\mu}_{\boldsymbol{\ell}_s} \right)^{\top} .
\end{aligned} \tag{5.14}$$

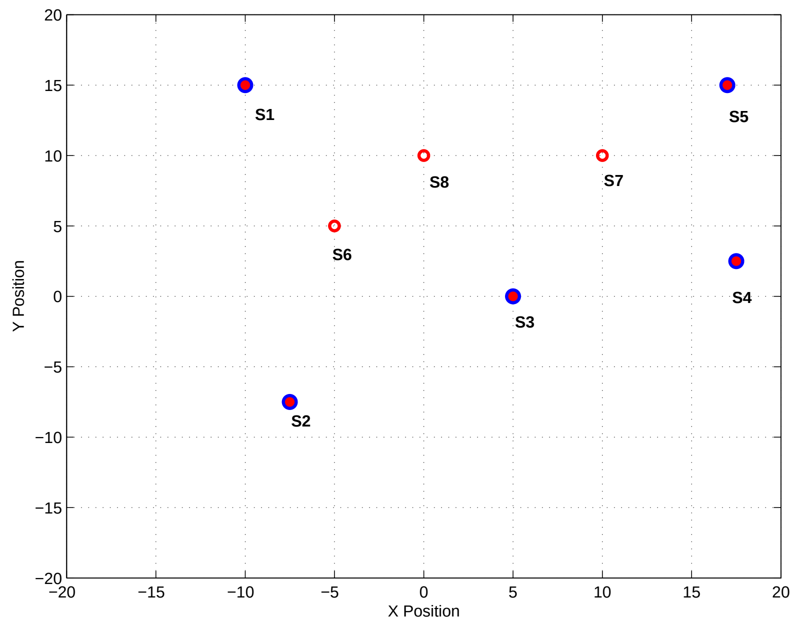


Figure 5.2: Sensor Network.

As an example, consider Figure 5.2, where beacons $S1, S2, S3, S4, S5$ are represented by Gaussian distributions. In the 1st timeslot, sensors $S6, S7$ estimate their positions. These estimates are approximated using Gaussian distributions as shown in Figure 5.3. Similarly in the second timeslot, $S8$ estimates its position using the estimates of $S6, S7$ and approximates its distribution with a Gaussian as shown in Figure 5.4.

In summary, the main steps of this importance sampling-based procedure are:

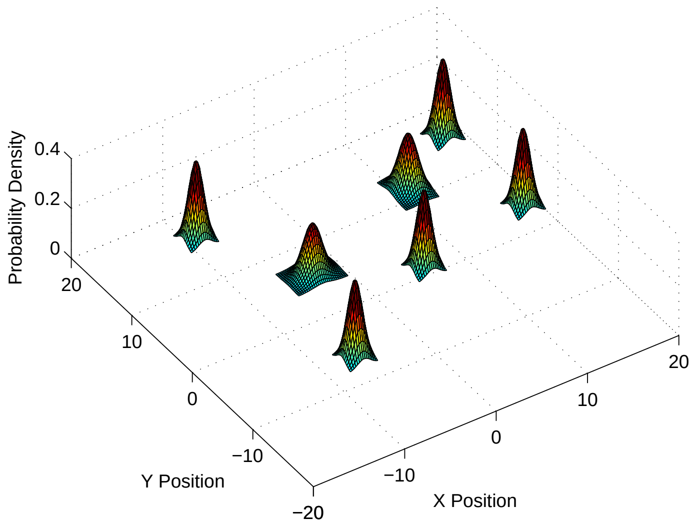


Figure 5.3: Iterative Stages:Timeslot 1

- Generation of beacon and sensor samples $\{\boldsymbol{\ell}_s^{(m)}, \boldsymbol{\ell}_1^{(m)}, \cdots, \boldsymbol{\ell}_K^{(m)}\}$ using the beacon prior location descriptions and the sensor location importance function.
- Computation of weights using (5.12).
- Estimation of the distribution of the sensor locations through parametric distributions.

Note that these fusion algorithms for sensor localization are suboptimal. Due to lack of space, we do not present the optimal fusion algorithm which has significant communication, topological, and intelligent processing requirements. These requirements arise because the marginal posterior distributions of the sensors are no longer independent but entwined with one another.

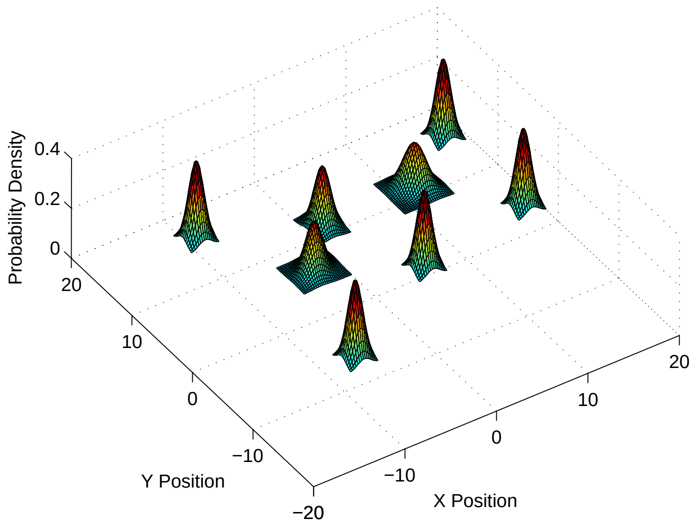


Figure 5.4: Iterative Stages:Timeslot 2

5.4.2 Cost based Monte Carlo-based sampling method (CS)

In the above proposed IS method, the computation of the weights in (5.12) requires the evaluation of the likelihood term for which the probability distribution of the measurement noise should be completely known. In many scenarios, such probabilistic information is not available. We propose an alternative sampling method which does not make probabilistic assumptions of the model noise. In the following methods we only require that the noise process is zero mean.

As earlier, each beacon represents its location uncertainty using spatial distributions. Under assumption of zero mean noise, a least squares criterion for obtaining the sensor location given the measurements $y_{s,1} \cdots y_{s,K}$ can be formulated as

$$\hat{\boldsymbol{\ell}}_s = \underset{\boldsymbol{\ell}_s}{\operatorname{argmin}} \left\{ \mathcal{C}(\boldsymbol{\ell}_s) = \sum_{b=1}^{K} |y_{s,b} - f(\boldsymbol{\ell}_s, \boldsymbol{\ell}_b)|^2 \right\}. \tag{5.15}$$

Note that in this formulation we have not explicitly included the beacon location information. However, the optimization is carried out over regions given by the beacon location descriptions. Using this formulation as a starting point, we propose a procedure where we draw samples representing the beacon and sensor locations and associate costs with these samples using the measurements and beacon location descriptions. The main steps of the algorithm are as follows:

- *Generation of beacon and sensor samples*: The beacon location samples $\{\boldsymbol{\ell}_1^{(m)}, \cdots, \boldsymbol{\ell}_K^{(m)}\}$ are drawn using the beacon location descriptions. The sensor samples $\boldsymbol{\ell}_s^{(m)}$ are obtained using an importance function as discussed in Section 5.4.3. These beacon and sensor samples can be considered as representative solutions of the beacon and sensor locations.

- *Obtaining costs*: For each of the samples generated in the previous step we associate a cost that defines their quality under sensor measurements. In general the cost function depends on the observations and the sampled locations. We assign the costs according to

$$\mathcal{C}^{(m)} = \sum_{j=1}^{K} \varrho(y_{s,j}, \boldsymbol{\ell}_s^{(m)}, \boldsymbol{\ell}_j^{(m)}) = \sum_{j=1}^{K} \varrho(\epsilon_{s,j}) \tag{5.16}$$

  where $\epsilon_{s,j} = y_{s,j} - f(\boldsymbol{\ell}_s^{(m)}, \boldsymbol{\ell}_j^{(m)})$ and $\varrho(\cdot,\cdot,\cdot)$, a nonnegative function of its assignment. The cost function is chosen such that samples which are more representative of the locations have smaller costs and samples which are far away from the true locations have larger costs. We use the following cost functions in our simulations:

1. L2 cost function: $\varrho(\epsilon) = |\,\epsilon\,|^2$
2. L1 cost function: $\varrho(\epsilon) = |\,\epsilon\,|$
3. Fair function: $\varrho(\epsilon) = 2k^2 \left[\frac{|\epsilon|}{k} log\left(1 + \frac{|\epsilon|}{k}\right)\right]$ with $k = 1.3998$ [36].

Using these costs we form a pseudo probability measure such that,

$$\begin{aligned} \tilde{\pi}(\boldsymbol{\ell}^{(m)}) &\propto \frac{1}{\mathcal{C}^{(m)}} \\ \text{such that } \sum_m \tilde{\pi}(\boldsymbol{\ell}^{(m)}) &= 1. \end{aligned} \tag{5.17}$$

- *Estimating sensor location regions*: We can approximate the sensor location distribution by using standard probability distributions such as the Gaussian distribution (as in the previous subsection). These distributions can be obtained using the sample mean sensor location and the covariance matrix which are calculated using the pseudo probability measure (5.17). These statistics are obtained as earlier with

$$\begin{aligned} \tilde{\boldsymbol{\mu}}_{\boldsymbol{\ell}_s} &\approx \sum_{m=1}^{M} \tilde{\pi}\ (\boldsymbol{\ell}^{(m)})\boldsymbol{\ell}_s^{(m)} \\ \tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\ell}_s} &\approx \sum_{m=1}^{M} \tilde{\pi}(\boldsymbol{\ell}^{(m)}) \left(\boldsymbol{\ell}_s^{(m)} - \tilde{\boldsymbol{\mu}}_{\boldsymbol{\ell}_s}\right) \left(\boldsymbol{\ell}_s^{(m)} - \tilde{\boldsymbol{\mu}}_{\boldsymbol{\ell}_s}\right)^\top \\ &= \begin{pmatrix} \tilde{\sigma}_{\boldsymbol{\ell}_{s,xx}} & \tilde{\sigma}_{\boldsymbol{\ell}_{s,xy}} \\ \tilde{\sigma}_{\boldsymbol{\ell}_{s,yx}} & \tilde{\sigma}_{\boldsymbol{\ell}_{s,yy}} \end{pmatrix}. \end{aligned} \tag{5.18}$$

Another possibility is to use spatial regions such as circles, squares or elliptical regions.

1. Square region: A square is completely characterized by its center $\boldsymbol{\mu}_c$ and

the length of its side ρ_c. We represent this region as $\mathcal{S}q(\boldsymbol{\mu}_c, \rho_c)$ with center $\boldsymbol{\mu}_c = \tilde{\boldsymbol{\mu}}_{\boldsymbol{\ell}_s}$ and the length of the side $\rho_c = \kappa \max(\sqrt{2\tilde{\sigma}_{\boldsymbol{\ell}_s,xx}}, \sqrt{2\tilde{\sigma}_{\boldsymbol{\ell}_s,yy}})$ where κ is a scaling factor. We chose $\kappa = 2$ in our simulations.

2. Circular region: This region is completely specified by its center $\boldsymbol{\mu}_c$ and radius ρ_s. We represent the region as $\mathcal{C}i(\boldsymbol{\mu}_c, \rho_c)$ with $\boldsymbol{\mu}_c = \tilde{\boldsymbol{\mu}}_{\boldsymbol{\ell}_s}$ and $\rho_c = \kappa \max(\sqrt{2\tilde{\sigma}_{\boldsymbol{\ell}_s,xx}}, \sqrt{2\tilde{\sigma}_{\boldsymbol{\ell}_s,yy}})$.

3. Elliptical region: This region is completely specified by its center $\boldsymbol{\mu}_c$, the lengths of its major ρ_a and minor axis ρ_b and the angle of inclination ϕ of the major axis with respect to the horizontal direction. Thus, this region can be represented as $\mathcal{E}l(\boldsymbol{\mu}_c, \rho_a, \rho_b, \theta)$. These parameters can be obtained from the covariance matrix by

$$\begin{aligned} \rho_a, \rho_b &= \kappa \left(\frac{\tilde{\sigma}_{\boldsymbol{\ell}_s,xx} + \tilde{\sigma}_{\boldsymbol{\ell}_s,yy} \pm \sqrt{(\tilde{\sigma}_{\boldsymbol{\ell}_s,xx} + \tilde{\sigma}_{\boldsymbol{\ell}_s,yy})^2 + 4\tilde{\sigma}^2_{\boldsymbol{\ell}_s,xy}}}{2} \right)^{1/2} \\ \theta &= \frac{1}{2} \arctan \left(\frac{2\tilde{\sigma}_{\boldsymbol{\ell}_s,xy}}{\tilde{\sigma}_{\boldsymbol{\ell}_s,yy} - \tilde{\sigma}_{\boldsymbol{\ell}_s,xx}} \right). \end{aligned} \tag{5.19}$$

Here we have considered an inclined ellipse, whose angle of inclination is obtained using (5.19). Similarly, we can construct regions for inclined spatial regions.

5.4.3 Construction of the Importance Function for Sensor Location

Samples representing the beacons' positions are drawn from their prior or marginal posterior distributions of their locations. However, in absence of any prior of the sensor location we construct importance functions for the sensor location. Intuitively, we would like to draw samples from regions around the sensors true location density. Some approaches for constructing this density are as follows:

1. Uniform sampling: A naive approach is to draw samples from the entire sensing field by assuming a uniform importance function with the parameters specified by the boundaries of the sensor field. This approach is simple but it requires a large number of samples so that the regions of interest, around the true sensor location are adequately represented. Clearly with this scheme, there is a huge waste of samples and computational resources.

2. Beacon-assisted sampling: A sensor receiving signals from beacons, indicates that the sensor is in the region occupied by the beacon. We can exploit this aspect and draw samples from the regions occupied by the beacons. Denoting the region of transmission of beacon b as A_b, samples $\boldsymbol{\ell}_s^{(m)}$ are drawn from the region $A = \bigcup_b A_b$. A simple way to achieve this is by assuming that the regions A_b are rectangular or circular.

3. Trilateration-assisted sampling Scheme-I: More efficient sampling densities are those from which we can draw particles belonging to the region $A = \bigcap_b A_b$. This is closely related to solving the sensor localization problem itself. For RSS-based measurements we obtain initial estimates of the sensor location using fast

trilateration methods for constructing the prior density. We first convert the RSS measurements to distance measurements. Consider the measurement $y_{s,b}$ received by sensor s from beacon b and define

$$\begin{aligned} z_{s,b} &= \frac{\Psi_0 - y_{s,b}}{10\alpha} \qquad b = 1, 2, \cdots K \\ d_{s,b} = 10^{z_{s,b}} &\approx \sqrt{(l_{s,x} - \hat{l}_{b,x})^2 + (l_{s,y} - \hat{l}_{b,y})^2} \end{aligned} \tag{5.20}$$

where $d_{s,b}$ represents an initial estimate of the Euclidean distance between the sensor and the beacon, $z_{s,b}$ is an intermediate variable and $[\hat{l}_{b,y}, \hat{l}_{b,y}]$ are estimates of the beacon position. When beacons approximate their location information using Gaussian distributions, this estimate is readily available through the mean of the distribution. Using (5.20) and a fast trilateration scheme we obtain the initial estimates $\tilde{\boldsymbol{\mu}}_s = [\tilde{l}_{s,x}, \tilde{l}_{s,y}]^\top$ of the sensor position. Thus, we can construct a Gaussian importance function, $\pi(\boldsymbol{\ell}_s) = \mathcal{N}(\tilde{\boldsymbol{\mu}}_s, \tilde{\boldsymbol{\Sigma}}_s)$ for drawing samples to compute the weights in (5.12). The choice of the covariance matrix $\tilde{\boldsymbol{\Sigma}}_s$ is arbitrary. In our simulations we have chosen $\tilde{\boldsymbol{\Sigma}}_s = \tilde{\sigma}_s^2 I_2$ with a judicious choice of $\tilde{\sigma}_s$.

4. Trilateration-assisted sampling Scheme-II: Another approach is to run the iterative algorithms discussed in Section 5.3 for smaller number of iterations and use these rough estimates to construct Gaussian distributions as discussed above.

5.5 Hybrid Cramér-Rao bounds for sensor self-localization

We now derive Cramér-Rao Bounds (CRBs) for sensor self-localization for the two scenarios (a) a single sensor which receives measurements and location information from multiple beacons and (b) multiple sensors with multiple beacons which localize themselves using the proposed distributive framework.

The CRB is the bound on the variance of unbiased estimators for non-random parameters and the Bayesian or the Van-Trees version of the CRB is the bound of mean square error estimators of random parameters [57, 60]. We consider here a hybrid Cramér-Rao bound (HCRB) which is a bound on the mean square estimation error for random and non-random parameters [61, 62].

5.5.1 Single sensor, multiple beacons

Recall that $\boldsymbol{\ell} = [\boldsymbol{\ell}_s, \boldsymbol{\ell}_1, \cdots \boldsymbol{\ell}_K] = \left[\boldsymbol{\ell}_s, \bar{\boldsymbol{\ell}}\right]$, is the unknown set of parameters to be estimated, i.e., the vector containing locations of the sensor $\boldsymbol{\ell}_s$ and the beacons $\bar{\boldsymbol{\ell}}$ which are modeled as non-random and random parameters, respectively, and $\mathbf{y}_s = [y_{s,1} \cdots y_{s,K}]$ is the set of beacon signal measurements made by sensor s. The HCRB has the following form:

$$\boldsymbol{\Sigma}_{\boldsymbol{\ell}} = \mathbb{E}\left\{[\hat{\boldsymbol{\ell}} - \boldsymbol{\ell}][\hat{\boldsymbol{\ell}} - \boldsymbol{\ell}]^\top\right\} \geq \mathbf{J}^{-1} \tag{5.21}$$

where $\boldsymbol{\Sigma}_{\boldsymbol{\ell}}$ is the estimation error covariance matrix and $\mathbf{J}$ is the hybrid Fisher information matrix (HFIM) defined as

$$\begin{aligned}\mathbf{J} &= \mathbb{E}_{\boldsymbol{\ell},\mathbf{y}_s}\left[\left\{\nabla_{\boldsymbol{\ell}} \log p(\mathbf{y}_s,\boldsymbol{\ell}) \ \nabla_{\boldsymbol{\ell}}^{\top} \log p(\mathbf{y}_s,\boldsymbol{\ell})\right\}\right] \\ \mathbf{J} &= \mathbb{E}_{\boldsymbol{\ell}}\left[\mathbb{E}_{\mathbf{y}_s\,|\,\boldsymbol{\ell}}\left[\left\{\nabla_{\boldsymbol{\ell}} \log p(\mathbf{y}_s\,|\,\boldsymbol{\ell}) \ \nabla_{\boldsymbol{\ell}}^{\top} \log p(\mathbf{y}_s\,|\,\boldsymbol{\ell})\right\}\right]\right] + \mathbf{J}_b \\ \mathbf{J} &= \mathbf{J}_s + \mathbf{J}_b. \end{aligned} \tag{5.22}$$

with

$$\mathbf{J}_b = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbb{E}_{\bar{\boldsymbol{\ell}}}\left[\left\{\nabla_{\bar{\boldsymbol{\ell}}} \log p(\bar{\boldsymbol{\ell}}) \ \nabla_{\bar{\boldsymbol{\ell}}}^{\top} \log p(\bar{\boldsymbol{\ell}})\right\}\right] \end{pmatrix} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{bb} \end{pmatrix}.$$

and $\nabla_{\boldsymbol{\ell}} = [\frac{\partial}{\partial \boldsymbol{\ell}_s}, \frac{\partial}{\partial \boldsymbol{\ell}_1} \cdots \frac{\partial}{\partial \boldsymbol{\ell}_K}]^{\top}$. We model the beacon prior location distribution as a Gaussian, therefore we have the matrix $\mathbf{J}_{bb}$ equal to the block diagonal matrix of inverses of all the covariance matrices of the beacons' location distributions. Details of the computation of these CRBs are further elaborated in Appendix **??**.

5.5.2 Multiple sensors, multiple beacons

We calculate bounds for the proposed distributed framework where at every timeslot sensors with known position information broadcast their location information and sensors with unknown information attempt to localize. Also sensors with known information utilize measurements from new beacons to update their information. These aspects are considered in the calculation of the HCRB for the proposed distributive framework.

With a slight change in the notation, we write the received signal at sensor s as

$$\mathbf{y}_s = \mathbf{f}_s\left(\boldsymbol{\ell}\right) + \mathbf{v}_s \tag{5.23}$$

with $\mathbf{f}_s\left(\boldsymbol{\ell}\right) = [f_{s,1}(\cdot)\cdots f_{s,N}(\cdot)]^\top$ and $\mathbf{v}_s = [v_{s,1}\cdots v_{s,N}]$. We form a vector $\hat{\mathbf{y}}$ by stacking all the sensor measurements $\hat{\mathbf{y}} = [\boldsymbol{y}_1{}^\top, \cdots, \boldsymbol{y}_N{}^\top]^\top$. However only sensors with known location information broadcast their position information. Therefore the entries in this vector corresponding to sensors which do not broadcast is zero. We obtain the expression

$$\hat{\mathbf{y}} = \hat{\mathbf{f}}\left(\boldsymbol{\ell}\right) + \hat{\mathbf{v}}. \tag{5.24}$$

Considering $\boldsymbol{\ell}$ as the set of sensor locations with prior information as random parameters and the set of sensor locations with no prior information as non-random parameters, the HFIM at timeslot n can be obtained as in (5.22)

$$\mathbf{J}^{(n)} = \mathbb{E}_{\boldsymbol{\ell},\,\hat{\mathbf{y}}^{1:n}}\left[-\left\{\Delta_{\boldsymbol{\ell}}^{\boldsymbol{\ell}}\log p(\hat{\mathbf{y}}^{1:n},\boldsymbol{\ell})\right\}\right] \tag{5.25}$$

with expectation performed over sensors with resolved positions and $\hat{\mathbf{y}}^{1:n}$, the set of all measurements from timeslot 1 to n. We have that $p(\hat{\mathbf{y}}^{1:n},\boldsymbol{\ell}) = p(\hat{\mathbf{y}}^{n}\,|\,\boldsymbol{\ell})p(\hat{\mathbf{y}}^{1:n-1},\boldsymbol{\ell})$ and therefore we can write (5.25)

$$\begin{aligned}
\mathbf{J}^{(n)} &= \mathbb{E}_{\boldsymbol{\ell},\,\hat{\mathbf{y}}^{n}}\left[-\left\{\Delta_{\boldsymbol{\ell}}^{\boldsymbol{\ell}}\log p(\hat{\mathbf{y}}^{n}\,|\,\boldsymbol{\ell})\right\}\right] + \mathbb{E}_{\boldsymbol{\ell}\,\hat{\mathbf{y}}^{1:n}}\left[-\left\{\Delta_{\boldsymbol{\ell}}^{\boldsymbol{\ell}}\log p(\hat{\mathbf{y}}^{1:n-1},\boldsymbol{\ell})\right\}\right] \\
&= \mathbb{E}_{\boldsymbol{\ell},\,\hat{\mathbf{y}}^{n}}\left[-\left\{\Delta_{\boldsymbol{\ell}}^{\boldsymbol{\ell}}\log p(\hat{\mathbf{y}}^{n}\,|\,\boldsymbol{\ell})\right\}\right] + \mathbf{J}^{(n-1)} \\
&= \mathbb{E}_{\boldsymbol{\ell}}\left[\mathbb{E}_{\hat{\mathbf{y}}^{n}\,|\,\boldsymbol{\ell}}\left[\left\{\nabla_{\boldsymbol{\ell}}\log p(\hat{\mathbf{y}}^{n}\,|\,\boldsymbol{\ell})\ \ \nabla_{\boldsymbol{\ell}}^{\top}\log p(\hat{\mathbf{y}}^{n}\,|\,\boldsymbol{\ell})\right\}\right]\right] + \mathbf{J}^{(n-1)} \\
&= \mathbb{E}_{\boldsymbol{\ell}}\left[\left\{\nabla_{\boldsymbol{\ell}}\hat{\mathbf{f}}^{\top}(\boldsymbol{\ell})\ \boldsymbol{\Sigma}_v^{-1}\ \nabla_{\boldsymbol{\ell}}^{\top}\hat{\mathbf{f}}^{\top}(\boldsymbol{\ell})\right\}\right] + \mathbf{J}^{(n-1)}
\end{aligned} \tag{5.26}$$

with $\mathbf{J}^{(0)} = \mathbb{E}_{\boldsymbol{\ell}}\left[\left\{\nabla_{\boldsymbol{\ell}} \log p(\boldsymbol{\ell}) \;\; \nabla_{\boldsymbol{\ell}}^{\top} \log p(\boldsymbol{\ell})\right\}\right]$. If the sensor in a particular timeslot is able to resolve its position then its HCRB is smaller than when it is not able to resolve its position. Using this criterion we identify sensors which can become beacons in the next timeslot. The information obtained in timeslot $n-1$ is treated as the prior information in timeslot n. We rearrange the matrix $\mathbf{J}^{(n-1)}$ such that the elements corresponding to sensors with unknown locations are all 0's and term this new prior matrix as $\tilde{\mathbf{J}}_b^{(n-1)}$. We write (5.26) as $\mathbf{J}^{(n)} = \mathbf{J}_s^{(n)} + \tilde{\mathbf{J}}_b^{(n-1)}$. In the computation of the $\mathbf{J}_s^{(n)}$, the averaging is done over sensors with some prior location, i.e, sensors which are beacons at the end of timeslot $n-1$. We assume the prior location distribution as a Gaussian with mean equal to their true location and covariance matrix as $\tilde{\mathbf{J}}_b^{(n-1)}$.

5.6 Incremental Beacon Selection

In the proposed methods for sensor self-localization, each sensor uses the prior information from all the beacons in a timeslot to obtain its position information. If a beacon has faulty or imprecise prior information, clearly incorporation of this measurements will cause a decrease in the accuracy of the sensor determining its location. In [63], the criterion for selecting sensors for localization is choosing beacon nodes which decrease the entropy, while in [64] each sensor's utility factor is obtained via a utility function using which sensors are selected. Here, we consider a simple variation of our methods proposed in earlier sections for incremental beacon selection.

Suppose that in a certain timeslot we have K beacons and in the next timeslot the sensor receives an additional set of B new beacons broadcasting their positions.

For each new beacon $n = 1 \cdots B$

- Collect measurement and prior information: Stack the old measurement set with the new beacon measurement and form the measurement vector $\mathbf{y}_s^n$. Similarly incorporate the new prior beacon information as discussed earlier.

- Solve for a small number of iterations to obtain a rough estimate of the sensors' and beacons location. Denote this new solution of sensor and beacon locations as $\hat{\boldsymbol{\ell}}_{K+1}^{n*}$.

- Compute the following metric for ranking the new beacons in terms of their contribution in improving the accuracy of the sensor location estimates:

$$\begin{aligned}\xi_{K+1}^n &= \frac{1}{K+1}[(\mathbf{y}_s^n - \mathbf{f}(\hat{\boldsymbol{\ell}}_{K+1}^{n*}))^\top \boldsymbol{\Sigma}_v^{-1,n}(\mathbf{y}_s^n - \mathbf{f}(\hat{\boldsymbol{\ell}}_{K+1}^{n*})) + \\ &\quad (\hat{\boldsymbol{\ell}}_{K+1}^{n*} - \boldsymbol{\ell}_{K+1}^0)^\top \boldsymbol{\Sigma}_l^{-1}(\hat{\boldsymbol{\ell}}_{K+1}^{n*} - \boldsymbol{\ell}_{K+1}^0)] \qquad (5.27)\end{aligned}$$

with $\boldsymbol{\ell}_{K+1}^0$ being the initial prior mean. The terms in the bracket refer to the residual error with the measurements and the prior information. The first term $\frac{1}{K+1}$ normalizes the total residual error with addition of a new beacon. This ensures that we can compare this normalized residual error accumulated in this timeslot with the normalized error at the previous timeslots. We denote the normalized residual error at the previous timeslot with K beacons as ξ_K^{min}.

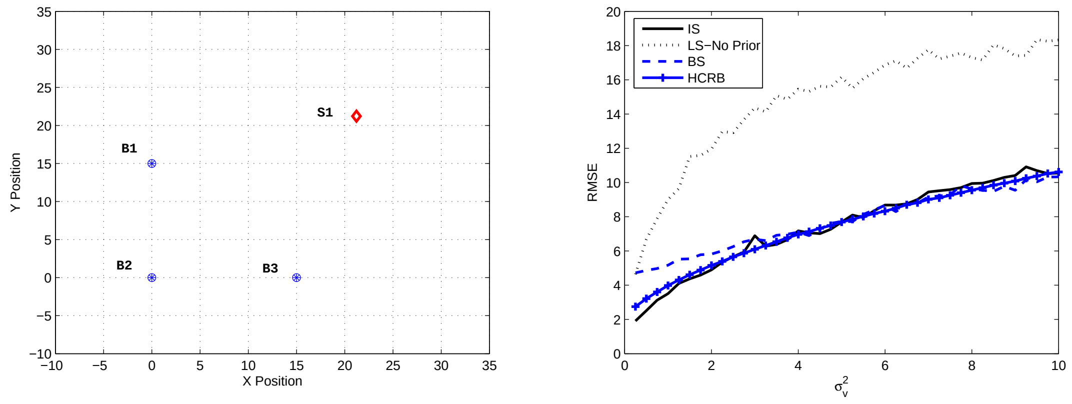
Selection Criterion: Choose the beacon with the minimum ξ_{K+1}^n and also if $\xi_{K+1}^n < \xi_K^{min}$. Having chosen this beacon, we can now perform a more rigorous search for the sensor's position.

5.7 Simulation Results

To assess the performance of the proposed algorithms, we performed several simulation experiments with the purpose of

- motivating the need of prior information as in the LS, BS, and IS methods for sensor localization.
- studying the performance of these algorithms for a large network, and
- studying the performance of the beacon selection algorithm.

5.7.1 Motivation



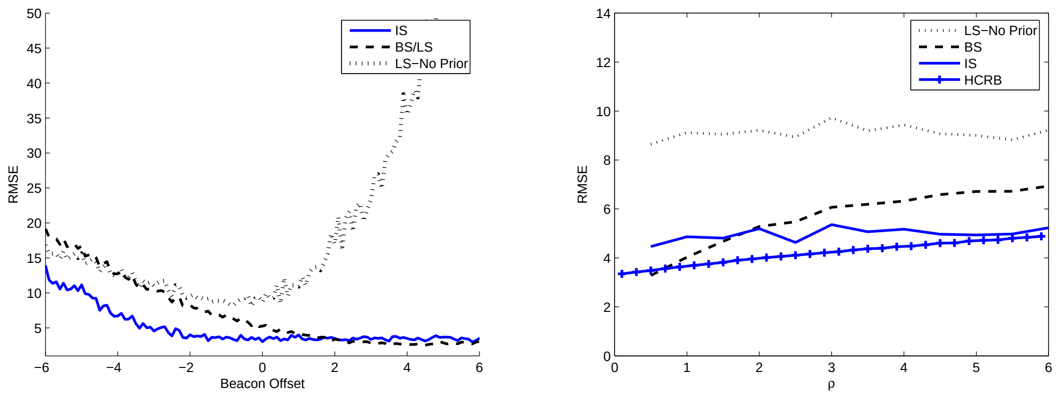
(a) Sensor Network. (b) RMSE

Figure 5.5: Sensor Network and RMSE using the LS, BS, IS methods with prior information and LS without prior information

We consider the sensor network as shown in Figure 5.5(a). The beacon nodes, $B1, B2$, and $B3$ with some prior information are represented with shaded circles and the sensor $S1$ with unknown location is denoted by a diamond mark. The beacons and sensor, $B1, B2, B3, S1$ are located at $(0, 15), (0, 0), (15, 0)$ and $(21.213, 21.213)$ respectively. We assumed a Gaussian prior location distributions $\mathcal{N}(\mu_{\mathbf{l}_b}, \rho\mathbf{I}_2)$ for the beacons position, where the prior mean is at offset b from its true position, i.e, $\mu_{\mathbf{l}_b} = [\mathbf{l}_{b,x} + \mathsf{b}, \mathbf{l}_{b,y} + \mathsf{b}]^\top$. For the LS and BS methods, the sensor position was initialized with the average of the three beacon mean locations. Recall that the LS and BS methods are iterative procedures. These procedures were run for a fixed number ($G = 50$) of iterations. Alternately, a simple stopping procedure for these methods is to compare the change in the residual error between successive iterations. For the IS method, in the construction of the sensor proposal distribution we ran the LS and BS method with fewer iterations $\acute{G} = 10$ and utilized this sensor estimate in the construction of the prior proposal distribution as outlined in Section 5.4.3 with $\tilde{\sigma_s} = 3$. The number of location samples drawn for each node was $M = 1000$.

In Figure 5.5(b), we plotted the root mean square errors (RMSEs) in estimating the sensors position with varying σ_v^2. In our simulations the reference power $\Psi_0 = -50dB$, the measurement noise variance σ_v^2 across all sensors was the same, the beacon offset was set to $\mathsf{b} = 0$ and the variance parameter to $\rho = 2$. We also simulated the LS algorithm with $\mathbf{\Sigma}_l^{-1} = \mathbf{0}$, i.e., we assumed that the mean location of the beacons was their true positions and did not take into account any prior information. We termed this method as LS-No Prior. Clearly the LS-No Prior has the worst performance because the other methods incorporated uncertainty in beacon position.

In another set of simulations, we studied the effect of the sensor position estimate by varying the offset of beacon $B2$. The beacons $B1$ and $B3$ had zero offsets. In Figure 5.6(a) we plotted the corresponding RMSEs with varying b_2. For this scenario, the



(a) Varying b (b) Varying ρ

Figure 5.6: RMSE using the LS,BS, IS method

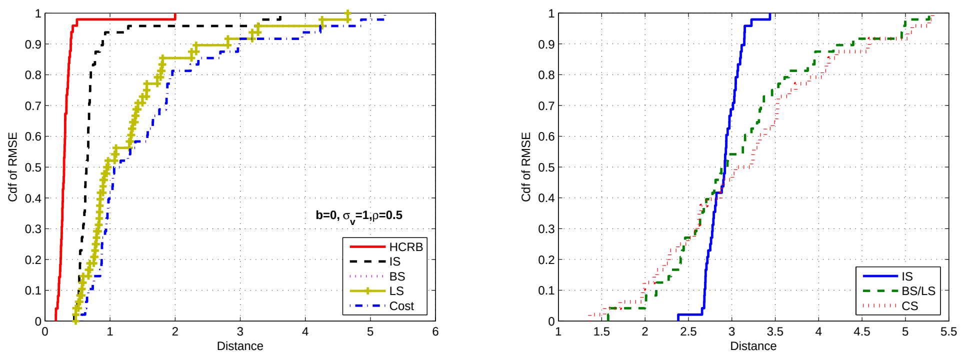
performance of the IS procedure is similar to that of the LS and BS methods when b_2 is positive. Also when the offset is positive, the LS-No Prior method had a large error in estimating the sensor's position. This is because with a reasonable positive offset, the three beacons appear collinear to the LS-No prior method thereby causing ambiguity for it to determine the sensors positions. When the offset is negative, for small values of b_2, the IS outperforms the LS/BS methods. Thus, within reasonable limits of beacon position offsets, the IS and LS/BS which take into account the beacon prior position information produce reliable sensor estimates. Clearly, we can see that the performance of the LS-No prior method is worse than the other procedures which incorporate prior information about the beacon's location.

5.7.2 Localization in a large network

In this experiment, we studied the performance of the importance sampling and cost-based methods for self-localization in large networks. To this end, we considered a network which consisted of 48 randomly distributed sensors with unknown locations and 16 beacons with some location information. As earlier, the prior location distributions of the beacon nodes were modeled using Gaussian distributions $\mathcal{N}(\boldsymbol{\mu}_{l_b}, \sigma^2 I)$ with $\boldsymbol{\mu}_{l_b} = [l_{b,x} + \mathsf{b}, l_{b,y} + \mathsf{b}]^\top$ where b represented the offset in the beacon location information. Through this network we studied the performance of the proposed algorithms for which we utilize the cumulative distributive function (CDF) of the RMSE as our metric.

We considered $K = 100$ different realizations of the measurements for this network over which the RMSEs in estimating the sensors positions were calculated. In the first set of simulations we studied the effect of b in determining the sensors' locations using the proposed methods. For this set of simulations we assumed $\rho = 0.5$, $\Psi = -50$dB, and $\alpha = 2.5$. The measurement noise across all the sensors was formulated as a Gaussian process with $\mu_v = 0$ and $\sigma_v = 1$. In 5.7(a) and 5.7(b), we plotted the CDFs of the RMSE for $\mathsf{b} = 0$ and $\mathsf{b} = 2$ obtained with the IS, LS, BS and CS methods. Further, to rank the performances of our methods we obtained the value of $d_{0.95}$ such that $\mathbb{P}(RMSE < 0.95) = d_{0.95}$. From Figure 5.7(a), $d_{0.95,IS} = 1.25$, $d_{0.95,BS} = 3.25$, $d_{0.95,CS} = 4.2$. Clearly, for these scenarios, the IS procedure had the best performance. In 5.7(b), it can be seen that with large offsets in beacon position from the true locations, the tails of the CDFs of the BS and CS algorithms are more towards the left than for the IS algorithm, suggesting that the best performance of the IS is lower than the best performance of the other algorithms. However, if one considers the $d_{0.95}$ performance metric, the IS procedure produces a better RMSE characteristic on an average for all the sensors.

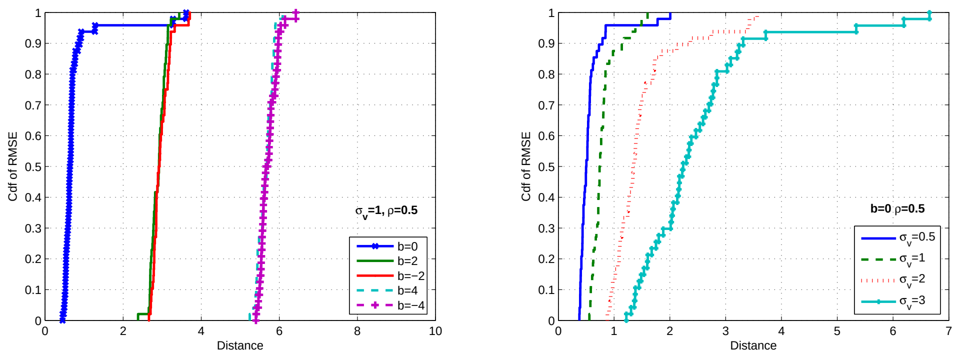
In Fig 5.8(a), we show more results of our study of the IS procedure. We ran it



(a) $\mathsf{b} = 0$ (b) $\mathsf{b} = 2$

Figure 5.7: CDF of RMSEs using the IS,LS,BS and CS methods.

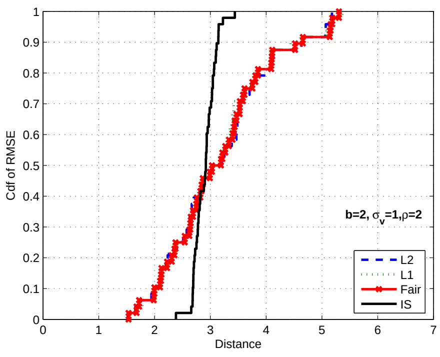
for various positive and negative values of b. As can be seen, the method is robust to the sign of the offsets. In 5.8(b), we varied the standard deviation of the measurement noise and obtained the corresponding RMSEs. As expected, with increasing noise, the CDF curve shifts to the right indicating increase in estimation error.



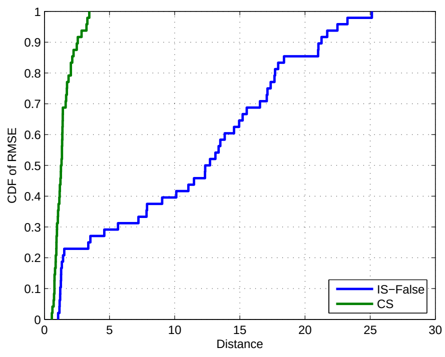
(a) RMSE with various b (b) RMSE with various σ_v

Figure 5.8: CDF of RMSEs using the IS.

We also studied the performance of the CS methods in comparison to the IS methods. In 5.9(a), we plotted the performance of the IS and CS method with $L2$, $L1$ and Fair cost functions. The performance of the CS method which makes no assumptions of the noise distributions is comparable with the performance of the IS method. Furthermore, we conducted another set of simulations to study the robustness of the algorithms when we do not have information about the distributions. The probabilistic IS algorithm assumed wrong measurement noise distribution $\hat{p}(v) = \mathcal{N}(0, 0.1^2)$ while the true distribution of the noise was $p(v) = 0.8\mathcal{N}(0,1) + 0.2\mathcal{N}(3, 0.2^2)$. When the IS method made wrong assumptions, it had a poorer performance than the CS method. Clearly, Figure 5.9(b) shows the robustness of the CS algorithms and the sensitivity of the probabilistic algorithm on the knowledge of the distributions of the noises.



(a) L2, L1 and Fair cost function



(b) CS and IS with false assumptions

Figure 5.9: CDF of RMSE using the CS and IS methods.

5.7.3 Beacon Selection

We consider the scenario shown in Figure 5.10, to understand the importance and the effect of beacon selection. In this network there are 5 nodes. Beacon nodes $B1, B2, B3$ and $B5$ have zero offsets in their position with covariance matrix $\rho\mathbf{I}_2$, $\rho = 2$. The sensor is at the center of circle radius $R = 15$, beacons $B1, B2, B3$ are on

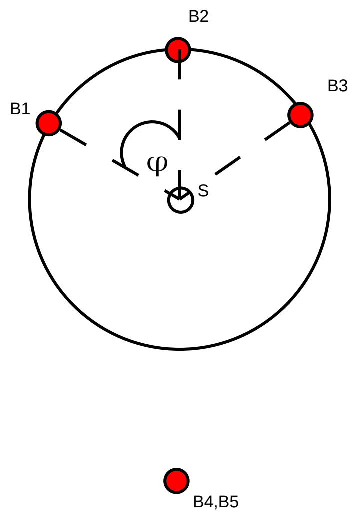
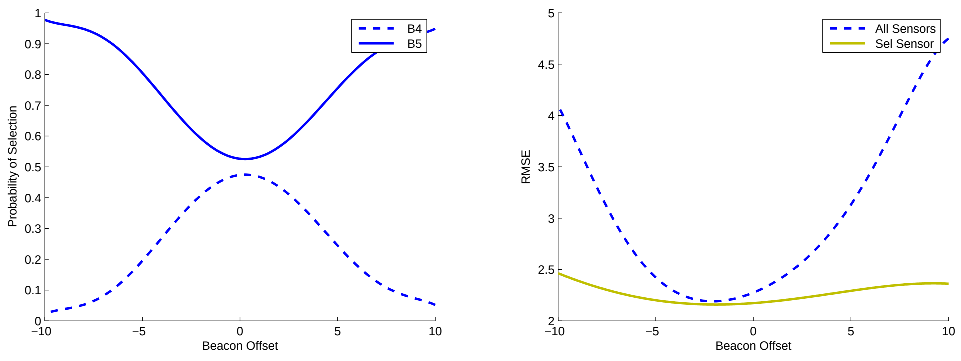


Figure 5.10: Beacon Selection

this circle with $\phi = \frac{\pi}{6}$ and the beacons $B4, B5$ which are at the same position, are at a distance of $2R$ from the sensor. The variance of the noise $\sigma_v^2 = 1.0$. Initially the sensor chooses the three beacons which have the maximum signal strength, performs a localization using the proposed localizations methods. It then selects either beacon $B4$ or $B5$ using the criteria which we have discussed in 5.6. In Figure 5.11(a), we plot the probability of the number of times, the algorithm selects either of the sensors with varying beacon offset for beacon node $B4$. Clearly one can notice that as the beacon offset for $B4$ goes beyond 0, the algorithm selects node $B5$ more often. We plot the corresponding RMSE in estimating the sensor position vs beacon offset in Figure 5.11(b) with beacon selection and also when all the beacons are considered for localization. Here we can notice that at large beacon offsets, the accuracy of the sensor location decreases when we consider all the sensors for localization. However by suitably selecting the beacons the sensor location accuracy is maintained. Thus this process of beacon selection can be easily incorporated to also detect beacon nodes



(a) Selection Probability. (b) RMSE

Figure 5.11: Selection probability and RMSE

with incorrect position estimates.

5.8 Summary

We proposed distributed algorithms for sensor self-localization with beacon position uncertainty. Being distributed they scale well for large networks and have considerable savings in power over centralized methods. The proposed algorithms can be classified as iterative least squares (LS) and Bayesian methods (BS), Monte Carlo importance sampling (IS) and cost-based (CS) methods. The iterative LS and Monte Carlo CS methods do not require knowledge of the model noise distributions while the iterative BS and Monte Carlo IS do require this knowledge. Through computer simulations we have observed the performance of the IS method over a wide range of such scenarios to be reasonable. However when the IS method makes wrong assumptions about the statistics of the measurement noise, its performance can be degraded considerably. In such scenarios, the cost-based approach has relatively good performance.

s

Chapter 6

Target tracking by particle filtering in binary sensor networks

Particle filtering algorithms for tracking a single target using data from binary sensors are proposed. The sensors transmit signals that identify them to a central unit if the target is in their neighborhood; otherwise they do not transmit anything. The central unit uses a model for the target movement in the sensor field and estimates the target's trajectory, velocity, and power using the received data. In Section 6.3, we describe the binary sensor network and define the addressed problem in a mathematical form. We propose tracking by employing auxiliary particle filtering and cost-reference particle filtering in Section 6.4. We also extend the method to include estimation of constant parameters, and we derive the posterior Cramér-Rao bounds for the states in Section 6.6. We show the performances of the proposed methods by extensive computer simulations in Section 6.7 and compare them to the derived bounds.

6.1 Introduction

Recent advances in low-power micro sensors, actuators, embedded sensors, radio, and in general, digital wireless communication technologies have allowed development of wireless sensor networks with unparalleled capabilities [65]. Their use may span a vast range of fields, and their effectiveness is already being felt both in commercial and military applications as well as in the further development of science and engineering [66, 67, 68].

In this chapter we consider wireless sensor networks with two basic components, sensors and a fusion center (FC). The sensors sense and measure signals that provide information about an event or events of interest and send binary information to the FC. The FC combines the received information to obtain estimates about the observed phenomenon. Here the object of interest is a single target that moves in a field of sensors that measure signal power, and the objective is the online estimation of its trajectory, velocity, and power.

Often sensor networks have to operate with sensors that have limited power as well as limited communication and computational resources. At a strategic assessment workshop organized by the U.S. Army Research Lab it was concluded that [69] "it is not practical to rely on sophisticated sensors with large power supply and communication [demands]. Simple, inexpensive individual devices deployed in large numbers are likely to be the source of the battlefield awareness in the future." One type of networks that fits this description is the class of binary sensor networks, where the sensors transmit only binary information about sensed events (the event is sensed or is not sensed). The signals that reach the FC of these networks are therefore highly compressed and pose challenging problems for recovering the sensed information by the sensors.

The sensed information here is the intensity (signal power) of the transmitted signals by one or more sources that attenuates as a function of distance from the sources.[1] For the tracking we propose to apply the auxiliary particle filter (APF) [70] and the cost-reference particle filter (CRPF) [35, 34]. The two methods are sequential statistical signal processing procedures with distinct features but with similar algorithmic outline. Convergence properties of the particle filtering methods can be found in [71] and [72] and of the CRPF in [73] and [34]. The main contribution is in the application of the APF and CRPF on binary signals and in the presence of hidden complete measurements by the sensors. We show how the signals from the binary sensors contribute to producing evolving random grids of the methods, how the associated metrics of the grid nodes are updated, and how the estimates of the unknowns are obtained. We also outline the steps for computing the posterior Cramér-Rao bounds of the state estimates.

6.2 A brief literature survey

Binary sensors have already been addressed in the wide literature. In [74], a cooperative tracking method based on acoustic binary sensors was described. There, the tracking is based on a cooperative algorithm where the model of the target path is a piece-wise linear curve and is different from the one that we use. In [75], a particle filtering-type algorithm for tracking was introduced. However, the authors focus on geometric properties of the sensors' configuration and derive their algorithm based

[1]We note that binary sensors of other sensing modalities can be used analogously along the lines presented here.

on that geometry. Also, the method is based on sensor data that provide information about approaching or receding targets with respect to the individual sensors, whereas in our approach we only use information about the proximity of the target to the sensor. In [76], a tracking method was proposed that first estimates the positions of a target in its most recent past and then fits them with a piece-wise trajectory. In [77], another method for distributed tracking in binary sensor networks was developed. It is derived by using hidden state estimation and the Viterbi algorithm.

In other recent publications on signal processing for binary sensor networks, various problems have been addressed. For example, in [78] acoustic binary sensors were used for maximum likelihood localization (not tracking) of a source, and in [79] they were applied for system identification. Decentralized detection by binary sensors was presented in [80] where the sensors use a multiple access channel of limited capacity. Sufficient conditions were found that allow for minimal probability of error at the FC.

6.3 Network description and mathematical models

6.3.1 Network description

In a binary sensor network, the deployed sensors measure a signal of interest and if the level of the measured signal is above a predefined threshold, they report to the FC with a signal that identifies them; otherwise they are silent. Their function is best described by a way of example. In Figure 6.1, a binary sensor is represented by the small circle and its range by the larger circle. When the target is outside the range of the sensor, the received signal is below the set threshold, and the sensor

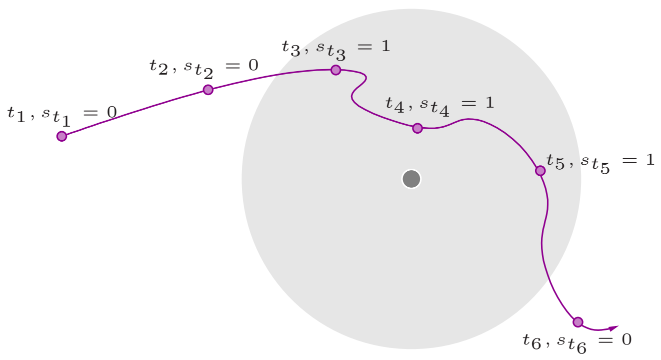


Figure 6.1: A binary sensor with a target passing nearby. The signals transmitted by the sensors are denoted by s_{t_k}.

does not transmit anything (instants t_1, t_2 and t_6). During the time when the target is inside the range of the sensor, the received signal is above the threshold, and the sensor transmits a "one" to the FC (instants t_3, t_4, and t_5). When at a given time the FC does not receive a signal from a particular sensor, this implies that the sensor transmits a "zero."

The network consists of N binary sensors that may be deployed randomly, deterministically, or both. In all cases, we assume that the FC knows the locations of all the sensors and that the locations remain fixed for all time.

6.3.2 Mathematical models

The model for target movement is standard and is described by [81]

$$\boldsymbol{x}_t = \mathbf{G}_x \boldsymbol{x}_{t-1} + \mathbf{G}_u \boldsymbol{u}_t \tag{6.1}$$

where $\boldsymbol{x}_t = [x_{1,t} \;\; x_{2,t} \;\; \dot{x}_{1,t} \;\; \dot{x}_{2,t}]^\top \in \mathbb{R}^4$ is a state vector, which indicates the position and the velocity of the target in a two-dimensional Cartesian coordinate system, $\mathbf{G}_x$

and $\mathbf{G}_u$ are known matrices given by

$$\mathbf{G}_x = \begin{pmatrix} 1 & 0 & T_s & 0 \\ 0 & 1 & 0 & T_s \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{G}_u = \begin{pmatrix} \frac{T_s^2}{2} & 0 \\ 0 & \frac{T_s^2}{2} \\ T_s & 0 \\ 0 & T_s \end{pmatrix}$$

with T_s being the sampling period and $\boldsymbol{u}_t$, a 2×1 zero-mean vector representing the state noise process and which accounts for the acceleration of the target. The APF method uses an *additional* assumption about $\boldsymbol{u}_t$; it is a *Gaussian* vector with a covariance matrix $\mathbf{C}_u = \text{diag}(\sigma_{u_1}^2, \sigma_{u_2}^2)$.

The received power can be modeled as in [82] or as in [83]. We adopt the latter model, that is, the measurement of the $n-$th sensor is given by

$$\begin{aligned} y_{n,t} &= g_n(\boldsymbol{x}_t) + v_{n,t} && (6.2) \\ &= \frac{\Psi d_0^\alpha}{||\mathbf{r}_n - \mathbf{l}_t||^\alpha} + v_{n,t}, \quad n = 1, 2, \cdots, N && (6.3) \end{aligned}$$

where $g_n(\cdot)$ is a function that models the received signal power by the $n-$th sensor; $v_{n,t}$ is a noise process independent from $\boldsymbol{u}_t$; $\mathbf{r}_n \in \mathbb{R}^2$ is the position of the $n-$th sensor; $\mathbf{l}_t = [x_{1,t}\, x_{2,t}]^\top$ is the location of the target at time t; $||\mathbf{r}_n - \mathbf{l}_t||$ denotes the Euclidean distance between $\mathbf{r}_n$ and $\mathbf{l}_t$; Ψ is the emitted power of the target measured at a reference distance d_0; α is an attenuation parameter that depends on the transmission medium and is considered to be known and the same for all sensors. For the application of the APF, we need to know the distribution of $v_{n,t}$. In our model, we assume that $v_{n,t} \sim \mathcal{N}(\mu_v, \sigma_v^2)$ where $\mu_v = \sigma^2$ with σ^2 being the known power of the background measurement noise of one sample and $\sigma_v^2 = 2\sigma^4/L$, with L being the number of samples used to obtain the measured power. For the CRPF, we only assume that we know μ_v.

The $n-$th sensor ($n = 1, \ldots, N$) measures the received power $y_{n,t}$, processes it locally and transmits a single binary digit to the FC according to the following rule:

1. The sensor compares the actual observed power level, $y_{n,t}$, with a threshold, γ. If the sensed value is below γ, it does not transmit anything.

2. If the sensed value is greater than γ, the sensor transmits its identification code to the FC.

Therefore, the sensors in the network send signals to the FC *only* if the received power is greater than the sensor thresholds.

The received signal from the $n-$th sensor at the FC is modeled as

$$z_{n,t} = \beta_n s_{n,t} + \epsilon_{n,t} \tag{6.4}$$

where

$$s_{n,t} = \begin{cases} 1 & \text{if} \quad y_{n,t} > \gamma \\ 0 & \text{if} \quad y_{n,t} < \gamma \end{cases} \tag{6.5}$$

and where $\epsilon_{n,t}$ is the observation noise, and β_n is a *known* attenuation coefficient associated with the $n-$th sensor. Again, for the APF method we need the knowledge of the noise distribution of $\epsilon_{n,t}$, and we let $\epsilon_{n,t} \sim \mathcal{N}(0, \sigma_\epsilon^2)$, whereas for the CRPF method we only assume that the noise is zero mean.

In summary, the measurements made by the sensors are complete and are modeled by (6.3). The sensors, however, always transmit binary signals constructed according to (6.5), and the FC receives them as quantified by (6.4).

The objective is to track the evolving state $\boldsymbol{x}_{0:t} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_t)$ using the observations $\boldsymbol{z}_{1:t} = (z_{1,1:t}, \ldots, z_{N,1:t})$, that is, the observations up to time instant t of the first sensor, $z_{1,1:t}$, the second sensor, $z_{2,1:t}$, as well as the remaining $N-2$ sensors, $z_{3,1:t}, \cdots, z_{N,1:t}$. The APF uses probabilistic assumptions about all the noises in the model and about the prior of the states. The CRPF only needs knowledge of the first moments of the noises.

6.4 Tracking algorithms

First we present a tracking algorithm based on APF [30, 70] (subsection 6.4.1), and then we proceed with the presentation of a CRPF algorithm [35, 34] (subsection 6.4.2). In this section, the parameter Ψ is assumed known, but subsequently this assumption will be dropped.

6.4.1 APF algorithm

Recall that according to the theory of particle filtering, we track the a posteriori distribution of $\mathbf{x}_{0:t}$, $p(\boldsymbol{x}_{0:t} \mid \boldsymbol{z}_{1:t})$, by approximating it with a random measure, χ_t, composed of particles $x_t^{(m)}$ and weights $w_t^{(m)}$, where m is an index, and which we denote by $\chi_t = \{\boldsymbol{x}_{0:t}^{(m)}, w_t^{(m)}\}_{m=1}^{M}$ with M being the number of particles. At every time instant t, the particle filter carries out the following operations: (1) Selection of most promising particle streams, (2) particle propagation, (3) computation of particle weights, and (4) state estimation.

The APF attempts to draw from an importance function which is as close as possible to the optimal one. To that end, the selection of most promising particles is carried out by sampling from a multinomial distribution where the number of possible outcomes is M and the probabilities of the respective outcomes are $\tilde{w}_t^{(m)}$, $m = 1, 2, \cdots, M$, and

$$\tilde{w}_t^{(m)} \propto p(\boldsymbol{z}_t \mid \boldsymbol{\mu}_t^{(m)}) w_{t-1}^{(m)} \tag{6.6}$$

where $\boldsymbol{\mu}_t^{(m)}$ is some parameter that characterizes $\boldsymbol{x}_t^{(m)}$ given $\mathbf{x}_{t-1}^{(m)}$.

Since the noise samples $\epsilon_{n,t}$ from (6.4) are assumed independent, we have

$$p(\boldsymbol{z}_t \mid \boldsymbol{\mu}_t^{(m)}) = \prod_{n=1}^{N} p(z_{n,t} \mid \boldsymbol{\mu}_t^{(m)}). \tag{6.7}$$

For the factors $p(z_{n,t} \mid \boldsymbol{\mu}_t^{(m)})$, we can write

$$\begin{aligned} p(z_{n,t} \mid \boldsymbol{\mu}_t^{(m)}) &= p(z_{n,t} \mid s_{n,t} = 0, \boldsymbol{\mu}_t^{(m)}) P(s_{n,t} = 0 \mid \boldsymbol{\mu}_t^{(m)}) + \\ &\qquad p(z_{n,t} \mid s_{n,t} = 1, \boldsymbol{\mu}_t^{(m)}) P(s_{n,t} = 1 \mid \boldsymbol{\mu}_t^{(m)}) \\ &= p(z_{n,t} \mid s_{n,t} = 0) P(s_{n,t} = 0 \mid \boldsymbol{\mu}_t^{(m)}) + \\ &\qquad p(z_{n,t} \mid s_{n,t} = 1) P(s_{n,t} = 1 \mid \boldsymbol{\mu}_t^{(m)}) \end{aligned} \tag{6.8}$$

where

$$p(z_{n,t} \mid s_{n,t}) = \mathcal{N}(\beta_n s_{n,t}, \sigma_\epsilon^2) \tag{6.9}$$

and

$$P\left(s_{n,t} = 1 \mid \boldsymbol{\mu}_t^{(m)}\right) = Q\left(\frac{\gamma - g_n(\boldsymbol{\mu}_t^{(m)})}{\sigma_v}\right) \tag{6.10}$$

$$P\left(s_{n,t} = 0 \mid \boldsymbol{\mu}_t^{(m)}\right) = 1 - Q\left(\frac{\gamma - g_n(\boldsymbol{\mu}_t^{(m)})}{\sigma_v}\right) \tag{6.11}$$

where $Q(\cdot)$ denotes the standard normal complementary cumulative distribution function.

At the beginning, the initial set of particles $\boldsymbol{x}_0^{(m)}$, $m = 1, 2, \cdots, M$, are drawn from a prior distribution $\pi(\boldsymbol{x}_0)$, and the weights of the particles are set to $\frac{1}{M}$. Suppose now that at time instant $t-1$, we have the random measure $\chi_{t-1} = \{\boldsymbol{x}_{0:t-1}^{(m)}, w_{t-1}^{(m)}\}_{m=1}^{M}$. Then the steps of a particle filter recursion can be implemented as follows:

1. *Selection of most promising particle streams:*

For selection of the most promising particles, we use the conditional mean of $\boldsymbol{x}_t^{(m)}$ given $\boldsymbol{x}_{t-1}^{(m)}$ as a characterizing parameter of every stream, i.e.,

$$\boldsymbol{\mu}_t^{(m)} = E\big(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}^{(m)}\big). \tag{6.12}$$

The conditional means are computed readily from

$$\boldsymbol{\mu}_t^{(m)} = \mathbf{G}_x \boldsymbol{x}_{t-1}^{(m)}. \tag{6.13}$$

This is followed by computation of the weights according to (6.6) and their normalization. Finally, a set of indices $\{i_m\}$ are drawn from the probability mass function (pmf) represented by the normalized weights.

2. *New particle generation*:

The first two elements of the four-dimensional state $\boldsymbol{x}_t$ represent the location of the target in a two-dimensional space, and the remaining elements are the components of the velocity in this space. That implies that the generation of $\boldsymbol{x}_t^{(m)}$ requires drawing only two-dimensional random variables. The generation can be carried out, for example, by first, propagating the velocity components one step ahead using the joint distribution $p(\dot{x}_{1,t}, \dot{x}_{2,t} \,|\, \dot{x}_{1,t-1}, \dot{x}_{2,t-1})$ or $p(\dot{x}_{1,t}, \dot{x}_{2,t} \,|\, \dot{x}_{1,t-1}, \dot{x}_{2,t-1}, \boldsymbol{z}_t)$ and second, computing the locations according to

$$x_{1,t}^{(m)} = x_{1,t-1}^{(i_m)} + \frac{T_s}{2}\left(\dot{x}_{1,t}^{(m)} + \dot{x}_{1,t-1}^{(i_m)}\right) \tag{6.14}$$

$$x_{2,t}^{(m)} = x_{2,t-1}^{(i_m)} + \frac{T_s}{2}\left(\dot{x}_{2,t}^{(m)} + \dot{x}_{2,t-1}^{(i_m)}\right). \tag{6.15}$$

3. *Weight computation*:

The newly generated particles are assigned weights according to

$$w_t^{(m)} \propto \frac{p(\boldsymbol{z}_t \mid \boldsymbol{x}_t^{(m)})}{p(\boldsymbol{z}_t \mid \boldsymbol{\mu}_t^{(i_m)})}.$$

The likelihood terms of the numerator and denominator are calculated as in the APF using (6.7) – (6.11).

4. *State estimation*:

Once the weights are normalized, one can use χ_t to compute estimates of the unknown states. For example, the minimum mean square error (MMSE) estimate is obtained from

$$\hat{\boldsymbol{x}}_t = \sum_{m=1}^{M} w_t^{(m)} \boldsymbol{x}_t^{(m)}. \tag{6.16}$$

6.4.2 CRPF algorithm

The objective of CRPF is to estimate sequentially the evolution of the unknown state $\boldsymbol{x}_{0:t}$ from $\boldsymbol{z}_{1:t}$ without assumptions about the probability distributions of the noises in the model. It is similar in structure to that of the APF because CRPF, too, uses a discrete random measure. This random measure is composed of particles and costs associated to them, where the costs are user-defined. We denote the random measure by $\zeta_t = \left\{ \boldsymbol{x}_{0:t}^{(m)}, \mathcal{C}_t^{(m)} \right\}_{m=1}^{M}$, where $\boldsymbol{x}_{0:t}^{(m)}$ has the same meaning as before and $\mathcal{C}_t^{(m)}$ are the associated costs to $\boldsymbol{x}_{0:t}^{(m)}$. It is clear that the costs here play the role of the weights in APF. In fact, with appropriate choices of the costs, one can make the CRPF equivalent to the APF or the standard particle filter [35].

In general, the costs are updated according to [34]

$$\mathcal{C}_t^{(m)} = \mathcal{C}(\boldsymbol{x}_{0:t}^{(m)} | \boldsymbol{z}_{1:t}) \tag{6.17}$$

$$= \lambda \mathcal{C}(\boldsymbol{x}_{0:t-1}^{(m)} | \boldsymbol{z}_{1:t-1}) + \triangle \mathcal{C}(\boldsymbol{x}_t^{(m)} | \boldsymbol{z}_t) \tag{6.18}$$

where λ is a forgetting factor ($0 \leq \lambda \leq 1$), and $\triangle \mathcal{C}(\boldsymbol{x}_t^{(m)} | \boldsymbol{z}_t)$ is an incremental cost. Obviously, the value of λ controls how fast the state estimates can adapt to new values of the states. The incremental cost contributes to the total cost at time instant t and is a function of the particle value and the observation at that instant. A typical incremental cost has the form

$$\triangle\mathcal{C}(\boldsymbol{x}_t^{(m)}|\boldsymbol{z}_t) = ||\boldsymbol{z}_t - \hat{\boldsymbol{z}}_t^{(m)}||^q \tag{6.19}$$

where $\hat{\boldsymbol{z}}_t^{(m)}$ is a function of $\boldsymbol{x}_t^{(m)}$, and $q > 0$. So, CRPF proceeds analogously to the APF; with the vector of observations $\boldsymbol{z}_t$, the discrete random measure at time instant $t-1$, $\zeta_{t-1} = \left\{\boldsymbol{x}_{0:t-1}^{(m)}, \mathcal{C}_{t-1}^{(m)}\right\}_{m=1}^{M}$, is updated to $\zeta_t = \left\{\boldsymbol{x}_{0:t}^{(m)}, \mathcal{C}_t^{(m)}\right\}_{m=1}^{M}$ to reflect accurately the possible value of the unknown state at time instant t, $\boldsymbol{x}_t$.

The procedure has four steps: (1) selection of most promising particle streams, (2) propagation of particles, (3) cost update, and (4) state estimation. The initialization of the method is carried out by randomly drawing initial particles from some probability density function (pdf) $\pi(\boldsymbol{x}_0)$ whose support includes the space of $\boldsymbol{x}_0$. We propose that the method is implemented as follows:

1. *Selection of most promising particle streams:*

This step is reminiscent of the main idea of the APF, where resampling at time instant $t-1$ takes place by using measurements from time instant t. For CRPF, we define a risk function, $\mathcal{R}(\boldsymbol{x}_{t-1}^{(m)}|\boldsymbol{z}_t)$, which quantifies the quality of the particle $\boldsymbol{x}_{t-1}^{(m)}$ given the next set of observations, $\boldsymbol{z}_t$. As a risk function, one can use the incremental cost, that is, in our case

$$\mathcal{R}(\boldsymbol{x}_{t-1}^{(m)}|\boldsymbol{z}_t) = \triangle\mathcal{C}\left(\hat{\boldsymbol{x}}_t^{(m)}|\boldsymbol{z}_t\right)$$

with

$$\hat{\boldsymbol{z}}_t^{(m)} = h(\hat{\mathbf{y}}_t^{(m)}) \tag{6.20}$$

$$\hat{\mathbf{y}}_t^{(m)} = g(\hat{\boldsymbol{x}}_t^{(m)}) + \boldsymbol{\mu}_v \tag{6.21}$$

$$\hat{\boldsymbol{x}}_t^{(m)} = G_x \boldsymbol{x}_{t-1}^{(m)} \tag{6.22}$$

where the elements of $g(\cdot)$ are defined by (6.3) and those of $h(\cdot)$ by (6.4) and (6.5).

Once the risks are computed, they are added to their costs at $t-1$ to obtain the predicted costs, i.e.,

$$\hat{\mathcal{C}}_t^{(m)} = \lambda \mathcal{C}(\boldsymbol{x}_{0:t-1}^{(m)} | \mathbf{z}_{1:t-1}) + \mathcal{R}(\boldsymbol{x}_{t-1}^{(m)} | \boldsymbol{z}_t). \tag{6.23}$$

The particles are then sorted according to their predicted costs $\hat{\mathcal{C}}_t^{(m)}$ in ascending order and the first L of the M particles are replicated $J = \frac{M}{L}$ times (we assume that J is an integer). In other words, each of the surviving particles will have J children at time instant t [35]. We note that in previous versions of CRPF implementations we used functions to generate pmfs, where the latter were subsequently used for resampling of the particles. With the sorting scheme, we avoid the use of such functions and classical resampling and instead, we proceed directly with removing "bad" particles. Many simulation results have shown that with this simpler and faster scheme we do not sacrifice performance.

2. *Particle propagation:*

For particle propagation we can use a Gaussian proposal density in a similar way as is done with APF. Note that the functional form of the Gaussian is not used for computing the costs of the particles. Also, the use of a Gaussian is not required, and we can use any other density that is centered around the particle and that produces random variables with appropriate variance, like a uniform, or a Laplace, or a Cauchy density.

If we use a Gaussian, we draw the velocities of the target with mean $\dot{\boldsymbol{x}}_{t-1}^{(i_m)} = [\dot{x}_{1,t-1}^{(i_m)}\ \dot{x}_{2,t-1}^{(i_m)}]^\top$ and with covariance matrix $\sigma_{t-1}^{2,(i_m)}\mathbf{I}_{2\times 2}$, where the i_ms denote indexes of sorted particles, and $\dot{\mathbf{x}}_{t-1}^{(i_m)}$'s are *surviving* particles from step 1. The variance, $\sigma_{t-1}^{2,(i_m)}$, is recursively updated by (see [34])

$$\sigma_{t-1}^{2,(i_m)} = \frac{t-2}{t-1}\sigma_{t-2}^{2,(i_m)} + \frac{||\dot{\boldsymbol{x}}_{t-1}^{(i_m)} - \dot{\boldsymbol{x}}_{t-2}^{(i_m)}||^2}{2(t-1)}.$$

The locations are then obtained by (6.14) and (6.15).

3. *Cost update:*

The cost update is performed by using (6.18).

4. *State estimation:*

The state is estimated by using the particles and the associated costs. One way of carrying out the estimation is by creating a pmf from the costs. To that end, a monotonically decreasing function $\eta(\cdot)$ that converts the set of costs into probability masses $\pi_{c_t}^{(m)}$ is defined, that is,

$$\pi_{c_t}^{(m)} \propto \eta\left(\mathcal{C}_t^{(m)}\right).$$

For example, one function that has worked well for different problems is

$$\eta\left(\mathcal{C}_t^{(m)}\right) = \frac{1}{\left(\mathcal{C}_t^{(m)} - \min(\mathcal{C}_t) + 1/M\right)^2}.$$

Once the $\pi_{\mathcal{C}_t}^{(m)}$, $m = 1, 2, \cdots, M$ are computed, the estimation of $\boldsymbol{x}_t$ can be carried out readily.

6.5 Extension of the APF when Ψ is unknown

The particle filters described in the previous section are based on the assumption that in (6.3) the emitted power, Ψ, of the target measured at a reference distance is *known*. When it is unknown and varies with time randomly, the presented particle filters are modified straightforwardly so that Ψ_t becomes an element of the state vector $\boldsymbol{\xi}_t$, i.e., $\boldsymbol{\xi}_t = [x_{1,t} \;\; x_{2,t} \;\; \dot{x}_{1,t} \;\; \dot{x}_{2,t} \;\; \Psi_t]^\top$, and the state-space equation (6.1) is changed to reflect the evolution of Ψ_t with time.

In our model, Ψ is constant. It is well known that a special care must be taken for the estimation of constant parameters by particle filtering. Estimation of static parameters by particle filtering has already been addressed in the literature, for example, [84], [85], and [86]. The approach that we propose here exploits the concept behind Gaussian particle filtering [87]. CRPF *does not* have the problems of the APF in estimating constant parameters because it is *not* based on the use of probability distributions.

Let the parameter Ψ be denoted by Ψ_t even though it does not change with time.

Formally, we have

$$\Psi_t = \Psi_{t-1} \tag{6.24}$$

which implies that the prior proposal distribution of Ψ should be

$$p(\Psi_t \,|\, \Psi_{t-1}^{(m)}) = \delta(\Psi_t - \Psi_{t-1}^{(m-1)}). \tag{6.25}$$

The particle filter from the previous section remains the same except for the following modification: at time $t-1$, one approximates the marginal posterior of Ψ_{t-1} with a Gaussian (or some other) distribution [85]. If it is a Gaussian distribution, we compute its parameters (the mean and variance of Ψ_{t-1}) from $\Psi_{t-1}^{(m)}$ by

$$\begin{aligned}
\mu_{\Psi_{t-1}} &= \sum_{m=1}^{M} w_{t-1}^{(m)} \Psi_{t-1}^{(m)} \\
\sigma^2_{\Psi_{t-1}} &= \sum_{m=1}^{M} w_{t-1}^{(m)} \left(\Psi_{t-1}^{(m)} - \mu_{\Psi_{t-1}}\right)^2 .
\end{aligned} \tag{6.26}$$

Then we sample from $\mathcal{N}\left(\mu_{\Psi_{t-1}}, \sigma^2_{\Psi_{t-1}}\right)$, i.e., $\Psi_t^{(m)} \sim \mathcal{N}(\mu_{\Psi_{t-1}}, \sigma^2_{\Psi_{t-1}})$. The drawn particles $\Psi_t^{(m)}$ are particles of Ψ_t since $\Psi_t = \Psi_{t-1}$, and the remaining elements of the particle $\boldsymbol{x}_t^{(m)}$ are generated in the same way as in the previous section.

The CRPF method applies the same method. For finding the parameters of the Gaussian, it uses the pmf determined in step 4.

6.6 Posterior Cramér-Rao Bounds

The posterior Cramér-Rao bounds (PCRBs) provide in general the lower bound for mean square errors (MSEs) [60]. In our problem, the PCRBs represent the lower bounds of the MSEs of the estimated unknown position, velocity, and emitted power of the target. In particular, the covariance matrix of the estimation errors of $\boldsymbol{x}_t$, $\mathbf{C}_t$, has a lower bound, i.e.,

$$\mathbf{C}_t \triangleq E\left\{\left(\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t\right)\left(\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t\right)^\top\right\} \geq \mathbf{J}_t^{-1} \tag{6.27}$$

where $\mathbf{J}_t$ is the filtering information matrix, whose inverse is the PCRB of $\mathbf{x}_t$.

In the context of sensor networks, the PCRBs provide insights into various issues including the following:

- the accuracy attainable in estimating the dynamics of the target,
- the effect of the deployment geometry of the sensors on the PCRB, and
- the effect of the sensing properties of the sensor on the achievable accuracy.

When the dynamics of the state evolution of the target are given by (6.1), the prior distribution $p(\boldsymbol{x}_t \,|\, \mathbf{x}_{t-1})$ is not defined because this conditional distribution becomes singular. In [88], a recursive method for determining the PCRBs for such cases was described, and we followed that approach. We also used the same notation as in [88] to allow for the more compressed presentation that is given in the Appendix. The obtained PCRBs do not have analytical expressions in closed form but they can be computed using Monte Carlo simulation methods.

6.7 Simulations

We present some computer simulations that illustrate the performances of the proposed algorithms. We considered a scenario where the examined network consisted of $N = 264$ sensors deployed in a field with dimensions 800×500 m^2. The attenuation parameter was set to $\alpha = 2.5$ and the reference power parameter to $\Psi = 5,000$. The parameter Ψ was assumed unknown throughout the simulations and was also estimated. The sensing radius of the sensors, which depends on the threshold γ and the power parameter Ψ, was 25m. The corresponding threshold was set to 2. The covariance matrix of the state noise process was $\mathbf{C}_u = \text{diag}\{0.05, 0.01\}$, and the measurement noise in (6.3) had mean $\mu_v = 1$ were $\sigma_v^2 = 0.01$, and $\sigma_\epsilon^2 = 0.01$. The noise variance σ_ϵ^2 and the parameters β_n, unless otherwise stated, were chosen to yield signal-to-noise ratio (SNR) of 20dB at the FC. The sampling interval was $T_s = 1$s.

In the implementation of the APF and CRPF algorithms, we used $M = 1000$ particles. The prior for the target's location and velocity was a Gaussian distribution with mean $\bar{\boldsymbol{x}}_0 = [0\ 0\ 0.01\ 0.01]^\top$ and covariance matrix $\boldsymbol{\Xi} = \text{diag}\{10, 10, 0.1, 0.1\}$, and the initial particles of Ψ were drawn from a uniform distribution on $[10^3, 10^4]$. The cost function of the CRPF was defined by (6.18) and (6.19) with $\lambda = 0$ and $q = 2$.

We experimented with two sensor networks, with deterministically and randomly deployed sensors. In Figure 6.2, we can see the two networks (the sensors are displayed with small circles), a realization of a trajectory of one object and the obtained estimates by APF and CRPF, denoted by APF-bin, and CRPF-Bin, respectively. It can be seen that the algorithms track the target's trajectory closely.

In Figure 6.3, we display the root-mean square errors (RMSEs) of the location estimate of the target obtained by the APF and CRPF algorithms that use complete sensor measurements (APF-Comp and CRPF-Comp, respectively), and the APF and CRPF algorithms based on binary measurements (APF-Bin and CRPF-Bin, respectively) in the deterministic network. The RMSEs in this experiment were

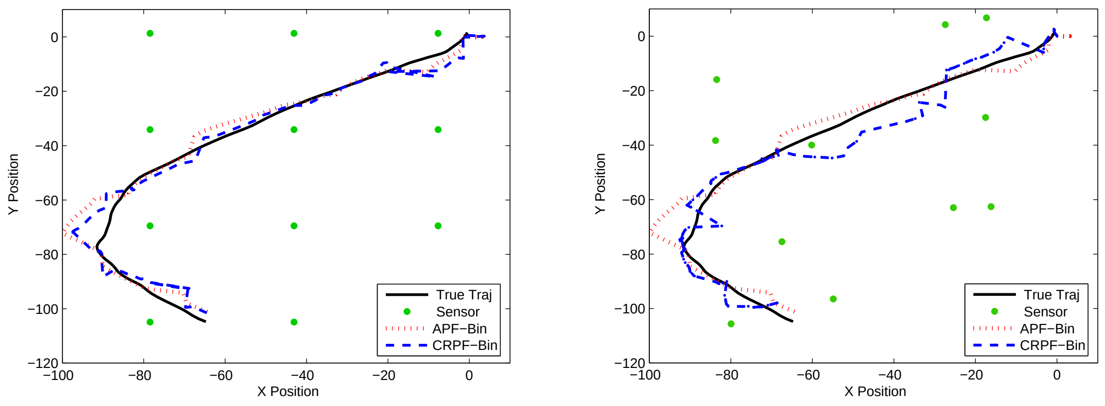


Figure 6.2: A realization of a target trajectory and its estimates by APF and CRPF for deterministically and randomly deployed sensor networks.

obtained by averaging over 100 different realizations. As expected, the performances of APF-Comp and CRPF-Comp were better than the performances of APF-Bin and CRPF-Bin (in RMSE by about 4m) at any time instant t. It should be observed, too, that the CRPF does not have much degraded performance with respect to the APF even though it does not use probabilistic information.

The RMSEs of the locations and velocities of the target and their PCRBs for the random sensor network are shown in Figure 6.4. Again, 100 different realizations were used in the experiment. Similar results were obtained as in the previous experiment. The RMSEs of the constant parameter Ψ are displayed in Figure 6.5 for the deterministically deployed sensor network.

In the next set of experiments, we studied the performance of proposed methods with respect to different SNRs at the FC. In Figure 6.6, we see the performances of these methods expressed by the cumulative distribution functions of the RMSEs of APF-Bin and CRPF-Bin. For example, Figure 6.6 shows that the probability of the RMSE being smaller than 6m is practically one for SNRs greater than or equal to

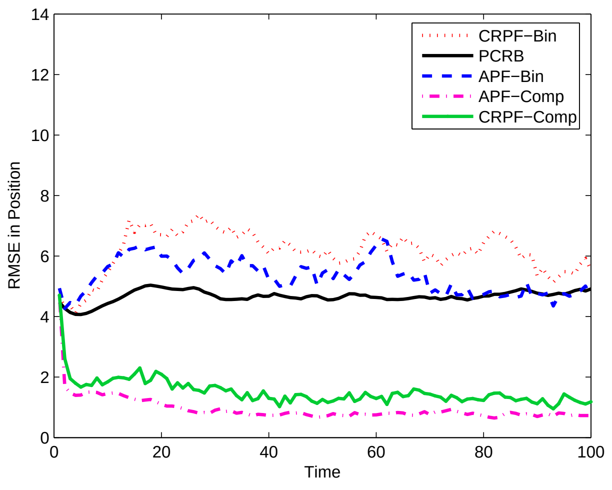


Figure 6.3: RMSEs of the location estimates of the target in deterministic network obtained by the APF and CRPF algorithms with complete sensor measurements (APF-Comp and CRPF-Comp, respectively), and the APF and CRPF algorithms with binary measurements (APF-Bin and CRPF-Bin, respectively).

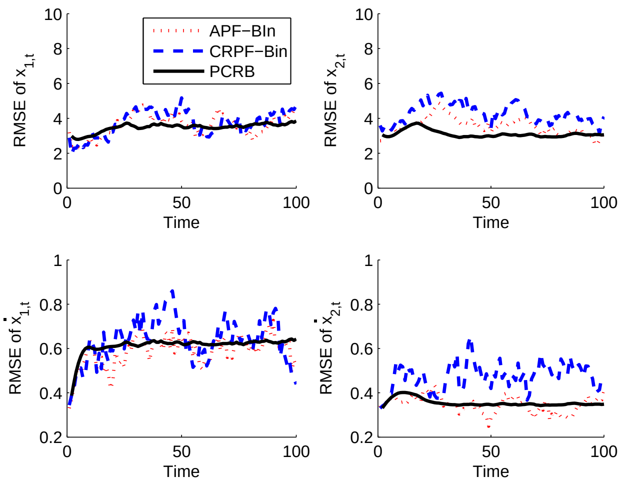


Figure 6.4: RMSEs of the locations and velocities obtained by APF-Bin and CRPF-Bin as functions of time obtained in a random network. The respective PCRBs are also plotted.

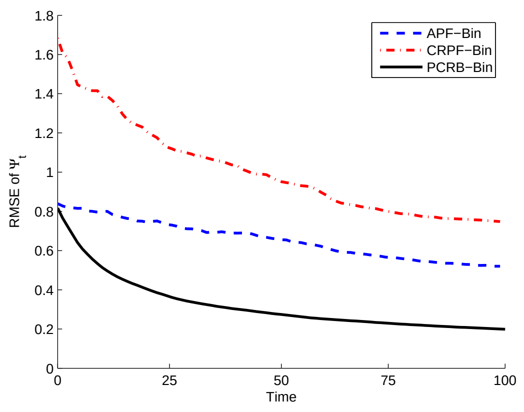


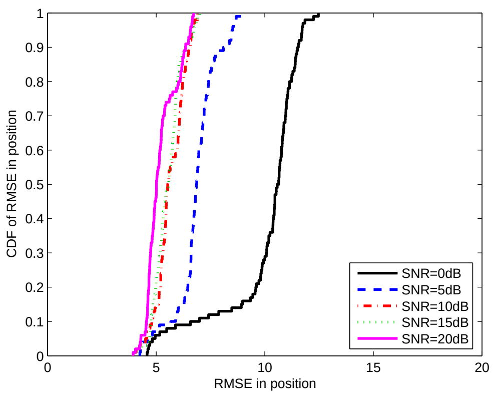
Figure 6.5: RMSEs of Ψ as a function of time.

10dB. At SNR $=$ 5dB, the probability is almost one if the RMSE is less than or equal to 9m. From the graphs, we see that the performance of the CRPF-Bin degrades more rapidly than that of the APF-Bin when the SNR decreases.

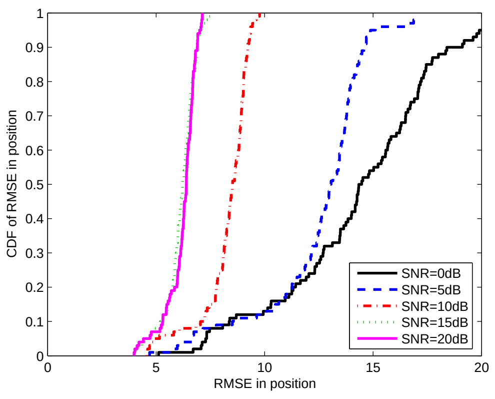
We further studied the impact of the threshold in detecting the presence of the target through the PCRBs for the deterministic network in the examples. The results shown in Figure 6.7 imply that the choice of threshold can be very important for the accuracy of the applied methods.

We have also performed simulations when the APF-Bin uses inaccurate distributions of the noise processes in the model. We generated the noise process in the state equations with a mixture $\boldsymbol{u}_t \sim 0.6\mathcal{N}(\mathbf{0}, \mathbf{C}_{u,1}) + 0.4\mathcal{N}(\mathbf{0}, \mathbf{C}_{u,2})$ with $\mathbf{C}_{u,1} = \text{diag}\{0.05, 0.02\}$ and $\mathbf{C}_{u,2} = \text{diag}\{0.5, 0.2\}$. Instead of the accurate pdf, the APF assumed a Gaussian pdf $\boldsymbol{u}_t \sim \mathcal{N}(\mathbf{0}, \tilde{\mathbf{C}}_u)$ where $\tilde{\mathbf{C}}_u = \text{diag}\{0.01, 0.02\}$. The transmission measurement noise was simulated using a Gaussian mixture as follows

$$\epsilon_{n,t} \sim 0.5\mathcal{N}(\mu_\epsilon, \sigma_\epsilon^2) + 0.5\mathcal{N}(-\mu_\epsilon, \sigma_\epsilon^2)$$



(a) APF.



(b) CRPF.

Figure 6.6: Performance of the APF-Bin and CRPF-Bin algorithms for various SNRs measured by the cumulative distribution functions of the RMSEs.

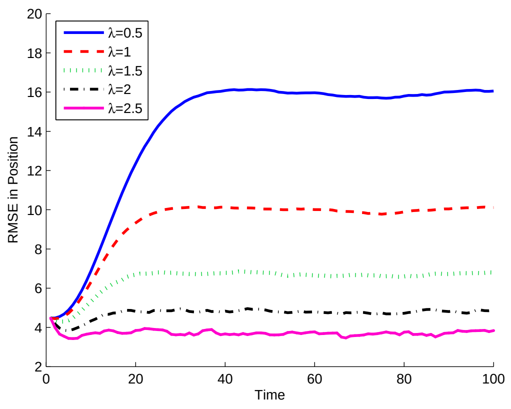


Figure 6.7: PCRBs in determining position for various sensor thresholds.

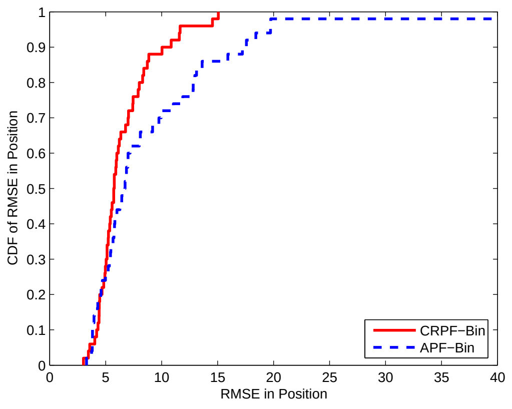


Figure 6.8: Cumulative distribution function of the RMSEs of APF-Bin and CRPF-Bin.

and the APF assumed the measurement noise to be zero mean and with a variance of $\tilde{\sigma}_\epsilon^2 = \mu_\epsilon^2 + \sigma_\epsilon^2$. This ensures that the first two moments of the true and assumed measurement noise statistics are the same. The simulated values of $\tilde{\sigma}_\epsilon^2$ and σ_ϵ^2 were 0.0081 and 0.001, respectively. The RMSEs of 100 different trajectories were computed and summed over the entire time period. In Figure 6.8, we plot the cumulative distribution functions of the RMSE of APF-Bin and CRPF-Bin. For example, the graph shows that 95% of the times, the RMSE accumulated by the CRPF-Bin is below 12m while the RMSE accumulated with the APF-Bin is less than 20m. Clearly CRPF-Bin outperforms the APF-Bin considerably and this is an important result. In practical applications, frequently the assumed distributions and their parameters may be inaccurate, which may cause the degradation of the APF-Bin. As expected in such scenarios CRPF-Bin performs much more robustly.

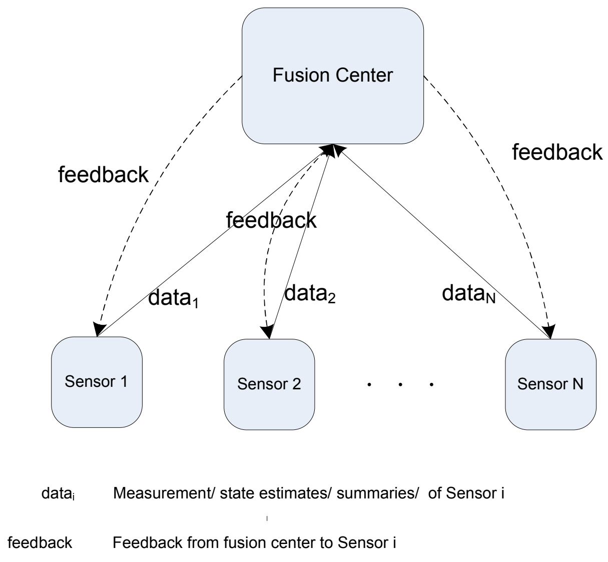
6.8 Summary

We focused on the use of binary wireless sensor networks for tracking a single target. This is a rather challenging problem because complete measurements are compressed to binary decisions of weather a target is or is not detected by a sensor. We applied two particle filtering algorithms for processing of the binary data, auxiliary particle filtering and cost-reference particle filtering. The adopted model of sensor measurements was the signal strength, although any other type of measurement model would be equally applicable. We also derived posterior Cramér-Rao bounds of the estimated unknowns. We conducted several sets of experiments, and the simulation results show that the proposed methods track with good accuracy.

Chapter 7

Target tracking by fusion of random measures

We propose fusion methods for tracking a single target in a sensor network. The sensors use sequential Monte Carlo (SMC) techniques to process the received measurements and obtain random measures of the unknown states. We apply standard particle filtering (SPF) and cost-reference particle filtering (CRPF) methods. Summaries of the random measures of these filters are sent to the fusion center which combines them into a global summary. Similarly, the fusion center may send a global summary to the individual sensors that use it for improved tracking. In Section 7.3 and 7.4 we provide the theory for fusion of measures. Through extensive simulations and comparisons with other methods, we study the performance of the proposed algorithms and show the validity of our approach in Section 7.5. We consider two examples: tracking using bearings only measurements and tracking in a hierarchical sensor network.



Fusion Center
feedback
feedback
feedback
data$_1$
data$_2$
data$_N$
Sensor 1
Sensor 2
Sensor N
data$_i$ Measurement/ state estimates/ summaries/ of Sensor i
feedback Feedback from fusion center to Sensor i

Figure 7.1: Pictorial representation of the considered sensor network framework.

7.1 Introduction

Multisensor data fusion refers to the processing and synergistic combination of data from different sensors to provide improved accuracy and reduced uncertainty about events of interest [89, 90]. Fusion of data from multiple sensors improves the robustness and reliability of the system and has a wide range of application including military, geosciences, robotics, statistical sciences, manufacturing and medicine [90]. Here, we study the problem of target tracking by fusion of information in a sensor network framework as shown in Figure 7.1. There each sensor applies a sequential Monte Carlo (SMC) method to obtain a random measure of the target state. The obtained information is transmitted to the fusion center (FC), which combines the received data and provides an estimate of the target state. Clearly, a better performance could be obtained by transmitting to the FC all the measurements received by the sensors without any processing and running the SMC method at the

FC. However, the transmission of all these measurements is often not practical, and therefore we consider local processing at the sensors. Moreover, we assume that the fusion processing occurs *periodically* or *by request* of the FC. A challenge associated to the proposed scheme comes from the fact that the local SMC methods produce random measures represented by large sets of samples and weights/costs. The transmission of the complete measures is therefore prohibitive. We propose solutions that summarize the random measures and allow for reduced overall communication load.

Fusion of SPF-processed sensor data in the context of target tracking has been previously addressed, whereas fusion of CRPF-processed data has not. In [16], the authors propose two methods for the fusion of SPF-processed data. There the fusion rule for obtaining the joint random measure is obtained as a product of the individual clique's random measures which is not an optimal fusion rule. In our work, we utilize optimal rules for fusion of random measures. In [18], two distributed particle filters for fusion of random measures are presented. The first one, closer in spirit to our work, is based on factorization of likelihood terms and relies on assumptions such as sensor nodes maintaining same particles and random number seeds. We do not utilize any such assumptions in our methods. A decentralized sensor fusion framework with an information theoretic approach for sensors to collect measurements is adopted in [91]. The sensors' belief measure is approximated by a small subset of randomly chosen particles for transmission to neighboring nodes. Also, the formulated fusion expressions require the transmission of "belief" particles of past states. These requirements can be quite prohibitive even when the chosen subset of particles is small. In contrast, our methods have lower communication and power requirements. Finally, compression of the random measures using support vector machine methods is proposed and studied in [92].

The main contributions here are:

1. Distributed algorithms with low communication and power requirements for fusion of SPF and CRPF-processed data are proposed. These methods are generic and can be easily adapted to various sensor network architectures. Previously proposed fusion methods either have large communication requirements or are applied to specific architectures.

2. We show the feasibility of our methods for target tracking with and without feedback from the FC. To the best of our knowledge previously proposed methods for fusion of random measures do not consider any feedback.

3. We provide simulation studies of target tracking in flat and hierarchical sensor networks with fusion of random measures.

7.2 Problem statement

The target state is obtained using a standard model of constant velocity [93] according to

$$\boldsymbol{x}_t = \mathbf{F}_x \boldsymbol{x}_{t-1} + \boldsymbol{\Gamma}_u \boldsymbol{u}_t \tag{7.1}$$

where $\boldsymbol{x}_t = [x_{1,t}, x_{2,t},\ \dot{x}_{1,t},\ \dot{x}_{2,t}]^\top \in \mathbb{R}^4$ comprises the target position and velocity in the two-dimensional space, and $\boldsymbol{u}_t$ is the random noise vector. Denoting by $\mathbf{I}_2$ and

$\mathbf{0}_2$ the 2×2 identity and zero matrices, the transition matrices are given by

$$\mathbf{F}_{\mathbf{x}} = \begin{pmatrix} \mathbf{I}_2 & T_s\mathbf{I}_2 \\ \mathbf{0}_2 & \mathbf{I}_2 \end{pmatrix} \quad \text{and} \quad \mathbf{\Gamma}_u = \begin{pmatrix} \frac{T_s^2}{2}\mathbf{I}_2 \\ T_s\mathbf{I}_2 \end{pmatrix}$$

with T_s being the sampling period. The sensor measurement signals $\boldsymbol{y}_t^n$ are typically non-linear functions of the target dynamics. We denote by $\boldsymbol{x}_{0:t}$ the target dynamics from time instant 0 to time instant t, and by $\mathbf{y}_{1:t}^n$ the data observed by the n^{th} sensor up to time instant t.

Recall that, when the distribution of the noise processes are known, each sensor using a SPF method approximates $p(\boldsymbol{x}_{0:t} \mid \boldsymbol{y}_{1:t}^n)$ by a random measure $\chi_t^n = \{\boldsymbol{x}_{0:t}^{(m),n}, w_t^{(m),n}\}_{m=1}^M$, where $\boldsymbol{x}_{0:t}^{(m),n}$ are particles of the random measure, $w_t^{(m),n}$ are the weights associated to the particles, and M denotes the number of particles. When the distribution of the noise processes are unknown, each sensor using CRPF methods obtains a random measure of the state $\boldsymbol{x}_{0:t}$, which is represented by $\zeta_t^n = \{\boldsymbol{x}_{0:t}^{(m),n}, c_t^{(m),n}\}_{m=1}^M$, where $c_t^{(m),n}$ denotes the costs assigned to the particles. The objective of the proposed fusion methods is to obtain the joint random measures χ_t or ζ_t from the individual sensor random measures χ_t^n or ζ_t^n, $n = 1, \cdots, N$, and from them obtain estimates of the unknown state $\boldsymbol{x}_t$.

7.3 Fusion with and without feedback

In this section we describe the theoretical expressions for fusion of probability distributions in two scenarios.[1] In the first, the global probability distribution (GPD)

[1]Note that this is a theoretical study. It is not possible in practice for the sensors to transmit to the FC the required information. The theory explained in this Section will be applied to obtain

obtained by fusion of the individual probability distributions (IPDs) from the sensors is not reported back to the sensors, and in the second, the FC broadcasts the GPD back to the sensors, which is then used for improved tracking. These scenarios are depicted in Figure 7.1. The strategy with feedback is particularly advantageous in situations when some of the filters start diverging.

7.3.1 Fusion without feedback

For the sake of simplicity let us consider fusion of densities from two sensors. The obtained result can readily be generalized to an arbitrary number of sensors. The GPD, $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}^1_{1:t}, \boldsymbol{y}^2_{1:t})$, can be written as

$$\begin{aligned} p(\boldsymbol{x}_{0:t}|\boldsymbol{y}^1_{1:t}, \boldsymbol{y}^2_{1:t}) &= p(\boldsymbol{x}_{0:t}|\boldsymbol{y}^1_t, \boldsymbol{y}^2_t, \boldsymbol{y}^1_{1:t-1}, \boldsymbol{y}^2_{1:t-1}) \\ &\propto p(\boldsymbol{y}^1_t, \boldsymbol{y}^2_t|\boldsymbol{x}_t, \boldsymbol{x}_{0:t-1}, \boldsymbol{y}^1_{1:t-1}, \boldsymbol{y}^2_{1:t-1})p(\boldsymbol{x}_t|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}^1_{1:t-1}, y^2_{1:t-1}) \\ &\quad \times p(\boldsymbol{x}_{0:t-1}|\boldsymbol{y}^1_{1:t-1}, \boldsymbol{y}^2_{1:t-1}). \end{aligned} \tag{7.2}$$

Assuming independence among the sensor measurements $\{\boldsymbol{y}^1_t, \boldsymbol{y}^2_t\}$ conditioned on $\boldsymbol{x}_t$, we have

$$p(\boldsymbol{y}^1_t, \boldsymbol{y}^2_t|\boldsymbol{x}_t, \boldsymbol{x}_{0:t-1}, \boldsymbol{y}^1_{1:t-1}, \boldsymbol{y}^2_{1:t-1}) = p(\boldsymbol{y}^1_t|\boldsymbol{x}_t, \boldsymbol{x}_{0:t-1}, \boldsymbol{y}^1_{1:t-1})p(\boldsymbol{y}^2_t|\boldsymbol{x}_t, \boldsymbol{x}_{0:t-1}, \boldsymbol{y}^2_{1:t-1}).$$

We know that

$$p(\boldsymbol{x}_t|\boldsymbol{y}^n_{1:t}) = g^{-1}(y_t)p(\boldsymbol{x}_t|\boldsymbol{y}^n_{1:t-1})p(\boldsymbol{y}^n_t|\boldsymbol{x}_t, \boldsymbol{x}_{0:t-1}, \boldsymbol{y}^n_{1:t-1})$$

practical methods later.

where $g(y_t)$ represents the normalization terms not involving $\boldsymbol{x}_t$. Therefore

$$p(\boldsymbol{y}_t^n|\boldsymbol{x}_t, \boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}^n) \quad = \quad g(y_t)\frac{p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}^n)}{p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}^n)}. \tag{7.3}$$

From (7.2) and (7.3) we get

$$\begin{aligned} p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t}^1, \boldsymbol{y}_{1:t}^2) \quad \propto \quad & \frac{p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}^1)}{p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}^1)} \, \frac{p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}^2)}{p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}^2)} \, p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \\ & \times p(\boldsymbol{x}_{0:t-1}|\boldsymbol{y}_{1:t-1}^1, \boldsymbol{y}_{1:t-1}^2). \end{aligned} \tag{7.4}$$

Note that the distributions $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ and $p(\boldsymbol{x}_{0:t-1}|\boldsymbol{y}_{1:t-1}^1, \boldsymbol{y}_{1:t-1}^2)$ are known to the FC. The former is obtained from the state equation and the latter is the GPD at time instant $t-1$. The two sensors transmit to the FC information about $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}^1)$ and $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}^2)$, whereas $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}^n)$, $n = 1, 2$, can in principle be obtained from $p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1}^n)$ and $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$, which are known at the FC.

Generalizing equation (7.4) for N sensors, we have

$$p(\boldsymbol{x}_{0:t} \mid \mathbf{y}_{1:t}^{1:N}) \propto p(\boldsymbol{x}_{0:t-1} \mid \mathbf{y}_{1:t-1}^{1:N}) \, p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) \prod_{n=1}^{N} \frac{p(\boldsymbol{x}_t \mid \mathbf{y}_{1:t}^n)}{p(\boldsymbol{x}_t \mid \mathbf{y}_{1:t-1}^n)} \tag{7.5}$$

which is the optimal recursive fusion equation.

7.3.2 Fusion with feedback

As before, we first consider fusion of posterior distributions from two sensors but with combined posterior feedback from the FC. At time instant t, the FC feeds back to the sensors the GPD $p(\boldsymbol{x}_{0:t-1}|\boldsymbol{y}_{1:t-1}^1, \boldsymbol{y}_{1:t-1}^2)$. Then the posterior of the first sensor is formed according to

$$p(\boldsymbol{x}_{0:t}|\boldsymbol{y}^1_{1:t}, \boldsymbol{y}^2_{1:t-1}) \propto p(\boldsymbol{y}^1_t|\boldsymbol{x}_t)p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})p(\boldsymbol{x}_{0:t-1}|\boldsymbol{y}^1_{1:t-1}, \boldsymbol{y}^2_{1:t-1}). \tag{7.6}$$

The posterior of the second sensor is obtained analogously.

The FC receives the posteriors $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}^1_{1:t}, \boldsymbol{y}^2_{1:t-1})$ and $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}^1_{1:t-1}, \boldsymbol{y}^2_{1:t})$ and fuses them by

$$\begin{aligned} p(\boldsymbol{x}_{0:t}|\boldsymbol{y}^1_{1:t}, \boldsymbol{y}^2_{1:t}) \propto\ & \frac{p(\boldsymbol{x}_t|\boldsymbol{y}^1_{1:t}, \boldsymbol{y}^2_{1:t-1})}{p(\boldsymbol{x}_t|\boldsymbol{y}^1_{1:t-1}, \boldsymbol{y}^2_{1:t-1})} \frac{p(\boldsymbol{x}_t|\boldsymbol{y}^1_{1:t-1}, \boldsymbol{y}^2_{1:t})}{p(\boldsymbol{x}_t|\boldsymbol{y}^1_{1:t-1}, \boldsymbol{y}^2_{1:t-1})} p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \\ & \times p(\boldsymbol{x}_{0:t-1}|\boldsymbol{y}^1_{1:t-1}, \boldsymbol{y}^2_{1:t-1}). \end{aligned} \tag{7.7}$$

This expression was derived along the same lines of reasoning as (7.4).

When we generalize (7.7) for N sensors, the fusion rule becomes

$$p(\boldsymbol{x}_{0:t}|\boldsymbol{y}^{1:N}_{1:t}) \propto p(\boldsymbol{x}_{0:t-1}|\boldsymbol{y}^{1:N}_{1:t-1})p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \prod_{n=1}^{N} \frac{p(\boldsymbol{x}_t|\boldsymbol{y}^1_{1:t-1}, \cdots \boldsymbol{y}^n_{1:t} \cdots \boldsymbol{y}^N_{1:t-1})}{p(\boldsymbol{x}_t|\boldsymbol{y}^1_{1:t-1}, \cdots \boldsymbol{y}^n_{1:t-1} \cdots \boldsymbol{y}^N_{1:t-1})}. \tag{7.8}$$

7.4 Fusion by using random measures

The derived fusion rules from the previous section are of little practical value. In our case where we use random measures to represent our knowledge about the evolving state, the situation is even worse. The transmission of the random measures would require sending a large number of particle values and weights/costs. This altogether would be much more demanding in communication resources than the transmission of the actual measurements to the FC and would beat the whole purpose of using SMC at the sensors. We propose schemes that alleviate the sending of complete measures and by sending summaries of the random measures. First we describe strategies for summarizing the random measures constructed by SPF and then for those of CRPF.

7.4.1 Fusion of summaries of SPF random measures

In the SPF framework the random measures approximate distributions. If these distributions are unimodal, we propose that their random measures are summarized by Gaussians.[2] This type of approximation has already been used in the framework of Gaussian particle filtering [32]. There the individual posterior and predictive distributions are approximated by Gaussian distributions, i.e.,

$$\begin{aligned} p(\boldsymbol{x}_t \mid \mathbf{y}^n_{1:t-1}) &\simeq \mathcal{N}(\tilde{\boldsymbol{\mu}}^n_t, \tilde{\boldsymbol{\Sigma}}^n_t) \\ p(\boldsymbol{x}_t \mid \mathbf{y}^n_{1:t}) &\simeq \mathcal{N}(\hat{\boldsymbol{\mu}}^n_t, \hat{\boldsymbol{\Sigma}}^n_t) \\ p(\boldsymbol{x}_t \mid \mathbf{y}^{1:N}_{1:t}) &\simeq \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t). \end{aligned} \tag{7.9}$$

With these approximations and from (7.5) we have

$$\begin{aligned} p(\boldsymbol{x}_t \mid \mathbf{y}^{1:N}_{1:t}) &\propto \prod_{n=1}^{N} \frac{\mathcal{N}(\hat{\boldsymbol{\mu}}^n_t, \hat{\boldsymbol{\Sigma}}^n_t)}{\mathcal{N}(\tilde{\boldsymbol{\mu}}^n_t, \tilde{\boldsymbol{\Sigma}}^n_t)} \, p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) \, p(\boldsymbol{x}_{0:t-1} \mid \mathbf{y}^{1:N}_{1:t-1}) \\ &\propto \frac{\mathcal{N}(\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t)}{\mathcal{N}(\tilde{\boldsymbol{\mu}}_t, \tilde{\boldsymbol{\Sigma}}_t)} \, p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) \, p(\boldsymbol{x}_{0:t-1} \mid \mathbf{y}^{1:N}_{1:t-1}) \end{aligned}$$

[2]In Appendix A, we provide conditions for the validity of the Gaussian assumptions.

where

$$
\begin{aligned}
\hat{\boldsymbol{\Sigma}}_t^{-1} &= \hat{\boldsymbol{\Sigma}}_t^{1^{-1}} + \hat{\boldsymbol{\Sigma}}_t^{2^{-1}} + \cdots + \hat{\boldsymbol{\Sigma}}_t^{N^{-1}} \\
\hat{\boldsymbol{\mu}}_t &= \hat{\boldsymbol{\Sigma}}_t \left(\hat{\boldsymbol{\Sigma}}_t^{1^{-1}} \hat{\boldsymbol{\mu}}_t^1 + \hat{\boldsymbol{\Sigma}}_t^{2^{-1}} \hat{\boldsymbol{\mu}}_t^2 + \cdots + \hat{\boldsymbol{\Sigma}}_t^{N^{-1}} \hat{\boldsymbol{\mu}}_t^N \right) \\
\tilde{\boldsymbol{\Sigma}}_t^{-1} &= \tilde{\boldsymbol{\Sigma}}_t^{1^{-1}} + \tilde{\boldsymbol{\Sigma}}_t^{2^{-1}} + \cdots + \tilde{\boldsymbol{\Sigma}}_t^{N^{-1}} \\
\tilde{\boldsymbol{\mu}}_t &= \tilde{\boldsymbol{\Sigma}}_t \left(\tilde{\boldsymbol{\Sigma}}_t^{1^{-1}} \tilde{\boldsymbol{\mu}}_t^1 + \tilde{\boldsymbol{\Sigma}}_t^{2^{-1}} \tilde{\boldsymbol{\mu}}_t^2 + \cdots + \tilde{\boldsymbol{\Sigma}}_t^{N^{-1}} \tilde{\boldsymbol{\mu}}_t^N \right).
\end{aligned}
\tag{7.10}
$$

The parameters of the Gaussians can readily be computed from the SPF random measure. For example, if the approximation of $p(\boldsymbol{x}_t \mid \mathbf{y}_{1:t}^n)$ is given by $\sum_{m=1}^{M} w_t^{(m),n} \delta(\boldsymbol{x}_t - \boldsymbol{x}_t^{(m),n}) \approx \mathcal{N}(\boldsymbol{\mu}_t^n, \boldsymbol{\Sigma}_t^n)$, then

$$
\begin{aligned}
\boldsymbol{\mu}_t^n &= \sum_{m=1}^{M} w_t^{(m),n} \boldsymbol{x}_t^{(m),n} \\
\boldsymbol{\Sigma}_t^n &= \sum_{m=1}^{M} w_t^{(m),n} (\boldsymbol{x}_t^{(m),n} - \boldsymbol{\mu}_t^n)(\boldsymbol{x}_t^{(m),n} - \boldsymbol{\mu}_t^n)^\top.
\end{aligned}
\tag{7.11}
$$

We define transmission length ($\mathcal{T}_L$), as the number of real numbers transmitted to the FC by each sensor. Therefore with this fusion scheme we have $\mathcal{T}_L = \frac{d(d+3)}{2}$ where d is the dimension of the state.

When the posterior densities have more than one mode, their representation by a single Gaussian density can be inaccurate. In such situations, one possibility is to summarize the random measure by mixture-Gaussians, i.e.,

$$
\begin{aligned}
p(\boldsymbol{x}_t \mid \mathbf{y}_{1:t-1}^n) &\simeq \sum_{l=1}^{L} \tilde{\pi}_{t,l} \mathcal{N}(\tilde{\boldsymbol{\mu}}_{t,l}^n, \tilde{\boldsymbol{\Sigma}}_{t,l}^n) \\
p(\boldsymbol{x}_t \mid \mathbf{y}_{1:t}^n) &\simeq \sum_{l=1}^{L} \hat{\pi}_{t,l} \mathcal{N}(\hat{\boldsymbol{\mu}}_{t,l}^n, \hat{\boldsymbol{\Sigma}}_{t,l}^n),
\end{aligned}
\tag{7.12}
$$

where L is the number of mixands.

From (7.5) and (7.12) we have

$$p(\boldsymbol{x}_t \mid \mathbf{y}_{1:t}^{1:N}) \propto \prod_{n=1}^{N} \frac{\sum_{l=1}^{L} \tilde{\pi}_{t,l} \mathcal{N}(\tilde{\boldsymbol{\mu}}_{t,l}^{n}, \tilde{\boldsymbol{\Sigma}}_{t,l}^{n})}{\sum_{l=1}^{L} \hat{\pi}_{t,l} \mathcal{N}(\hat{\boldsymbol{\mu}}_{t,l}^{n}, \hat{\boldsymbol{\Sigma}}_{t,l}^{n})} p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})\, p(\boldsymbol{x}_{0:t-1} \mid \mathbf{y}_{1:t-1}^{1:N}). \tag{7.13}$$

Thus the FC obtains the GPD through (7.13). The parameters of the mixture Gaussians can be obtained by the expectation-maximization (EM) algorithm which is an iterative two-step method [94]. Here, the posterior distribution is approximated by a Gaussian mixture, therefore as above, the transmission length, $\mathcal{T}_L = L\frac{d(d+3)}{2} + L - 1$.

7.4.2 Fusion of summaries of CRPF random measures

Recall that in the CRPF framework, each sensor maintains a random measure with costs associated to the particles. We propose that the random measure of the n^{th} sensor, ζ_t^n, is summarized by one of the following approaches:

1. CRPF-Pdf: summary-based on a probability distribution

2. CRPF-Par: summary-based on the best particle.

CRPF-Pdf: Summary based on a probability distribution function

In this method, the sensors convert the costs of the particles into probability masses. Once the conversion is accomplished, the summarization is carried out in the same way as described in the previous section. For e.g., one way of converting the costs is to use

$$\pi_{c,t}^{(m),n} \propto \frac{1}{(c_t^{(m),n} - \min(c_t^{(m),n}) + \frac{1}{M})^2}. \tag{7.14}$$

When there is a feedback, the FC sends back to the sensors the Gaussian constructed from the individual Gaussians. The sensors use the received Gaussian to generate particles and assign to each of them zero costs. At each sensor, the CRPF is then implemented in the usual way until the sensors receive the next feedback from the FC. Here, too, as in the SPF method when the posterior is approximated by a single Gaussian, we have $\mathcal{T}_L = \frac{d(d+3)}{2}$.

CRPF-Par: Summary based on the best particle

This method is the simplest of all. Here each sensor transmits to the FC the particle that has the minimal cost and the FC computes the mean of all these samples. If the sensors also transmit their minimum costs, the corresponding "best" estimate can be a weighted estimate. When the sensors have good estimates and the regions of uncertainty are small, these methods of fusing single sensor estimates are efficient. However when the sensors have large regions of uncertainty, the resulting fused estimate may not be accurate. With this method, we have a transmission length of $\mathcal{T}_L = d + 1$.

Another possibility of using the best particles and their costs is to convert the costs to a pmf and from it construct a Gaussian that can be sent back to the sensors as a feedback.

7.5 Simulations and results

7.5.1 Bearings only target tracking

In many radar and sonar applications, target tracking is performed using only bearing measurements. Here the sensors operate in a passive mode and measure the direction of arrival of the signal emitted by the target. We consider the sensor network shown in Figure 7.2 where the three sensors are denoted by small circles and are placed at positions $(-50, -10)$, $(75, 10)$ and $(80, -50)$. The bearing measurements of the target relative to the n^{th} sensor location are mathematically modeled as

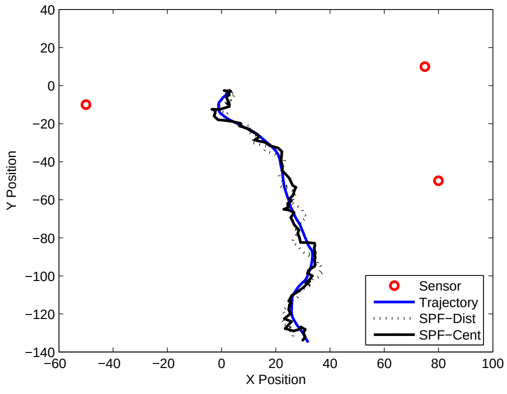
$$y_t^n = \arctan\left(\frac{x_{2,t} - l_y^n}{x_{1,t} - l_x^n}\right) + v_t^n \tag{7.15}$$

where $\{l_x^n, l_y^n\}$ are the coordinates of the sensor [26], and v_t^n is zero mean white Gaussian noise with variance σ_v^2.

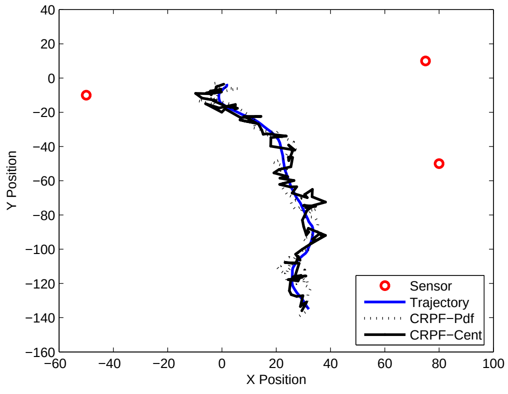
The initial state $\boldsymbol{x}_0$ of the target was drawn from $\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ with $\boldsymbol{\mu}_0 = [0,\ 0,\ 0.1,\ 0.0]^\top$ and $\boldsymbol{\Sigma}_0 = diag\,(10,\ 10,\ 0.1,\ 0.1)$.[3] The target's dynamics were modeled using (7.1) with the covariance matrix of the zero mean Gaussian state process noise $\mathbf{C}_u = diag(0.05,\ 0.02)$. The target's trajectory was simulated for $N_t = 100$ time instants with a sampling period $T_s = 1$ s. The standard deviation of the measurement noise σ_v was set to 0.05. In computing the RMSEs of the unknowns, we used 100 trajectory runs.

In Figure 7.2(a) we show a target trajectory and its estimates using the proposed SPF method (labeled as *SPF-Dist*, meaning distributed). For comparison purposes we also included the performance of the system where the sensors sent

[3]The symbol $diag(\boldsymbol{x})$ represents a diagonal matrix formed with vector $\boldsymbol{x}$ as its diagonal elements.



(a) SPF



(b) CRPF

Figure 7.2: The multisensor network and estimates of the target trajectory.

their measurements to the FC. The FC used the measurements to run a SPF-based algorithm (labeled as *SPF-Cent*, meaning centralized). In Figure 7.2(b), we show analogous results for the CRPF-Pdf and the centralized CRPF (labeled as *CRPF-Cent*) methods operating on the full measurements. We chose the risk and cost functions outlined in equations (4.1) and (4.4) with $q = 2$. This method is thus analogous to a least squares fitting approach where decision on the particles is made using the residuals.

In Figs. 7.3 and 7.4, we plot the RMSEŠs obtained by the sensors and the FC using SPF methods with and without feedback. Each of the sensors maintained a random measure with $M = 1000$ samples. The SPF methods utilized the prior for propagation of the particles. In both scenarios, the sensors transmitted to the FC the summaries of their random measures at every time instant. In the feedback scenario, the FC also transmitted to the sensors the global summary which was used to re-initialize their filters. It can be observed that with feedback, the RMSEs of each of the individual sensors is smaller than in situations without any feedback. The overall

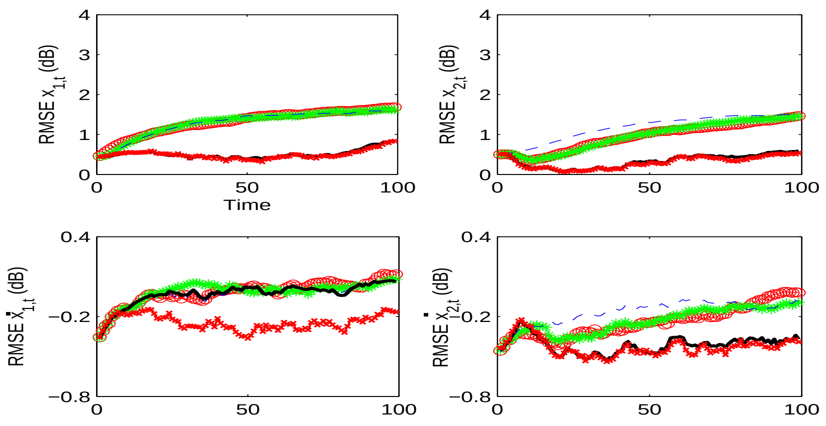


Figure 7.3: RMSEs (dB) of the target dynamics by the sensors and FC using SPF *without* feedback. -o- Sensor 1, -*- Sensor 2, — Sensor 3, – FC SPF-Dist, -x FC SPF-Cent

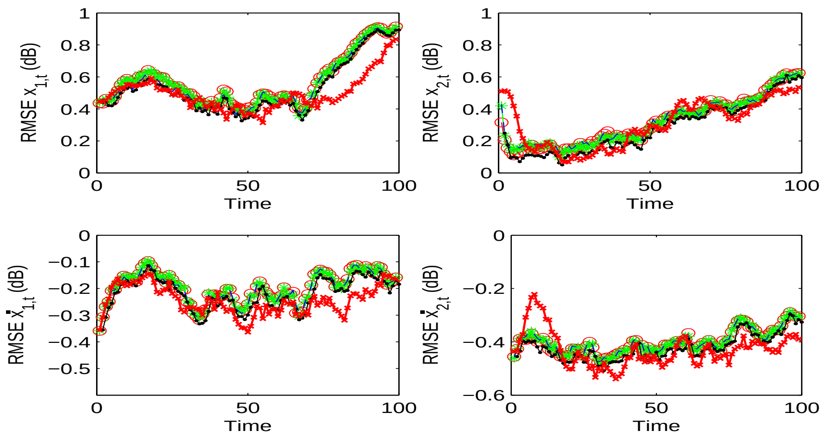
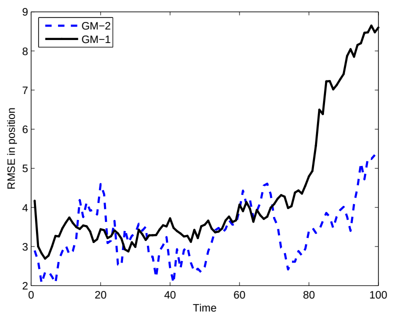


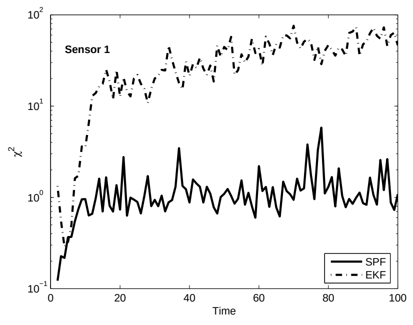
Figure 7.4: RMSEs of the target dynamics by the sensors and FC using SPF *with* feedback. -o- Sensor 1, -*- Sensor 2, — Sensor 3, – FC SPF-Dist, -x FC SPF-Cent

improvement in the RMSEs of the FC estimates, however, is small.

In Figure 7.5(a), we plot the RMSEs of the position when the SPF algorithms approximated the random measure with a single Gaussian density (GM-1) and with



(a) GM-GMM



(b) χ^2 Measure

Figure 7.5: (a) Comparison of RMSEs with a single Gaussian approximation and with a Gaussian mixture approximation. (b) χ^2-Divergence for SPF and EKF based posterior approximations at sensor 1.

a Gaussian mixture with two mixands (GM-2). We use the χ^2 divergence[4] measure to compare the approximation of the posterior by a Gaussian distribution using SMC and EKF methods. In Figure 7.5(b), we plot the χ^2 divergence measure due to approximation of the sampled density with a Gaussian distribution at sensor 1. Clearly the performance of the SPF-based approximation is better than the Gaussian approximation with the EKF by an order of magnitude. The plots are similar for the approximations made at other sensors. In Figure 7.6, we plot the mean error

Table 7.1: Bias(m)

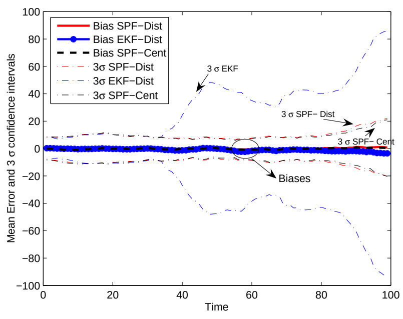
Time(s)	**Sen 1**	**Sen 2**	**Sen 3**	**FC**
20	-2.83	-0.46	-0.23	0.19
40	-0.77	-2.18	-2.92	-0.47
60	1.61	-5.63	-5.04	-0.49
80	2.70	-8.23	-4.22	-0.07
100	5.94	-7.16	-0.12	0.81

Table 7.2: Standard Deviation(m)

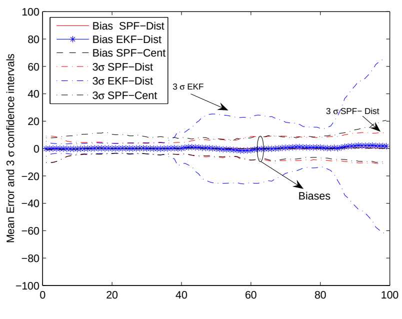
Time(s)	**Sen 1**	**Sen 2**	**Sen 3**	**FC**
20	10.96	11.40	10.61	3.62
40	19.83	24.17	22.53	2.68
60	32.76	31.03	30.99	2.60
80	40.82	43.12	35.80	2.99
100	50.86	50.61	43.28	7.03

Bias and standard deviation of location estimates of the targets at different times.

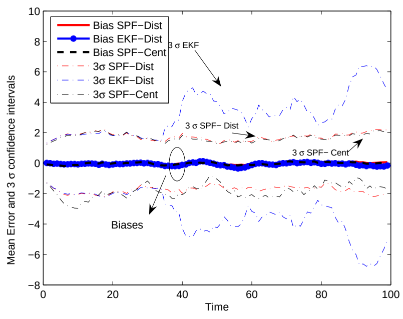
[4]The definition and computation of this measure are described in Appendix B.



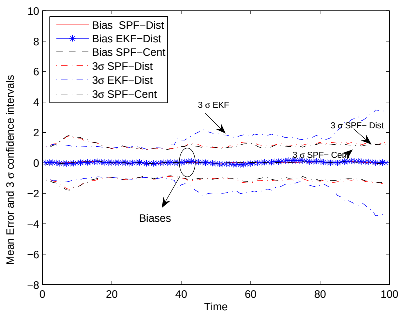
(a) $x_{1,t}$



(b) $x_{2,t}$

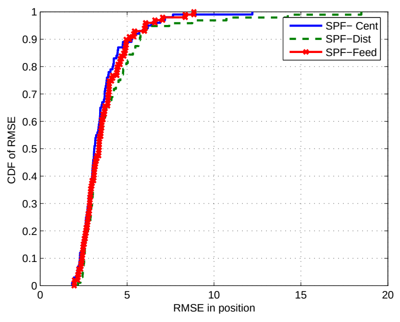


(c) $x_{3,t}$

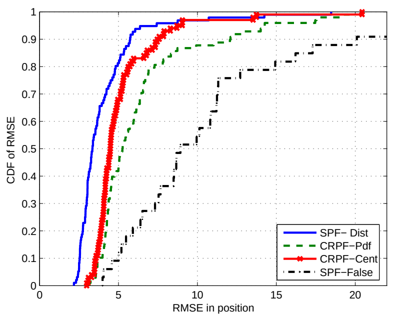


(d) $x_{4,t}$

Figure 7.6: Spread of errors in estimating the state at the FC.



(a) SPF



(b) CRPF

Figure 7.7: Comparison of RMSEs with centralized and proposed SPF and CRPF methods.

and the 3σ confidence interval depicting the spread in error in estimating the target dynamics at the FC. We summarize the results in Tables 7.1 and 7.2. Clearly we can see through these plots that the spreads in errors are similar for the SPF-based summary approach and the SPF method with complete measurements. However the EKF exhibits divergence as can be seen from the high spread of errors. The results in the table show that the errors increase with time, which is due to the target leaving the sensor field. We also use the cumulative distribution function (cdf) of the total RMSE accumulated over time as our metric for analyzing the performance of the proposed and the centralized algorithms. It can be seen from Figure 7.7(a) that there is a small loss of performance with the proposed method over the centralized methods. We also plot in Figure 7.7(b) the cdf of the RMSEs obtained using the CRPF methods. The performance of the CRPF-Pdf method is comparable with that of the SPF-Dist method which makes full assumptions of the noise processes. Moreover, the performance of the proposed CRPF-Dist method is very similar to the CRPF-Cent. We also simulated the case when the SPF methods make wrong assumptions about the noise process. In Figure 7.7(b), the filter associated with the SPF-False method assumed $\mathbf{C}_u = diag(0.005,\ 0.002)$ and $\sigma_v = 0.01$. With a single Gaussian approximation we have, the probability P that the RMSE in position is less than or equal to $10m$, i.e., $P(\text{RMSE}_{\text{pos}} \leq 10m) = 0.96$ while with a Gaussian mixture ($L = 2$), $P(\text{RMSE}_{\text{pos}} \leq 10m)) \approx 1.0$; and with CRPF-Pdf, $P(\text{RMSE}_{\text{pos}} \leq 10m) = 0.86$. However when the probability distributions of the noise processes are unknown, for the SPF with incorrect noise statistics, $P(\text{RMSE}_{\text{pos}} \leq 10m) = 0.57$. Clearly the performance of the SPF method under wrong noise assumptions is poor, motivating the use of other methods which make less assumptions of the noise processes when their distributions are unknown.

7.5.2 Target tracking in a hierarchical sensor network

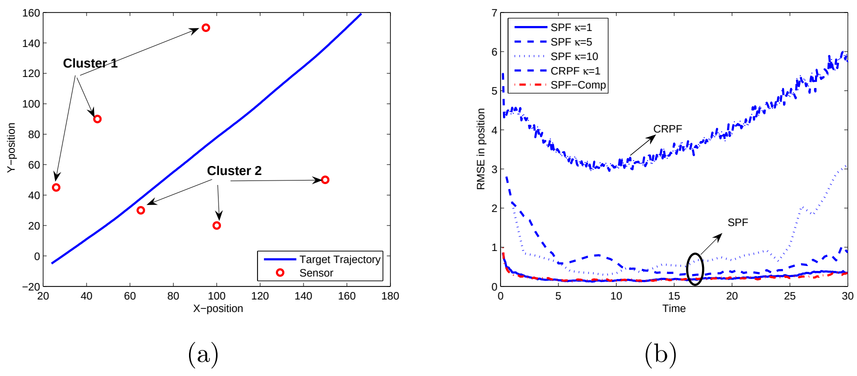
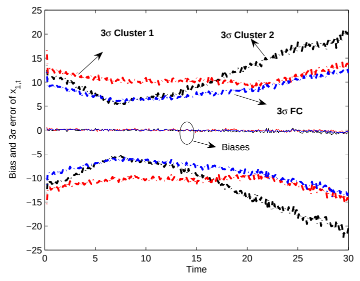
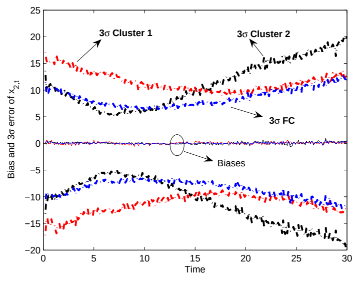


Figure 7.8: (a) A hierarchical sensor network. (b) RMSEs of the target position by using SPF and CRPF for various κ

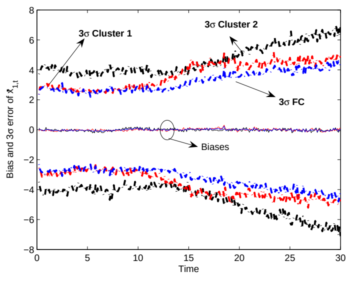
An important application of data fusion is target tracking in hierarchical sensor networks (HSN), where sensors form clusters and transmit their measurements to a specialized node known as a leader node (LN). This LN is a specialized node



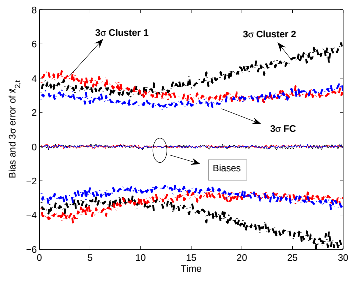
(a) Bias and 3σ of $x_{1,t}$



(b) Bias and 3σ of $x_{2,t}$



S



(c) Bias and 3σ of $\dot{x}_{1,t}$

(d) Bias and 3σ of $\dot{x}_{2,t}$

Figure 7.9: Spread of errors in estimating the state at FC.

which has greater computational and communication capabilities than the sensors. Upon obtaining the measurements from the sensors, the LNs estimate the posterior density of the target's dynamics. The summaries of the random measures are then transmitted to the FC which combines them to obtain a joint summary. In this experiment, we implemented our proposed ideas for target tracking in the HSN shown in Figure 7.8(a). There are six sensors placed at $(26, 45)$, $(45, 90)$, $(65, 30)$, $(95, 150)$, $(100, 20)$, $(150, 50)$ which form two clusters. The sensors collected three different measurements, the bearing, the power of the signal emitted by the target and the relative velocity of the target [81]. These three signal modalities can be mathematically written as

$$\begin{aligned} y_{1,t}^n &= \arctan\left(\frac{x_{2,t} - l_y^n}{x_{1,t} - l_x^n}\right) + v_{1,t}^n && (7.16) \\ y_{2,t}^n &= \Psi - 5\log_{10}\left((x_{1,t} - l_x^n)^2 + (x_{2,t} - l_y^n)^2\right) + v_{2,t}^n && (7.17) \\ y_{3,t}^n &= \nu_{r,t}^n + v_{3,t}^n && (7.18) \end{aligned}$$

where $\Psi = -50$dB is the signal strength within a known reference distance, $\nu_{r,t}^n$ is relative velocity of the target relative to the nth sensor, and $v_{1,t}^n \sim N(\mu_{v_1}, \sigma_{v_1}^2)$, $v_{2,t}^n \sim N(\mu_{v_2}, \sigma_{v_2}^2)$, $v_{3,t}^n \sim N(\mu_{v_3}, \sigma_{v_3}^2)$, are zero mean Gaussian measurement noise processes. We assumed the variance of the noise to be the same across all the sensors. In our simulations we used $\sigma_{v_1} = 0.01$, $\sigma_{v_2} = 1$, and $\sigma_{v_3} = 1$. The initial target dynamics were drawn from $\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ with $\boldsymbol{\mu}_0 = [0,\ 0,\ 5,\ 5]^\top$ and $\boldsymbol{\Sigma}_0 = diag\,(10,\ 10,\ 0.01,\ 0.01)$. The target trajectories were obtained as earlier, with the noise covariance matrix and $\mathbf{C}_u = diag(0.1,\ 0.2)$. The sampling period was $T_s = 0.1$ s and the length of each trajectory was $N_t = 300$, i.e., the target was observed for 30s.

Here we also study the scenarios when the LNs transmit the summaries of their random measures to the FC for every $\kappa = 1, 5$, and 10 time instants. The RMSEs were computed over 50 different trajectory runs. In Figure 7.8(b), the RMSEs of the proposed SPF method at the FC are shown. Here, the LNs' approximate their random measure using single Gaussians. There is a very small loss in performance with the LNs transmitting the summaries to the FC every $\kappa = 10$ time instants, thus considerably saving power and communication resources. In Figure 7.9, we display the bias and the spread of the errors with the CRPF-par method. Clearly, it can be seen that the estimate by the FC is for most of the time better than the estimates of the LNs.

7.6 Summary

SPF and CRPF fusion algorithms for target tracking have been presented for various scenarios, including those of known and unknown noise probability distributions. The sensors implement these SMC methods and periodically transmit summaries of their results instead of the complete random measures. Therefore, low communication and power requirements are needed. In SPF methods, the summaries are represented by mean and covariance matrices. When the posterior distribution is unimodal, these random measures are adequately approximated with a single Gaussian which has lower transmission requirements. With CRPF methods, the summaries are represented either with a parametric distribution or by using the best particles of the random measure.

We have demonstrated the performance of the proposed algorithms through two target tracking examples in a flat and hierarchical sensor network. Comparisons of the proposed methods with the standard centralized approach is performed and their good performance is illustrated by way of simulations. Approximations of random measures with a single Gaussian have slightly larger RMSE than approximations with Gaussian mixtures. The CRPF-based summaries have moderate error statistics but outperform the SPF with incorrect noise statistics. We have also investigated the effect of feedback from the FC and observed that feedback from the FC improves the individual sensor tracking performance.

Chapter 8

Target Tracking in a Two-Tiered Hierarchical Sensor Network

To deal with sensor nodes with limited energy supply and communication bandwidth we consider energy-efficient hierarchical architectures for solving the target tracking problem. In these networks, sensors form clusters and transmit minimal quantized information about a sensed event to a specialized node, known as a cluster head. Cluster heads are equipped with capability of communicating over large distances with a fusion center or a base station. In Section 8.2, the main components we define in greater detail the mathematical models associated with the sensor, cluster head and fusion center. We consider two different hierarchical architectures (a) the target dynamics are probabilistically estimated at the cluster heads and their statistics combined at the fusion center, and (b) the cluster heads perform simple compression rules on the quantized sensor data and the fusion center estimates the target dynamics using these severely compressed data. Sequential

Monte Carlo algorithms for estimation of the target dynamics are developed in Section 8.3. Through computer simulations the performances of these two architectures are studied in Section 8.4.

8.1 Introduction

A typical sensor network comprises of a large number of sensors, which perform tasks of sensing, local data processing and transmission of data to a central unit, known as a base station (BS) or a fusion center (FC). In flat network architectures, where all the sensors transmit the data to the FC either directly or route the data to the FC through intermediary sensor nodes, a considerable amount of sensor energy is dissipated in communication. In view of these drawbacks, hierarchical sensor network (HSN) architectures have been proposed [95, 96]. At the lowest level (tier 0), sensors form a cluster and a selected node receives the sensed data transmitted from the sensors in the cluster. The selected node is known as leader node (LN) or cluster head (CH). The CHs may form a second level (tier 1) of hierarchy. In a two-tiered hierarchical architecture, communication proceeds among (a) sensor nodes and CHs and (b) CHs and FC, thereby adverting the need for direct communication between sensors and FC. As a result, with CHs relatively closer to the sensors than the FC, the energy consumption by each sensor for data transmission is considerably reduced. While sensors may have multiple CHs, here we consider the situation where a sensor has only one CH.

Here, we address target tracking in two-tiered hierarchical sensor networks. The CHs are assumed capable of communicating directly with the FC [97]. The sensors use

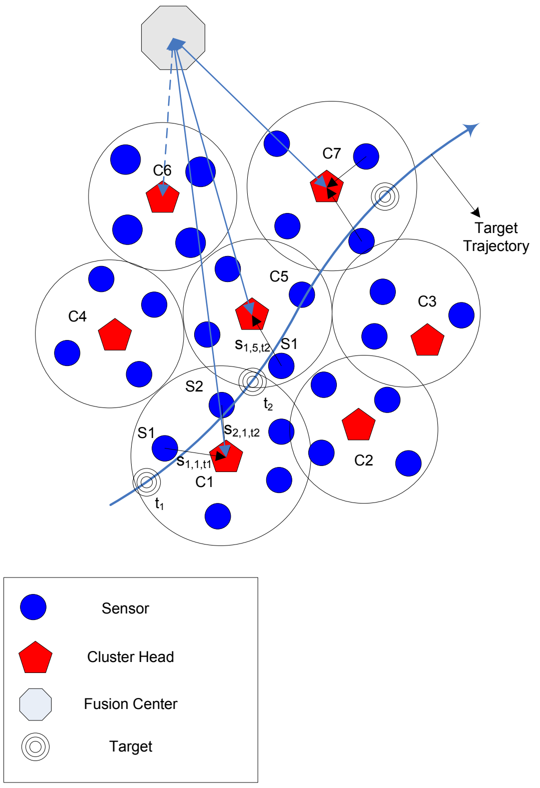


Figure 8.1: Hierarchical Sensor Network

a simple quantization scheme that results in transmission of a '`1`' when the target is in their vicinity and a '`0`' otherwise. We consider the following two HSN architectures.

- When CHs have medium to high computational resources, target tracking is performed at the CH using the quantized data. We term this architecture as HSN-Type I.

- When CHs have low computational resources, the CHs employ simple compression rules and transmit this compressed data to the FC. We term this architecture as HSN-Type II. Here, the compression rule accounts for the transmission of a single number that represents the number of sensors reporting the event of interest in their neighborhood.

Figure 8.1 shows an example of a HSN. We consider a cluster as active, when any of its sensors report events of activity in their neighborhood. Clusters are inactive if there are no sensors that sense any activity in their neighborhood. In the example, at time instant t_1 C1 is an active cluster while CHs C2 and C4 are inactive. At time instant t_1, when the target is within the vicinity of sensor S1, the sensor transmits a `1` to the CH C1, which processes the data and transmits them to the FC. At time instant t_2, sensors S2 and S1 of CHs C1 and C5 respectively, report the presence of the target. In a HSN-Type I, the CH C1 and C5 independently estimate the target dynamics and the FC combines these estimates, while in HSN-Type II, the CHs transmit the number of `1`'s, that each of them received from its sensors. We propose sequential Monte Carlo (SMC) algorithms for target tracking and data fusion in HSNs-Type I and Type II.

8.2 System Overview

8.2.1 Sensor Model

Following the work in [98], we model the strength of the signal emitted by target and received at the sensor as

$$y_{n,c,t} \quad = \quad \min\left(\boldsymbol{\Psi}_0, \frac{\boldsymbol{\Psi}_0 d_0^\alpha}{|\mathbf{r}_{n,c} - \mathbf{d}_t|^\alpha}\right) + v_{n,c,t} \tag{8.1}$$

where $\boldsymbol{\Psi}_0$ is the signal strength within a known reference distance d_0, $\mathbf{r}_{n,c} \in \mathbb{R}^2$ is the position of the n-th sensor in the c-th cluster, $\mathbf{d}_t \in \mathbb{R}^2$ denotes the location of the target at time t, $|\cdot|$ denotes norm (length) of a vector, α is the attenuation parameter and $v_{n,c,t} \sim N(\mu_v, \sigma_v^2)$ is a noise process with Gaussian probability density function whose statistics are assumed known. In equation (8.1), the unknown quantities are $\boldsymbol{\Psi}_0$ and $\mathbf{d}_t$. If the observed energy level $y_{n,c,t}$, exceeds a threshold γ, the sensor transmits a binary `0`, otherwise it transmits a `1` [85]. Mathematically this processing is modeled as

$$s_{n,c,t} = \begin{cases} 1 & \Leftarrow \quad y_{n,c,t} \geq \gamma \\ 0 & \Leftarrow \quad y_{n,c,t} < \gamma. \end{cases} \tag{8.2}$$

8.2.2 Cluster Head Model

In a HSN-Type I, the active CHs using the binary data, $s_{n,c,t} \in \{1, 0\}$, from the sensors, estimate the target dynamics and transmit these statistics to the FC. In a HSN-Type II, the CH transmits the number of active sensors in its cluster to the FC. The transmitted signal to the FC is modeled as

$$z_{c,t} = \sum_{n=1}^{N_c} s_{n,c,t} \tag{8.3}$$

where N_c is the number of sensors in the c-th cluster. The total number of clusters in the proposed networks is C.

8.2.3 Fusion Center Model

Reiterating, the role of the FC in HSNs-Type I, is to combine the estimates of the CHs when multiple clusters are active and to provide the initialization parameters of the SMC filter implemented on the active clusters. In these networks the FC does not require any location information of the sensors. In HSNs-Type II, the FC collects the severely quantized sensor data from the CHs and estimates the target statistics. In this type of network the FC needs to know the location of the sensors and the indices of the CHs to which they belong.

8.3 SMC Algorithms for Target Tracking

Denoting, as earlier $\boldsymbol{x}_t = [\dot{x}_{1,t}, \dot{x}_{2,t}, x_{1,t}, x_{1,t}, \Psi_t]^\top$, and defining $\mathbf{l}_t = [x_{1,t},\ x_{2,t},\ \Psi_t]^\top$ as the vector containing the position of the target and the reference signal strength and $\boldsymbol{v}_t = [\dot{x}_{1,t},\ \dot{x}_{2,t}]^\top$ as the velocity vector, we model the state transition equations as

$$\begin{aligned} \boldsymbol{v}_t &= \boldsymbol{v}_{t-1} + \mathbf{F}_{t-1}\mathbf{u}_t \\ \mathbf{l}_t &= \mathbf{G}^1_{t-1}\boldsymbol{v}_{t-1} + \mathbf{G}^2_{t-1}\mathbf{l}_{t-1} + \mathbf{G}^3_{t-1}\boldsymbol{v}_t \end{aligned} \tag{8.4}$$

where

$$\mathbf{F}_t = \begin{bmatrix} Ts & 0 \\ 0 & Ts \end{bmatrix}, \quad \mathbf{G}^1_{t-1} = \begin{bmatrix} \frac{Ts}{2} & 0 \\ 0 & \frac{Ts}{2} \\ 0 & 0 \end{bmatrix}, \mathbf{G}^2_{t-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{G}^3_{t-1} = \begin{bmatrix} \frac{Ts}{2} & 0 \\ 0 & \frac{Ts}{2} \\ 0 & 0 \end{bmatrix}$$

and $\mathbf{u}_t$ is a Gaussian noise process with zero mean and covariance matrix $\mathbf{C}_u$. The observation equation (8.1), can be rewritten as

$$y_{n,c,t} = h(x_{1:2,t}, \mathbf{r}_{n,c}) + v^c_{n,t} \tag{8.5}$$

where $h(x_{1:2,t}, \mathbf{r}_{n,c}) = \min\left(\boldsymbol{\Psi}_0, \frac{\boldsymbol{\Psi}_0 d_0^\alpha}{|\mathbf{r}_{n,c} - \mathbf{d}_t|^\alpha}\right)$. Within this framework, the objectives are

- Estimation of the posterior density $p(\boldsymbol{x}_t | s_{1,c,1:t}, \cdots, s_{N_c,c,1:t})$ by active CH c and fusion of the statistics of these densities of multiple CHs by the FC in a HSN-Type I.
- Estimation of the posterior density $p(\boldsymbol{x}_t | \mathbf{z}_{1,1:t}, \cdots, \mathbf{z}_{C,1:t})$ by the FC in a HSN-Type II.

8.3.1 CH Particle Filter Implementation in HSN-Type I

The following are the steps of the implementation of a particle filter at a CH in HSN-Type I networks.

1. *Initialization*: $\boldsymbol{x}_{t'}^{(m)} \sim \mathcal{N}(\boldsymbol{\mu}_{t'}, \Xi_{t'})$ where $\{\boldsymbol{\mu}_{t'}, \Xi_{t'}\}$ are provided by the FC and $m \in \{1 \cdots M\}$ denotes the particle index with M as the total number of particles.

2. *Particle generation* : The particles of the new state $\boldsymbol{x}_t^{(m)}$ are generated as

$$\begin{aligned}
\dot{x}_{(1,2),c,t}^{(m)} &\sim \mathcal{N}(\dot{x}_{(1,2),c,t-1}^{(m)}, T_s^2 \sigma_{(1,2),u}^2) \\
x_{(1,2),c,t}^{(m)} &= x_{(1,2),c,t-1}^{(m)} + \frac{T_s}{2}\left(\dot{x}_{(1,2),c,t}^{(m)} + \dot{x}_{(1,2),c,t-1}^{(m)}\right) \\
\Psi_{c,t}^{(m)} &\sim \mathcal{N}(\mu_{\Psi_{c,t-1}}, \sigma_{\Psi_{c,t-1}}^2). \qquad (8.6)
\end{aligned}$$

3. *Weight update and normalization:*

$$\begin{aligned}\tilde{w}_{c,t}^{(m)} &\propto w_{c,t-1}^{(m)}\prod_{n=1}^{N_c} p(s_{n,c,t}|\boldsymbol{x}_{0:t}^{(m)}) \\ w_{c,t}^{(m)} &= \frac{\tilde{w}_{c,t}^{(m)}}{\sum \tilde{w}_{c,t}^{(m)}}\end{aligned} \tag{8.7}$$

4. *Estimation of transmitted kernel parameters:*

$$\begin{aligned}\hat{\boldsymbol{\mu}}_{c,t} &= \sum_{m=1}^{M} w_{c,t}^{(m)}\boldsymbol{x}_{c,t}^{(m)} \\ \hat{\boldsymbol{\Sigma}}_{c,t} &= \sum_{m=1}^{M} w_{t}^{(m)}(\boldsymbol{x}_{c,t}^{(m)} - \hat{\boldsymbol{\mu}}_{c,t})(\boldsymbol{x}_{c,t}^{(m)} - \hat{\boldsymbol{\mu}}_{c,t})^{\top}.\end{aligned}$$

The likelihood in (8.8) is evaluated using the following expressions:

$$\begin{aligned}p(s_{n,c,t} = 1|\boldsymbol{x}_{0:t}^{(m)}) &= 1 - p(s_{n,c,t} = 0|\boldsymbol{x}_{0:t}^{(m)}) \\ &= \int_{\gamma}^{\infty} p(y_{n,c,t}|\boldsymbol{x}_{0:t}^{(m)})dy_{n,c,t} = Q\left(\frac{\gamma - h(x_{1:2,t}, \mathbf{r}_{n,c}) - \mu_v}{\sigma_v}\right).\end{aligned} \tag{8.8}$$

The FC collects the transmitted kernel parameters from the CHs and utilizes them in fusion of the posterior densities of multiple active CHs as described in the next subsection.

8.3.2 Fusion Center in HSN-Type I

We denote $\mathbf{l}_{c,t} = \{s_{1,c,t}, \cdots, s_{N_c,c,t}\}$ as the set of sensor measurements in cluster c at time instant t, $\mathbf{l}_{c,1:t} = \{s_{1,c,1:t}, \cdots, s_{N_c,c,1:t}\}$ as the set of sensor measurements from time instant 1 to t and

$\mathbf{l}_{1:C_a,t} = \{\mathbf{l}_{1,1:t}, \cdots, \mathbf{l}_{C_a,1:t}\}$ as the set of all sensor measurements in C_a clusters from time instant 1 to t. Let us consider measurements evolving from sensors of two different clusters ($C_a = 2$). The joint posterior density of the target dynamics can then be derived as

$$\begin{aligned} p(\boldsymbol{x}_{0:t}|\mathbf{l}_{1:2,1:t}) &= p(\boldsymbol{x}_{0:t}|\mathbf{l}_{1:2,t}, \mathbf{l}_{1:2,1:t-1}) \\ &\propto p(\mathbf{l}_{1,t}, \mathbf{l}_{2,t}|\boldsymbol{x}_t, \boldsymbol{x}_{0:t-1}, \mathbf{l}_{1:2,1:t-1}) p(\boldsymbol{x}_t|\boldsymbol{x}_{0:t-1}, \mathbf{l}_{1:2,1:t-1}) \\ &\qquad \times p(\boldsymbol{x}_{0:t-1}|\mathbf{l}_{1:2,1:t-1}) \\ &\propto \frac{p(\boldsymbol{x}_t|\mathbf{l}_{2,1:t})}{p(\boldsymbol{x}_t|\mathbf{l}_{1,1:t-1})} \frac{p(\boldsymbol{x}_t|\mathbf{l}_{2,1:t})}{p(\boldsymbol{x}_t|\mathbf{l}_{2,1:t-1})} p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) p(\boldsymbol{x}_{0:t-1}|\mathbf{l}_{1:2,1:t-1}). \end{aligned} \tag{8.9}$$

assuming independence among the sensor measurements $\{\mathbf{l}_{1,t}, \mathbf{l}_{2,t}\}$ conditioned on $\boldsymbol{x}_t$ [99]. Generalizing equation (8.9) for C_a number of active clusters, we have

$$p(\boldsymbol{x}_{0:t} \mid \mathbf{l}_{1:C_a,1:t}) \propto \prod_{c=1}^{C_a} \frac{p(\boldsymbol{x}_t \mid \mathbf{l}_{c,1:t})}{p(\boldsymbol{x}_t \mid \mathbf{l}_{c,1:t-1})} \times p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) p(\boldsymbol{x}_{0:t-1} \mid \mathbf{l}_{1:C_a,1:t-1}) \tag{8.10}$$

which is the optimal recursive fusion equation. The CHs approximate the sampled based representations of the distributions $p(\boldsymbol{x}_t \mid \mathbf{l}_{c,1:t-1})$ and $p(\boldsymbol{x}_t \mid \mathbf{l}_{c,1:t})$ as Gaussians, i.e., $p(\boldsymbol{x}_t \mid \mathbf{l}_{c,1:t-1}) \simeq \mathcal{N}(\tilde{\boldsymbol{\mu}}_{c,t}, \tilde{\boldsymbol{\Sigma}}_{c,t})$ and $p(\boldsymbol{x}_t \mid \mathbf{l}_{c,1:t}) \simeq \mathcal{N}(\hat{\boldsymbol{\mu}}_{c,t}, \hat{\boldsymbol{\Sigma}}_{c,t})$. Therefore, from (8.10) we have

$$\begin{aligned} p(\boldsymbol{x}_t \mid \mathbf{l}_{1:C_a,1:t}) &\propto \prod_{c=1}^{C_a} \frac{\mathcal{N}(\hat{\boldsymbol{\mu}}_{c,t}, \hat{\boldsymbol{\Sigma}}_{c,t})}{\mathcal{N}(\tilde{\boldsymbol{\mu}}_{c,t}, \tilde{\boldsymbol{\Sigma}}_{c,t})} \times \; p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) \, p(\boldsymbol{x}_{0:t-1} \mid \mathbf{l}_{1:C,1:t-1}) \\ &\qquad\propto \frac{\mathcal{N}(\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t)}{\mathcal{N}(\tilde{\boldsymbol{\mu}}_t, \tilde{\boldsymbol{\Sigma}}_t)} \, p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) \, p(\boldsymbol{x}_{0:t-1}|\mathbf{l}_{1:C_a,1:t-1}) \end{aligned}$$

where

$$\begin{aligned}
\hat{\boldsymbol{\Sigma}}_t^{-1} &= \hat{\boldsymbol{\Sigma}}_{1,t}^{-1} + \hat{\boldsymbol{\Sigma}}_{2,t}^{-1} + \cdots + \hat{\boldsymbol{\Sigma}}_{C_a,t}^{-1} \\
\hat{\boldsymbol{\mu}}_t &= \hat{\boldsymbol{\Sigma}}_t \left(\hat{\boldsymbol{\Sigma}}_{1,t}^{-1} \hat{\boldsymbol{\mu}}_{1,t} + \hat{\boldsymbol{\Sigma}}_{2,t}^{-1} \hat{\boldsymbol{\mu}}_{2,t} + \cdots + \hat{\boldsymbol{\Sigma}}_{C_a,t}^{-1} \hat{\boldsymbol{\mu}}_{C_a,t} \right) \\
\tilde{\boldsymbol{\Sigma}}_t^{-1} &= \tilde{\boldsymbol{\Sigma}}_{1,t}^{-1} + \tilde{\boldsymbol{\Sigma}}_{2,t}^{-1} + \cdots + \tilde{\boldsymbol{\Sigma}}_{C_a,t}^{-1} \\
\tilde{\boldsymbol{\mu}}_t &= \tilde{\boldsymbol{\Sigma}}_t \left(\tilde{\boldsymbol{\Sigma}}_{1,t}^{-1} \tilde{\boldsymbol{\mu}}_{1,t} + \tilde{\boldsymbol{\Sigma}}_{2,t}^{-1} \tilde{\boldsymbol{\mu}}_{2,t} + \cdots + \tilde{\boldsymbol{\Sigma}}_{C_a,t}^{-1} \tilde{\boldsymbol{\mu}}_{C_a,t} \right). \qquad (8.11)
\end{aligned}$$

A particle filter based implementation of the fusion equation (8.10), is similar to the CH particle filter implementation outlined in subsection 8.3.1 except for the weight calculation, which is performed as $\tilde{w}_t^{(m)} \propto \frac{\mathcal{N}(\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t)}{\mathcal{N}(\tilde{\boldsymbol{\mu}}_t, \tilde{\boldsymbol{\Sigma}}_t)}$. The posterior density is then approximated by a Gaussian distribution whose statistical terms are utilized in initializing the particle filter on the CHs that may be active in the next or future time instants.

8.3.3 FC Particle Filter Implementation in HSN-Type II

In a generic SMC algorithm, when the parameters of the state transition equation are known, the particles are easily generated and hence the main task lies in the calculation of the likelihood for updating the weights. The situation in HSN-Type II is also very similar. The initialization and particle generation proceeds as described in subsection 8.3.1. The expression for calculating the likelihood is given by

$$p(\mathbf{z}_t | \boldsymbol{x}_t) = \prod_{c=1}^{C} p(z_{c,t} | \boldsymbol{x}_t) = \prod_{c=1}^{C} p\left(\sum_{n=1}^{N_c} s_{n,c,t} | \boldsymbol{x}_t \right).$$

The probability mass functions $p(z_{c,t}|\boldsymbol{x}_t)$ can be obtained by summing over all possible combinations of sensor transmitted data such that $z_{c,t} = \sum_{n=1}^{N_c} s_{n,c,t}$. However the number of such combinations is relatively large for even small N_c and $z_{c,t}$. To reduce the computational complexity of the problem, we propose to estimate $s_{c,n,t}$ given $\{z_{c,t}, \boldsymbol{x}_t^{(m)}\}$. The task of assigning $s_{c,n,t} = \{\mathtt{1}, \mathtt{0}\}$ is performed as follows:

- Draw the samples of the state vector $\boldsymbol{x}_t^{(m)}$ and sensor measurements $y_{n,c,t}^{(m)}$ using (8.4) and (8.1), respectively.
- $\forall n \in \{1 \cdots N_c\}$ and $c \in \{1 \cdots C\}$, obtain $w_{n,c,t} = \frac{1}{M} \sum_m \mathrm{I}(y_{n,c,t}^{(m)} \geq \gamma)$ where $\mathrm{I}(\cdot)$ is the Bernoulli indicator function.
- $\forall c \in \{1 \cdots C\}$ sort the elements of the set $\{w_{1,c,t}, \cdots w_{N_c,c,t}\}$ in decreasing order and obtain the indices of the first $z_{c,t}$ sensors . Set the value of $s_{c,n,t}$ for the sensors with these indices to `1` and the remaining to `0`.

The likelihood is then approximated as

$$p(\mathbf{z}_t|\boldsymbol{x}_t) \approx \prod_{c=1}^{C} \prod_{n=1}^{N_c} p\left(s_{n,c,t}|\boldsymbol{x}_t\right) \tag{8.12}$$

and $p\left(s_{c,n,t}|\boldsymbol{x}_t\right)$ is obtained using (8.8).

8.4 Simulations, Results and Discussion

In our computer simulations we have considered a sensing field of dimensions $650m \times 750m$ with 50 CHs and 480 sensors in the sensing field deployed randomly (using stratified sampling methods). Sensors were clustered using standard

System Parameters	Value
Sensing Field Dimensions	$650m \times 750m$
No of Cluster Units	50
No of Sensors	480
Sensor Threshold γ	56.70
Reference Power Ψ_0	5000
No of particles	2000
Sampling Period	1s
Total Observation Period	60s
Mean of Sensor Noise μ_v	1
Variance of Sensor Noise μ_v	0.01
Observation noise parameter $\sigma^2_{1,u}$	0.1
Observation noise parameter $\sigma^2_{2,u}$	0.2

Table 8.1: System Parameters and their values

hierarchical clustering algorithms. Table 8.1 lists the values of the parameters used in simulating the target trajectory and sensor network. In the simulation of the SMC methods, particles were initially drawn from a Gaussian distribution with a known $\mu_0 = [0, 0, 5, 5]^\top$ and a covariance matrix Ξ=diag$(30, 30, 5, 5)$. One hundred target trajectories were generated for which the root mean squared error (RMSE) in estimating the target dynamics was computed.

In Fig 8.2, we plotted the RMSEs of the two networks for $\alpha = 2.5$ and spatially distributed known and unknown α. We modeled α as a spatially correlated truncated normal random variable $\sim \mathcal{N}_T(\mu_\alpha, \Xi_\alpha, a, b)$, with covariance matrix $\Xi_{\alpha,i,j} = cov(\alpha_i, \alpha_j) = e^{-0.3d_{ij}}$ where d_{ij} represents the Euclidean distance between points i and j. The spatial distribution is pictorially shown in 8.4The results show that the error in estimating the position of the target in the 2-D Cartesian coordinate system is around

10m and the errors in estimating the velocities are less than 1.6 m/s. Sensitivity on the knowledge of the attenuation parameter α in the proposed algorithms was more carefully studied. In another experiment, the attenuation parameter α was unknown, but constant for all the sensors. Figure 8.3 shows the RMSE errors when α was assumed to be $2, 3$, and 4 when its true value was 2.5. All these results suggest that when there are large errors in the assumed values of α, the RMSEs may be unacceptably large.

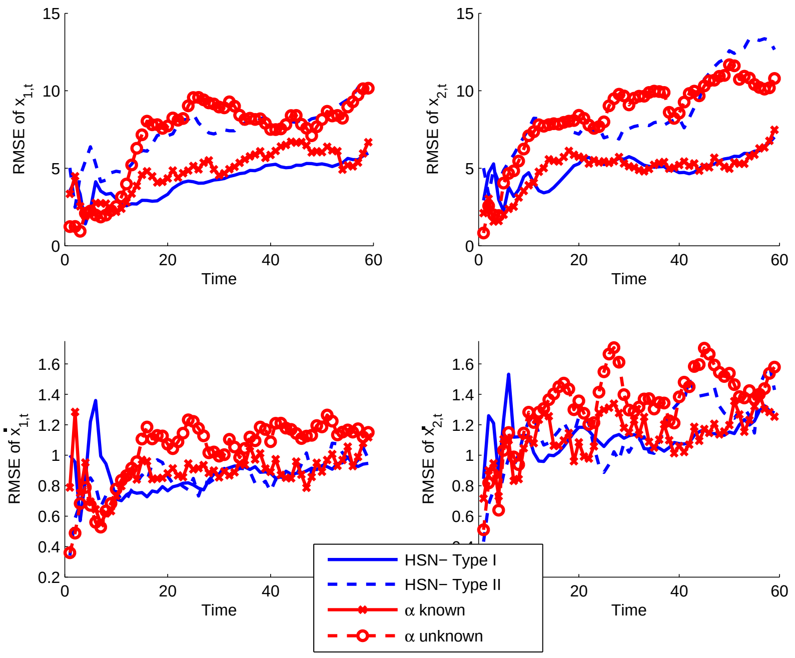


Figure 8.2: RMSE in HSNs-Type I and II with $\alpha = 2.5$ and spatially distributed α.

8.5 Summary

We have presented SMC algorithms for target tracking and data fusion in two-tiered HSNs with compressed sensor data. The proposed algorithms show

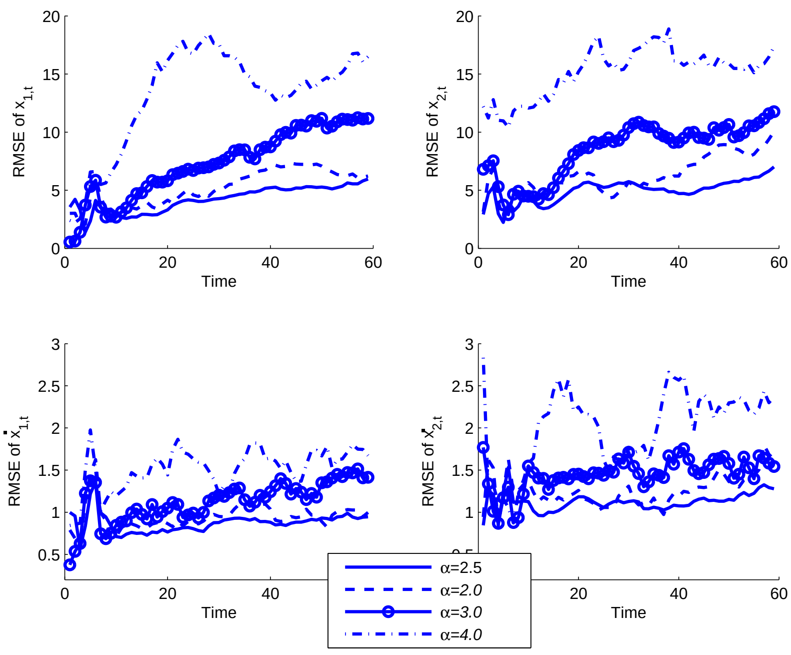


Figure 8.3: RMSE in HSNs-Type I, assumed $\alpha = 2.5, 2, 3, 4$, when true value is 2.5

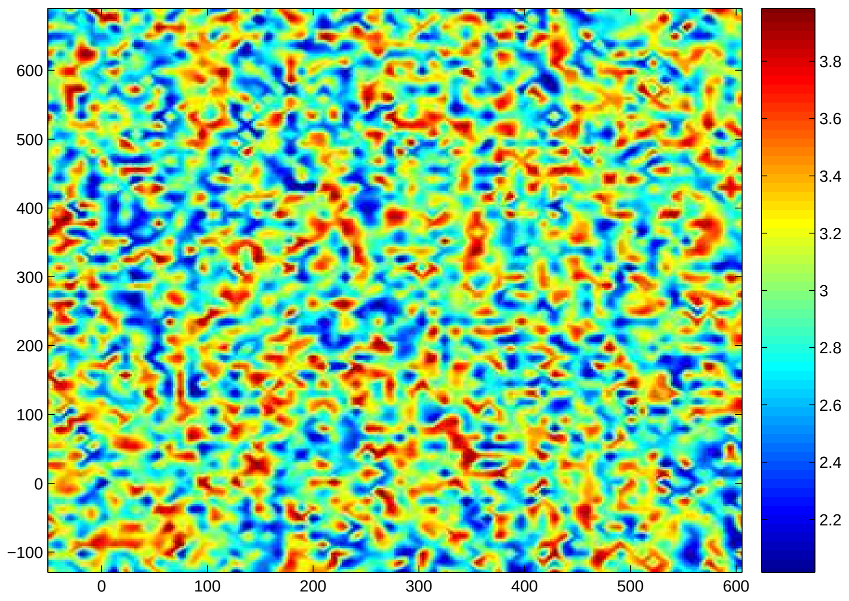


Figure 8.4: Spatial distribution of α.

good performance in estimating the dynamics of the target trajectory. We have also presented some simulation results that outline the sensitivity of the proposed algorithms when certain system parameters like the attenuation parameter of the sensor model are unknown.

Chapter 9

Target Tracking in an Asynchronous Wireless Sensor Network

Wireless sensor networks typically consist of a collection of sensor nodes, which acquire physical data related to the target dynamics, and a fusion center where the available data are processed together to sequentially estimate the target state (its instantaneous location and velocity). Very often, tracking algorithms are designed under the assumption that the network is synchronous, i.e., that the local clocks of the sensor nodes and the FC are perfectly aligned or, at least, that their offsets are known. In this chapter, we consider an asynchronous WSN, in which the local clocks of the sensors are misaligned and the corresponding offsets are unknown, and aim at designing recursive algorithms for optimal (Bayesian) tracking. In Section 9.2, we present the signal model and describe the estimation problem that we tackle. In particular, we propose sequential Monte Carlo (SMC) in Section 9.3 techniques that enable the approximation of the joint posterior probability distribution of the target state and the set of local clock offsets by means of a discrete probability measure with a random support. From this approximation, estimates of the target position and velocity, as well as of the clock offsets, can be readily derived. We illustrate the validity of the proposed approach and assess the performance of the resulting algorithms by means of computer simulations in Section 9.4.

9.1 Introduction

Wireless micro-sensors are transducers which "measure" a physical phenomenon and convert it into an information-bearing electrical signal through sampling, filtering and calibration. The latter, as well as other important steps of signal processing, depend on the internal clocks of the sensors. In the application of WSNs for target tracking, it is often assumed that the clocks of the sensors and the fusion center (FC) are synchronous, i.e., that all sensors sample the phenomenon of interest at the same instants or, alternatively, that the sampling times are possibly different but perfectly known at the FC. However, a misalignment in the clocks of the sensors and the FC due to the inherent drift in the clock frequencies occurs in practice [100]. Estimating and compensating the timing offsets at the sensors using time synchronization protocols results in a significant increase of the communication overhead in the WSN and, as a consequence, in an undesired reduction of the life of all battery-supported nodes [101]. Here, we tackle the problem of tracking a target when the sensors are asynchronous *and* their offsets are unknown.

The issue of handling the asynchronous activity of sensors has been addressed in [102], in the context of a multisensor-multitarget bias estimation problem, but assuming that all timing offsets are known. Similarly, in [103] sensor registration is performed using Kalman filtering for asynchronous sensors, but the misalignments of the sensor clocks are assumed known, too. In [104], the authors consider a localization system with asynchronous sensors and an object that periodically transmits a known signal. The inter-arrival time between the received signals is approximated and modeled as being statistically independent of the clock offsets, in order to enable localization using standard maximum-likelihood estimation (MLE) techniques.

In this chapter, we propose to use the sequential Monte Carlo (SMC) methodology [30] to jointly estimate the sensor offsets and the target trajectory and velocity. The problem is formally modeled as the joint Bayesian estimation of a set of static parameters and the time-varying state of a discrete-time random dynamical system. The SMC approach consists in approximating the posterior probability distribution of the random signals of interest using a discrete probability measure with random support, which enables the straightforward computation of estimates. This is not a simple task for our problem, though, since it is hard to guarantee convergence (to optimal solutions) of conventional SMC algorithms when static and dynamic random magnitudes must be handled together. The joint (static) parameter and (dynamic) state estimation problem has been previously addressed in [31], [59], [86], [105], [85]. In [31], an artificial evolution of the static parameters is proposed. In [59], the evolution of the parameters using kernel methods is described. A sampling scheme for fixed parameters that imposes restrictive assumptions on the probabilistic model is proposed in [86], while in [105] point-optimization methods are proposed. Finally, in [85], "density assisted" methods, which approximate the posterior distribution of static parameters using a model probability density function (pdf), are discussed. In this chapter, we propose two novel techniques, based on the general methodologies of [85] and [59].

9.2 Problem statement

We aim at recursively estimating the time-varying position and velocity of a target tracking that moves along a 2-dimensional region. The state of the target at continuous time t is $\boldsymbol{x}(t) = [x_1(t), x_2(t), \dot{x}_1(t), \dot{x}_2(t)]^\top \in \mathbb{R}^4$, where $[x_1(t), x_2(t)]^\top \in \mathbb{R}^2$ denotes the target location, $\dot{x}_i(t)$ is the time derivative of $x_i(t)$ and, therefore, $[\dot{x}_1(t), \dot{x}_2(t)]^\top \in \mathbb{R}^2$ is the target velocity vector at time t.

If the system is converted into discrete-time by sampling every T seconds (s), we obtain the dynamic state space (DSS) model,

$$\boldsymbol{x}_k = \mathrm{A}_T \boldsymbol{x}_{k-1} + \boldsymbol{u}_k, \quad k \in \mathbb{N}, \tag{9.1}$$

where $\mathbf{A}_T$ is a 4×4 transition matrix,

$$\boldsymbol{x}_k = [x_{1,k}, \dot{x}_{1,k}, x_{2,k}, \dot{x}_{2,k}]^\top = \boldsymbol{x}(kT)$$

is the target state at time kT (i.e., $[x_{1,k}, x_{2,k}] = [x_1(kT), x_2(kT)]^\top$ is the sampled position and $[\dot{x}_{1,k}, \dot{x}_{2,k}] = [\dot{x}_1(kT), \dot{x}_2(kT)]$ is the sampled velocity, both at time kT) and $\boldsymbol{u}_t \sim \mathcal{N}(0, \mathrm{C}_T)$ is zero-mean Gaussian noise with covariance matrix $\mathbf{C}_T$. Both the transition matrix, $\mathbf{A}_T$, and the noise covariance matrix, $\mathbf{C}_T$, are parameterized

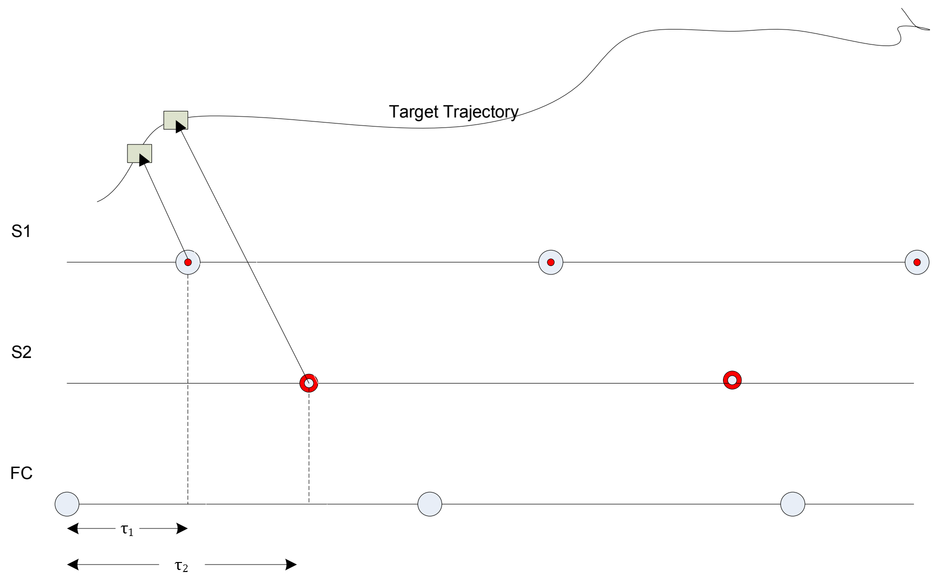


Figure 9.1: Timing diagram of the clocks of two sensors and the FC.

by T, specifically

$$\mathrm{A}_T = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathrm{C}_T = \sigma_u^2 \begin{bmatrix} \frac{T^3}{3} & \frac{T^2}{2} & 0 & 0 \\ \frac{T^2}{2} & T & 0 & 0 \\ 0 & 0 & \frac{T^3}{3} & \frac{T^2}{2} \\ 0 & 0 & \frac{T^2}{2} & T \end{bmatrix}.$$

Figure 9.1 depicts the timing diagram for the observations collected by two sensors, whose clocks are misaligned with the FC. Specifically, the knots in the line labeled “FC” denote the time instants at which the FC collects the data, while the knots in the lines labeled “S1” and “S2” indicate the sampling instants for sensors 1 and 2, respectively. We observe that the sensors have fixed, but random and unknown, offsets $\tau_1, \tau_2 > 0$, $\tau_1 \neq \tau_2$, hence they do not sample the function of the target trajectory in the same points.

In most target tracking applications, perfect timing is assumed (i.e., $\tau_1 = \tau_2 = 0$), which is hard to achieve in a practical situation. We study the effect of such offsets on the tracking problem. Let us denote time intervals as $t_k = [(k-1)T, kT)$, $k \in \mathbb{N}$. The set of observations that the FC receives from the n-th sensor during interval t_k is $\mathbf{y}_{n,t_k} = [y_{n,a,t_k}, y_{n,d,t_k}, y_{n,v,t_k}]^\top$, where

$$\begin{aligned}
y_{n,a,t_k} &= \tan^{-1}\left(\frac{x_{1,t_k+\tau_n} - s_{n,1}}{x_{3,t_k+\tau_n} - s_{n,2}}\right) + v_{n,a,t_k} && \text{(9.2a)}\\
y_{n,d,t_k} &= \sqrt{(x_{1,t_k+\tau_n} - s_{n,1})^2 + (x_{3,t_k+\tau_n} - s_{n,2})^2} \\
&\quad + v_{n,d,t_k} && \text{(9.2b)}\\
y_{n,v,t_k} &= \sqrt{(x_{2,t_k+\tau_n})^2 + (x_{4,t_k+\tau_n})^2} + v_{n,d,t_k}. && \text{(9.2c)}
\end{aligned}$$

The angle, distance and velocity observations given by (9.2a) (9.2b) and (9.2c) , respectively, depend on the sensor offset, τ_n, the target state at time $(k-1)T + \tau_n$, denoted as

$$\boldsymbol{x}_{t_k+\tau_n} = \mathbf{A}_{\tau_n} \boldsymbol{x}_{k-1} + \mathbf{u}_{t_k+\tau_n},$$

where $\mathbf{u}_{t_k+\tau_n} \sim N(\mathbf{0}, \mathbf{C}_{\tau_n})$, and the sensor positions, $\{s_{n,1}, s_{n,2}\}$. The measurement noise processes at the n-th sensor are denoted as v_{n,a,t_k}, v_{n,d,t_k} and v_{n,v,t_k} for angle, distance and velocity, respectively, and their pdf's are known. The complete set of observations during t_k is $\mathbf{y}_{t_k} = \{\mathbf{y}_{1,t_k}, \ldots, \mathbf{y}_{N_s,t_k}\}$, where N_s is the number of sensors in the WSN. Our objective is to estimate the sequence of target states $\boldsymbol{x}_{0:k} = \{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_k\}$ and clock offsets $\tau_{1:N_s} = \{\tau_1, \ldots, \tau_{N_s}\}$ using the measurements $\mathbf{y}_{t_1:t_k} = \{\mathbf{y}_{t_1}, \ldots, \mathbf{y}_{t_k}\}$.

9.3 Algorithms

All the statistical information needed to optimally solve the proposed estimation problem is contained in the *a posteriori* pdf $p(\boldsymbol{x}_{0:k}, \tau_{1:N_s}|\boldsymbol{y}_{t_1:t_k})$. Since the observations (9.2a) and (9.2b) are nonlinear, there is no feasible (optimal) analytical solution, and we propose to resort to SMC methodology [30]. SMC methods approximate the *a posteriori* pdf by means of a discrete random measure. The approximation with M particles takes the form

$$\begin{aligned} p_M(\boldsymbol{x}_{0:k}, \tau_{1:N_s}|\mathbf{y}_{t_1:t_k}) &= \sum_{m=1}^{M} w_k^{(m)} \delta(\boldsymbol{x}_{0:k} - \boldsymbol{x}_{0:k}^{(m)}) \times \\ &\quad \times \delta(\tau_{1:N_s} - \tau_{1:N_s}^{(m)}), \end{aligned} \tag{9.3}$$

where $w_k^{(m)}$ is the *importance weight* associated to the sample $(\boldsymbol{x}_{0:k}^{(m)}, \tau_{1:N_s}^{(m)})$ and $\delta(\cdot)$ denotes the Dirac delta function. A SMC method recursively updates the random measure approximation (9.3) when a new set of observations, $\mathbf{y}_{t_{k+1}}$, is received. It is difficult, in general, to guarantee the convergence of conventional SMC methods when there exist random (unknown) fixed parameters, because the dynamic system becomes non-ergodic [?]. In the sequel, we propose two algorithms that specifically take into account the fixed offsets, $\tau_{1:N_s}$, based on the density-assisted particle filtering (DAPF) methodology [47] and the sequential kernel approximation proposed by Liu and West [59].

9.3.1 Density Assisted Particle Filtering

The proposed SMC algorithm for the joint estimation of the target state and the sensor timing offsets is based upon the parametric approximation of the marginal posterior pdf of the n-th offset at time k by means of a beta pdf with properly chosen parameters[1], i.e., $p(\tau_n|\mathbf{y}_{t_1:t_k}) \approx \mathcal{B}(\tau_n; \pi_{n,k}, \phi_{n,k})$, which is updated, together with the importance weights, as new observations are collected. The steps of the algorithm are outlined below.

(i) *Intialization ($k = 0$)*: Target-state samples are drawn from the *a priori* pdf $p(\boldsymbol{x}_0)$. Offset samples are drawn from the single beta pdf $\mathcal{B}(\tau_n, 1, 1)$, i.e., we assume that, *a priori*, $p(\tau_n) = \mathcal{B}(\tau_n, 1, 1)$ for all n.

At time k, and given $\{\boldsymbol{x}_{0:k-1}^{(m)}, w_{k-1}^{(m)}\}_{m=1}^{M}$, the following recursive steps are taken.

(ii) *Particle propagation*: For each $n = \{1, 2, \cdots N_s\}$, timing offset samples are drawn from the beta-approximation of their marginal posterior pdf's at time $k-1$,

$$\tau_n^{(m)} \sim \mathcal{B}(\tau_n; \pi_{n,k-1}, \phi_{n,k-1}).$$

For each sample $m \in \{1, 2, \cdots M\}$, the offsets $\tau_{1:N_s}^{(m)}$ are sorted in the ascending order, i.e., we find a sequence $i_1, \ldots, i_{N_s}$ of distinct indices such that $i_k \in \{1, \ldots, N_s\}$ for all k and $\tau_{i_1}^{(m)} < \tau_{i_2}^{(m)} < \ldots < \tau_{i_{N_s}}^{(m)}$. State samples

$$\boldsymbol{x}_{t_k+\tau_{i_1}^{(m)}}^{(m)}, \ldots, \boldsymbol{x}_{t_k+\tau_{i_{N_s}}^{(m)}}^{(m)}$$

[1] $\mathcal{B}(\tau; \pi, \phi) = \frac{(\tau-a)^{\pi-1}(b-\tau)^{\phi-1}}{\beta(\pi,\phi)(b-a)^{\pi+\phi-1}}$ where $\beta(\pi, \phi) = \int_0^1 s^{\pi-1}(1-s)^{\phi-1} ds$ is the beta function, and a and b are the lower and upper bounds of τ.

are drawn using the (adequately parameterized) Markov state equation (9.1), and we denote the resulting set of particles as $\boldsymbol{\chi}_{t_k}^{(m)} = \{\boldsymbol{x}^{(m)}_{t_k+\tau^{(m)}_{1:N_s}}, \tau^{(m)}_{1:N_s}\}$. As an example, assume $N_s = 2$ and the offset samples $\tau_1^{(m)} < \tau_2^{(m)}$. We draw the state samples at the time instants $(k-1)T + \tau_1^{(m)}$ and $(k-1)T + \tau_2^{(m)}$ as follows

$$\begin{aligned}
\boldsymbol{x}^{(m)}_{t_k+\tau_1^{(m)}} &= A_{\tau_1^{(m)}} \boldsymbol{x}^{(m)}_{k-1} + \boldsymbol{u}^{(m)}_{t_k,1} \\
\boldsymbol{x}^{(m)}_{t_k+\tau_2^{(m)}} &= A_{\tau_2^{(m)}-\tau_1^{(m)}} \boldsymbol{x}^{(m)}_{t_k+\tau_1^{(m)}} + \boldsymbol{u}^{(m)}_{t_k,2}
\end{aligned}$$

where $\boldsymbol{u}^{(m)}_{t_k,1} \sim \mathcal{N}(\mathbf{0}, \mathrm{C}_{\tau_1^{(m)}})$ and $\boldsymbol{u}^{(m)}_{t_k,2} \sim \mathcal{N}(\mathbf{0}, \mathrm{C}_{\tau_2^{(m)}-\tau_1^{(m)}})$.

(iii) *Weight update*: Since the noise pdf's in (9.1), (9.2a), (9.2b) and (9.2c) are assumed known, the likelihood function, $p(\mathbf{y}_{t_k}|\boldsymbol{\chi}^{(m)}_{t_k})$, can be easily evaluated and the importance weights are recursively updated as

$$\tilde{w}_k^{(m)} = w_{k-1} p(\mathbf{y}_{t_k}|\boldsymbol{\chi}^{(m)}_{t_k}), \tag{9.4}$$

with

$$\begin{aligned}
p(\mathbf{y}_{t_k}|\boldsymbol{\chi}^{(m)}_{t_k}) = \prod_{n=1}^{N_s} p(y_{n,a,t_k}\,|\,\boldsymbol{\chi}^{(m)}_{t_k}) p(y_{n,s,t_k}\,|\,\boldsymbol{\chi}^{(m)}_{t_k}) \\
\times p(y_{n,v,t_k}\,|\,\boldsymbol{\chi}^{(m)}_{t_k}).
\end{aligned}$$

The weights in (9.4) need to be normalized, $w_k^{(m)} = \tilde{w}_k^{(m)} / \sum_{i=1}^{M} \tilde{w}_k^{(i)}$.

(iv) *Update of the posterior pdf's of the sensor offsets*: Using the set of weighted sensor offset samples, $\left\{w_k^{(m)}, \tau_n^{(m)}\right\}_{m=1}^{M}$, the sample mean and variance of τ_n are obtained,

$$\begin{aligned}\mu_{n,k} &= \sum_m w_k^{(m)} \tau_n^{(m)} \\ \sigma_{n,k}^2 &= \sum_m w_k^{(m)} (\tau_n^{(m)} - \mu_{n,k})^2.\end{aligned}$$

Then, the parameters of the beta-approximation to the posterior pdf of τ_n, $\pi_{n,k}, \phi_{n,k}$, can be calculated as

$$\begin{aligned}\pi_{n,k} &= \mu_{n,k} \left\{ \frac{\mu_{n,k}(1-\mu_{n,k})}{\sigma_{n,k}^2} - 1 \right\} \\ \phi_{n,k} &= (1-\mu_{n,k}) \frac{\pi_{n,k}}{\mu_{n,k}}. \end{aligned} \tag{9.5}$$

(v) *Estimation of parameter and states* : It is straightforward to approximate any moment of the posterior distribution of $\boldsymbol{x}_{t_k+\tau_{1:N_s}}$ and $\tau_{1:N_s}$ using the discrete probability measure given by the weighted set of particles $\{\boldsymbol{\chi}_{t_k}^{(m)}, w_k^{(m)}\}$. In particular, the target position and velocity at time $(k-1)T+\tau_n$ can be estimated as

$$\hat{\boldsymbol{x}}_{t_k+\tau_n} = \sum_{m=1}^{M} w_k^{(m)} \boldsymbol{x}_{t_k+\tau_n}^{(m)},$$

which is an approximation of the minimum mean square error (MMSE) estimate of $\boldsymbol{x}_{t_k+\tau_n}$ given $\mathbf{y}_{t_1:t_k}$. Similarly, $\mu_{n,k}$ is the (approximate) MMSE estimate of the sensor offset τ_n.

(vi) *Resample and Move*: Occasional resampling steps are needed in order to avoid the well-known phenomenon of weight degeneracy [30]. In our simulations, we perform a multinomial resampling step for every k. Following resampling of the

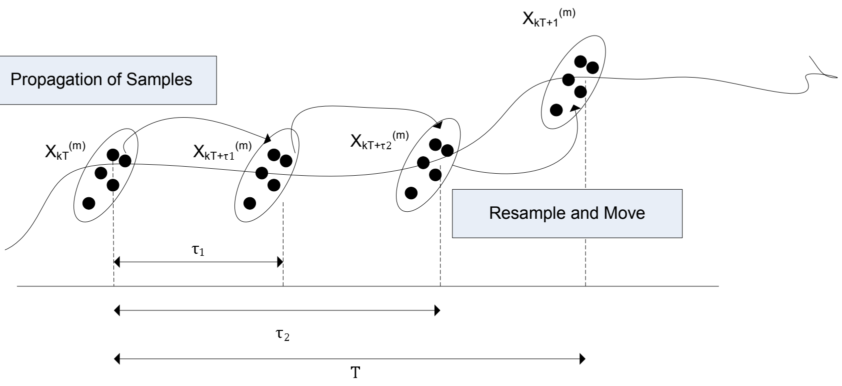


Figure 9.2: Series of operations during each time interval

sample $\{\boldsymbol{\chi}_{t_k}^{(m)}\}_{m=1}^{M}$, we obtain a new stream of particles $\{\tilde{\boldsymbol{\chi}}_{t_k}^{(m)}\}_{m=1}^{M}$. For each of this stream of particles, we first find the state particle corresponding to the maximum timing offset and then propagate the particle to time kT, i.e.,

$$\mathbf{x}_k^{(m)} = \mathbf{A}_{kT-\tau_{i_{N_s}}^{(m)}} \mathbf{x}_{t_k+\tau_{i_{N_s}}^{(m)}} + \mathbf{u}_k^{(m)},$$

where $\mathbf{u}_k^{(m)}$ is drawn from $\mathcal{N}(\mathbf{0}, \mathbf{C}_{kT-\tau_{i_{N_s}}^{(m)}})$.

Figure 9.2 is a graphical representation of the recursive steps of the proposed DAPF algorithm, where each cloud represents the state and sensor offset samples.

9.3.2 Liu and West algorithm (LW)

In the joint static-parameter and dynamic-state algorithm proposed in [59], a Gaussian mixture density is used for an artificial evolution of the fixed parameters. Clearly we cannot use a Gaussian mixture density for an artificial evolution of the sensor offset parameter τ_n because it is bounded, i.e., $0 \leq \tau_n \leq T$. Therefore, we propose the following truncated Gaussian mixture density for the evolution of τ_n,

$$p(\tau_{n,k+1} \,|\, \mathbf{y}_{t_1:t_k})) = \sum_k w_k^{(m)} T\mathcal{N}_{(0,T)}(\tau_{n,k+1}; \hat{\tau}_{n,k}^{(m)}, h^2\sigma_{n,k}^2) \tag{9.6}$$

where $T\mathcal{N}_{(0,T)}(\cdot)$ is the normal distribution truncated outside of the interval $(0, T)$, with mean $\hat{\tau}_{n,k}^{(m)}$ and variance $h^2\sigma_{n,k}^2$, the k subscript in $\tau_{n,k}^{(m)}$ is due to the artificial time evolution (not to the model dynamics) and $\mu_{n,k}$ and $\sigma_{n,k}^2$ are the sample mean and variance of τ_n at time k. The choices $\hat{\tau}_{n,k}^{(m)} = a\tau_{n,k}^{(m)} + (1-a)\mu_{n,k}$ and $h^2 = 1 - a^2$ ensure that the mixture density preserves the original sample mean and variance (this technique is termed "shrinkage" in [59]).

We now summarize the main steps of the algorithm. At time k, we have available $\{\boldsymbol{x}_{0:k-1}^{(m)}, \tau_{1:N_s}^{(m)}, w_{k-1}^{(m)}\}_{m=1}^M$ and receive the new observation $\mathbf{y}_{t_k}$.

(i) *Estimation of prior estimates*: For each $n = 1 \cdots N_s$ compute $\hat{\boldsymbol{x}}_{t_k+\tau_n^{(m)}}$ as

$$\hat{\boldsymbol{x}}_{t_k+\tau_n^{(m)}}^{(m)} = \mathrm{A}_{\tau_n^{(m)}} \boldsymbol{x}_{k-1}^{(m)}$$

(ii) *Sampling of sensor offset parameters*: Consider a discrete random variable V which takes values on the set $\{1, \ldots, M\}$ with probabilities proportional to $p(\mathbf{y}_{t_k} | \hat{\boldsymbol{x}}_{t_k+\tau_{1:N_s}^{(m)}}, \tau_{1:N_s}^{(m)})$. Draw M times from V and denote this set of samples as $\{V^{(1)}, \ldots, V^{(M)}\}$. Corresponding to each of these elements, draw $\tau_n^{(j)}$ from the $V^{(j)}$-th kernel of the truncated gaussian mixture in (9.6).

(iii) *Sampling target state parameters*: For each $m = 1, \ldots, M$ and each $n = 1, \ldots, N_s$, compute a state particle using the Markov prior (9.1), as described in step (ii) of the DAPF algorithm, i.e., compute the aggregated particles $\boldsymbol{\chi}_{t_k}^{(m)}$,

$m = 1, \dots, M$.

(iv) *Evaluation of the weights*:

$$w_k^{(m)} \propto \frac{p(\mathbf{y}_{t_k} | \boldsymbol{x}^{(m)}_{t_k + \tau^{(m)}_{1:N_s}})}{p(\mathbf{y}_{t_k} | \hat{\boldsymbol{x}}^{(V^{(m)})}_{t_k + \tau^{(V^{(m)})}_{1:N_s}})} \tag{9.7}$$

The likelihood factors are calculated as described in step (iii) of the DAPF algorithm. The estimation, resampling and move steps are also performed in the same way as for the latter algorithm.

9.4 Simulations

Consider a network with $N_s = 3$ sensors and observation period $T = 1$ s. The initial pdf of the target state is $p(\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{\mu}_0, \mathbf{K}_0)$, with $\boldsymbol{\mu}_0 = [0; 0.5; 0.0; 0.05]^\top$ and $\mathbf{K}_0 = diag\{1, \quad 1, \quad 0.01, \quad 0.01\}$. The measurement noise processes are Gaussian, with zero-mean and standard deviations $\sigma_{v_a} = 0.01$, $\sigma_{v_d} = 0.5$ and $\sigma_{v_v} = 0.1$ for angle, distance and velocity, respectively. We have set the number of particles to $M = 3000$ for all SMC algorithms.

We study the proposed methods for the scenario when the three sensors have different offsets, $\tau_1 = 0.2T$, $\tau_2 = 0.5T$, and $\tau_3 = 0.8T$. The nth sensor timing offset particles are all initially drawn from $\beta(\tau_n, 1, 1)$.

As an illustrative example, Fig 9.3 shows the estimation of a complete target trajectory for a single simulation run, using DAPF and LW algorithms. For comparison, we also depict the trajectory estimate obtained with a standard SMC

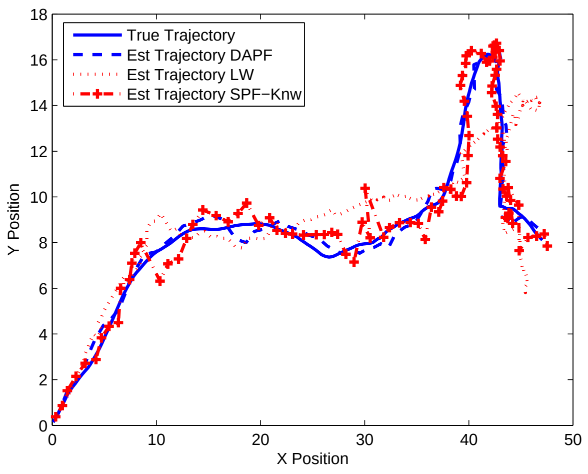


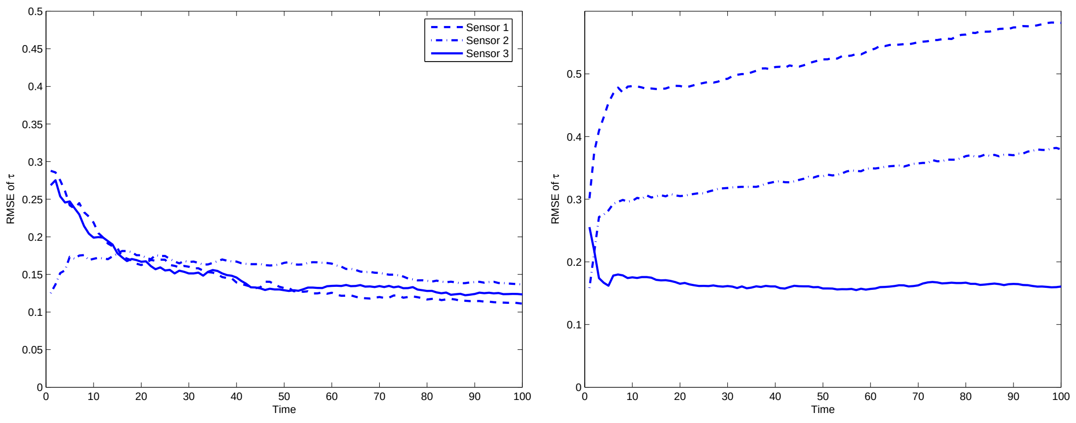
Figure 9.3: Trajectory and its estimates using the DAPF algorithm

algorithm with perfect knowledge of the timing offsets (labeled “SPF-Knw”). It can be seen that all three algorithms can track the target along a highly nonlinear trajectory. From this single trial, the LW algorithm seems to be the weakest technique.

In order to statistically asses the performance of the algorithms, we have considered the root mean square error (RMSE) as a figure of merit. In particular, for $L = 100$ independent simulation runs we have evaluated

$$\begin{aligned} RMSE_{\tau_{n,k}} &= \frac{1}{L}\sum_{l=1}^{L}(\hat{\tau}_{n,k,l} - \tau_{n,l})^2 \\ RMSE_{x_{i,k}} &= \frac{1}{L}\sum_{l=1}^{L}(\hat{x}_{i,k,l} - x_{i,k,l})^2 \\ RMSE_{\dot{x}_{i,k}} &= \frac{1}{L}\sum_{l=1}^{L}(\hat{\dot{x}}_{i,k,l} - \dot{x}_{k,l})^2, \end{aligned}$$

where $\tau_{n,l}$, $x_{i,k,l}$ and $\dot{x}_{i,k,l}$ are the nth offset, the ith state dynamic variable ($i \in \{1,2\}$) at time kT and its derivative, respectively, all of them for the lth simulation run. Their



(a) DAPF (b) LW

Figure 9.4: The RMSE of $\tau_{1:3}$ with the DAPF and LW algorithm

corresponding estimates are denoted as $\hat{\tau}_{n,k,l}$ (at time kT), $\hat{x}_{i,k,l}$ and $\hat{x}_{i,k,l}$, respectively.

Figure 9.4(a) shows the RMSEs obtained for the estimation of the three offsets, τ_1, τ_2, and τ_3, using the DAPF algorithm. The corresponding RMSEs resulting from the application of the LW technique are shown in Figure 9.4(b). The RMSEs obtained with the DAPF algorithm are similar for the three sensors, and clearly smaller than the RMSEs obtained with the LW method. This is a consequence of the misconvergence of the LW algorithm for some scenarios.

Figure 9.5 plots the RMSEs obtained for the estimation of the targets dynamics for the DAPF and SMC-Knw algorithms. We have found that the DAPF technique is consistently better than the conventional SMC-Knw algorithm, with known offsets, *for this particular scenario*. We conjecture that this is due to the smoothing effect of using a set of different offsets for each particle, which means that we assess the quality of the particle (through the computation of the corresponding likelihoods and weights) using a longer segment of the target state realization.

Finally, we show an example of how the estimation of the offsets with the DAPF

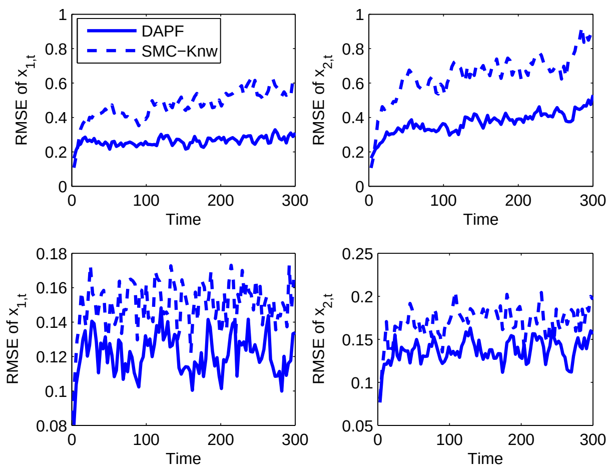


Figure 9.5: The RMSE of $\boldsymbol{x}_t$ with the DAPF and SMC-Knw algorithm. Upper left: RMSE of $x_{1,k}$. Upper right: RMSE of $x_{2,k}$. Lower left: RMSE of $\dot{x}_{1,k}$. Lower right: RMSE of $\dot{x}_{2,k}$.

algorithm evolves with time in a single simulation trial. In particular, Figure 9.6 shows the marginal posteriors of the sensor offsets, approximated by beta distributions, at time instants $t = 25, 50, 75$ and 100 s. It can be seen how the modes of the beta density get closer to the true offset values and then become narrower with time. Alternatively, Figure 9.7 plots the time evolution of the offset estimators ($\hat{\tau}_n$, $n = 1, 2, 3$). We can see how the three estimators get close to the true values.

9.5 Summary

In this chapter we have addressed the problem of target tracking in an asynchronous sensor network. We have proposed sequential Monte Carlo algorithms for the joint estimation of the sensor offsets and target state dynamics. In the DAPF algorithm, the marginal distributions of the sensors offsets are approximated by a

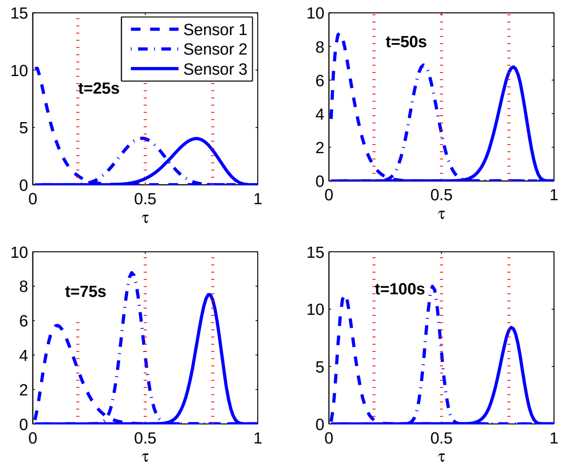


Figure 9.6: The evolution of the beta distribution for each of the sensor offsets at time instants t=25,50,75,100s

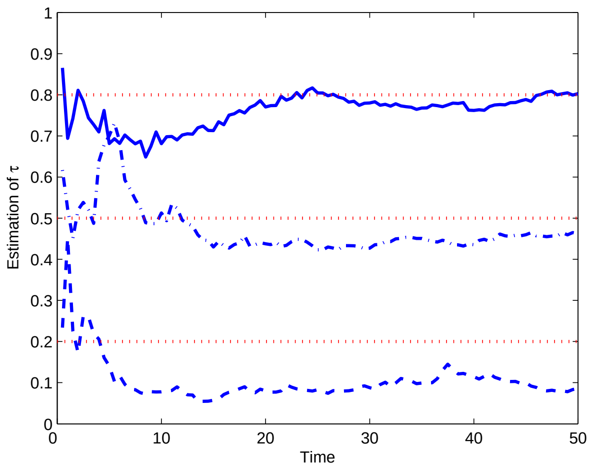


Figure 9.7: An estimate of the sensor offsets. The dashed horizontal lines are the true sensor offsets

generalized beta distribution. Using these distributions, offset samples are generated at each time interval, which are, in turn, used to propagate the target state samples. In the LW algorithm , the marginal distributions of the sensors offsets are approximated by a mixture of truncated Gaussian kernels. Through computer simulations we have compared these two methods and observed that the DAPF algorithm has a clearly better performance than the LW algorithm for the joint tracking of sensor offsets and target dynamics.

Some potential limitations of these techniques should also be remarked, though. It has been observed that, in a few cases, due to the apparent non-linearity of the offset parameters through the covariance matrix of the process noise in the DSS model, the marginal posterior of the sensor offsets is multimodal and the DAPF method can get stuck at local maxima. Good initialization or informative prior knowledge of the sensor offsets may be helpful. Also when very few offset samples have non-negligible weights and are concentrated in a particular region, approximation of the posterior using parametric densities does not seem to contribute much to sample diversity. This may be particularly harmful in scenarios where the filter gets stuck at a local maxima. A potential problem with the LW algorithm is that when the sensor offset estimates or target dynamic estimates are poor, the denominator in (9.7) may get close to zero and cause the particle filter to diverge.

Chapter 10

Performance Comparison of Gaussian-based Filters using Divergence Measures

In many situations solutions to non-linear discrete-time filtering problems are available through approximations. Many solutions are based on approximating the posterior distributions of the states with Gaussian distributions. We compare the performance of Gaussian-based filters including the extended Kalman filter, the unscented Kalman filter and the Gaussian particle filter. To that end, we measure the distance between the posteriors obtained by these filters and the one estimated by a sequential Monte Carlo (particle filtering) method. In Section 10.1, we formulate the problem, and in Section 10.2, we provide as a metric, the Kullback-Leibler and χ^2 distance measures. Through computer simulations we rank the performance of the three filters in Section 10.3

10.1 Introduction

The estimation of the filtering or posterior distribution $p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t})$ or their statistics is a standard filtering problem. Closed form analytical and optimal solutions to this filtering problem exist in a small number of cases, for example, when the functions $g(\cdot)$ and $h(\cdot)$ of the DSS model are linear, and the noise vectors $\mathbf{u}_t$ and $\mathbf{v}_t$ are zero-mean Gaussian with covariance matrices $\mathbf{C}_{u,t}$ and $\mathbf{C}_{v,t}$, respectively. However, in many real world scenarios, closed form solutions cannot be obtained [31].

Some approximate parametric solutions to these problems are obtained via the extended Kalman filter (EKF) and the unscented Kalman filter (UKF) [23], [24]. With the advent of more computing power, the last decade has seen a surge in Monte Carlo methods where the posterior distributions are approximated by a large weighted set of samples. These sequential Monte-Carlo (SMC) methods, also known as particle filters (PFs), have been proposed in the last decade as more robust and close to optimal solutions in determining posterior distributions and their statistics [30]. The Gaussian particle filter (GPF) is a SMC method which approximates the posterior distributions with Gaussian distributions [32]. In this chapter, we refer to the EKF, UKF, and GPF as Gaussian-based filters.

We reiterate that the common thread of these methods is that they approximate the sought posterior by a Gaussian. However, the process by which this approximation is obtained is different and therefore the resulting distributions are also different. The EKF achieves the Gaussian approximation through a linearization of the DSS model, the UKF obtains it by means of an unscented transformation while the GPF obtains the parameters of the approximations using SMC steps.

Popular metrics for assessing the performance of these approximations are through root mean square errors (RMSEs) of the point estimates of the states. An alternative is to estimate biases and variances of the estimates, thereby capturing only a limited picture of the filter's performance. Here, we provide a more complete performance comparison among the methods by measuring how close their posterior distributions are from the posterior distribution obtained by particle filtering. This has been suggested recently in [106]. The reason for choosing the posterior obtained by particle filtering is that the true posterior is not known and that particle filtering has the most ambitious aim while estimating unknown states by attempting to track the evolution of their posterior distributions. For measuring distance, we use divergence measures, more specifically the Kullback-Leibler (KL) and χ^2 divergence metrics. We develop and discuss the computation of these metrics in the context of the performance comparison of the filters.

10.2 Divergence metrics

Most recursive solutions to the filtering problem involve two key operations at each time instant: (a) propagation of the state estimate from the previous time instant to the current time instant and (b) updating of the state estimate using the current measurements. In the Gaussian-based filters the following approximations are made: the predictive density of the state is approximated by a Gaussian, $p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t-1}) \approx \mathcal{N}(\bar{\boldsymbol{x}}_{t|t-1}, \mathbf{P}_{t|t-1})$, where $\bar{\boldsymbol{x}}_{t|t-1}$ and $\mathbf{P}_{t|t-1}$ are the predictive mean and covariance matrix of $\boldsymbol{x}_t$ given $\boldsymbol{y}_{1:t-1}$, and the filtering density is approximated by another Gaussian, $p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t}) \approx \mathcal{N}(\bar{\boldsymbol{x}}_{t|t}, \mathbf{P}_{t|t})$ where $\bar{\boldsymbol{x}}_{t|t}$ and $\mathbf{P}_{t|t}$ are the mean and covariance matrix of $\boldsymbol{x}_t$ given $\boldsymbol{y}_{1:t}$.

For measuring the distance between the Gaussian posteriors produced by the Gaussian-based filters and the posterior obtained by the SPF,we use the KL and χ^2 divergence metrics [107]. However, the proposed approach for measuring performance is general and can be applied to a broad class of divergence measures.

With a slight abuse of notation, the KL and the χ^2 divergences are defined as follows:

- KL Divergence:
$$\mathcal{KL}(p,q) = \int p(\boldsymbol{x}) \log\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\right)\, d\boldsymbol{x}$$
- χ^2 Divergence:
$$\chi^2(p,q) = \int \frac{(p(\boldsymbol{x}) - q(\boldsymbol{x}))^2}{q(\boldsymbol{x})}\, d\boldsymbol{x} = \int \frac{p^2(\boldsymbol{x})}{q(\boldsymbol{x})}\, d\boldsymbol{x} - 1.$$

In our problem, $p(\boldsymbol{x})$ is the posterior obtained by the SPF and $q(\boldsymbol{x})$ is the posterior estimated by a Gaussian-based filter. A straightforward computation of these measures is not feasible because the random measures obtained by the SPF are discrete while the Gaussian posterior approximation is continuous. We avoid this by computing

$$\begin{aligned} p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t}) &\propto p(\boldsymbol{y}_t \mid \mathbf{x}_t, \mathbf{y}_{1,t-1}) p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) \\ &\propto p(\boldsymbol{y}_t \mid \boldsymbol{x}_t) \int p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) \times p(\boldsymbol{x}_{t-1} \mid \boldsymbol{y}_{1:t-1}) d\boldsymbol{x}_t. \end{aligned} \tag{10.1}$$

In the integral in (10.1), we express the posterior $p(\boldsymbol{x}_{t-1} \mid \boldsymbol{y}_{1:t-1})$ as

$$p(\boldsymbol{x}_{t-1} \mid \boldsymbol{y}_{1:t-1}) \;=\; \sum_{m=1}^{M} w_{t-1}^{(m)} \delta(\boldsymbol{x}_{t-1} - \boldsymbol{x}_{t-1}^{(m)}) \tag{10.2}$$

and for the posterior $p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t})$ we obtain

$$p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t}) \;\propto\; p(\boldsymbol{y}_t \mid \boldsymbol{x}_t) \sum_{m=1}^{M} w_{t-1}^{(m)} p(\boldsymbol{x}_t \mid \mathbf{x}_{t-1}^{(m)}). \tag{10.3}$$

We will write $p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t}) = c\tilde{p}(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t})$ where

$$\tilde{p}(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t}) \;=\; p(\boldsymbol{y}_t \mid \boldsymbol{x}_t) \sum_{m=1}^{M} w_{t-1}^{(m)} p(\boldsymbol{x}_t \mid \mathbf{x}_{t-1}^{(m)}). \tag{10.4}$$

10.2.1 Computation of $\mathcal{KL}$ divergence

With a slight abuse in notation, the $\mathcal{KL}$ distance at time instant t is expressed as

$$\mathcal{KL}_t \;=\; \int p \log\left(\frac{p}{q}\right) d\boldsymbol{x}_t = \int \frac{p}{q} \log\left(\frac{p}{q}\right) q d\boldsymbol{x}_t. \tag{10.5}$$

The integral in (10.5) can by computed by Monte Carlo integration by drawing samples from $q(\boldsymbol{x}_t)$. It is easy to show that

$$\mathcal{KL}_t \;\simeq\; \frac{1}{N} \sum_{n=1}^{N} \rho_t^{(n)} \log \rho_t^{(n)} \tag{10.6}$$

where N is the number of drawn samples from $q(\boldsymbol{x}_t)$ and

$$\rho_t^{(n)} = \frac{\tilde{\rho}_t^{(n)}}{\sum_{k=1}^{N} \tilde{\rho}_t^{(k)}} \tag{10.7}$$

and $\tilde{\rho}_t^{(n)} = \frac{\tilde{p}(\boldsymbol{x}_t^{(n)}|\boldsymbol{y}_{1:t})}{q(\boldsymbol{x}_t^{(n)})}$.

10.2.2 Computation of χ^2 divergence

The χ^2 at time instant t is given by

$$\begin{aligned} \chi_t^2 &= \int \frac{p^2}{q} d\boldsymbol{x}_t - 1 \\ &= \int \frac{p^2}{q^2}\, q\, d\boldsymbol{x}_t - 1. \end{aligned} \tag{10.8}$$

A Monte Carlo estimate of χ_t^2 is obtained in a similar way, and it is given by

$$\chi_t^2 \simeq \frac{1}{N} \sum_{n=1}^{N} \left(\rho_t^{(n)}\right)^2 - 1 \tag{10.9}$$

where the symbols have the same meaning as in (10.6).

10.3 Simulations

We provide two examples where we compare the performance of the EKF, UKF, and GPF.

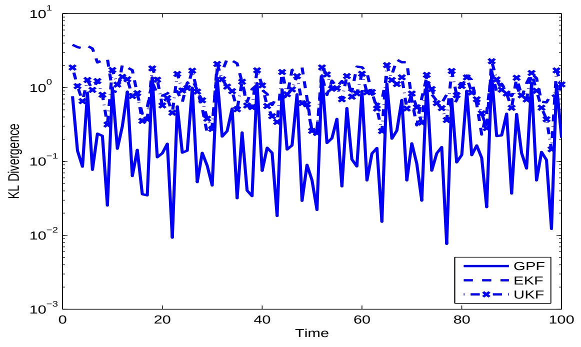


Figure 10.1: $\mathcal{KL}_t$ divergence of the GPF, EKF and UKF filters.

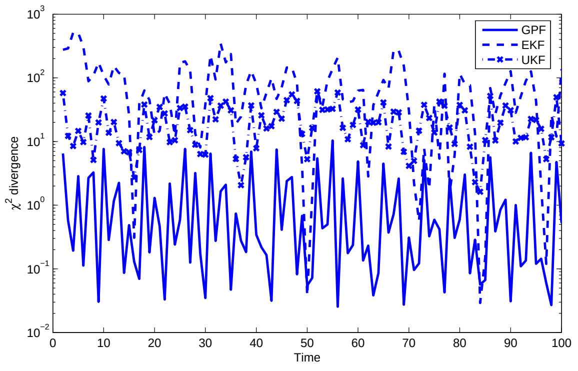


Figure 10.2: χ^2_t divergence of the GPF, EKF and UKF filters.

10.3.1 Univariate Non-Linear Model

Consider the following one dimensional non-linear time series

$$
\begin{aligned}
x_t &= 0.5x_{t-1} + 25\frac{x_{t-1}}{1+x_{t-1}^2} + 8\cos(1.2(t-1)) + u_t \\
y_t &= \frac{x_t^2}{20} + v_t.
\end{aligned}
\tag{10.10}
$$

Here, u_t and v_t are both zero mean Gaussian noise processes with unit variance. The initial distribution $p(x_0) \sim \mathcal{N}(0,1)$ [32]. In the simulation of the Monte Carlo

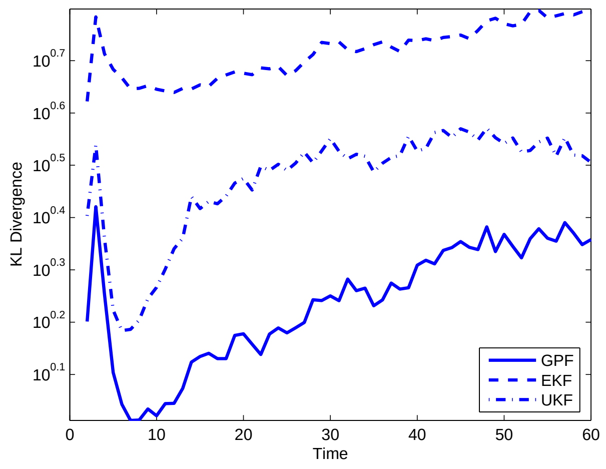


Figure 10.3: $\mathcal{KL}_t$-Divergence for the GPF, EKF and UKF filters

filters, $M = 1000$ particles were used. The number of samples N that were generated for computing the divergence measures was also 1000. For obtaining the weights associated with each sigma point of the UKF, the parameters α, β, and κ were set to $\alpha = 1, \beta = 0, \kappa = 2$ (as defined in [24]). We measured the performance of the algorithms by computing the divergence metrics using $K = 100$ different trajectories. In Figs. 10.1 and 10.2, the averaged $\mathcal{KL}_t$ and χ^2_t divergences are shown. From the plots, it can be seen that the Gaussian approximation to the posterior distribution with GPF is considerably better than that of the EKF and the UKF.

10.3.2 Bearings only target tracking

We considered a target tracking using angle measurements, which is a four-dimensional DSS model, with partially observed states

$$\boldsymbol{x}_t = \mathbf{F}\boldsymbol{x}_{t-1} + \mathbf{u}_t, \quad t \in \mathbb{N}, \tag{10.11}$$

where $\boldsymbol{x}(t) = [x_1(t), x_2(t), \dot{x}_1(t), \dot{x}_2(t)]^\top$ are the target's position and velocity in a two-dimensional plane, $\mathbf{F}$ is a 4×4 transition matrix, and $\mathbf{u}_t$ is a 4×1 Gaussian noise vector. The transition matrix $\mathbf{F}$ and the covariance matrix of the noise process are given by

$$\mathbf{F} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{C}_{u,t} = \sigma_u^2 \begin{bmatrix} \frac{T^3}{3} & \frac{T^2}{2} & 0 & 0 \\ \frac{T^2}{2} & T & 0 & 0 \\ 0 & 0 & \frac{T^3}{3} & \frac{T^2}{2} \\ 0 & 0 & \frac{T^2}{2} & T \end{bmatrix}$$

where T is the sampling interval.

The measurement equation is given by

$$y_t = \arctan\left(\frac{x_2(t)}{x_1(t)}\right) + v_t \tag{10.12}$$

where v_t is a measurement noise considered zero mean and Gaussian.

The initial distribution of $p(\boldsymbol{x}_0)$ is modeled as a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_0, \mathbf{P}_0)$ where $\boldsymbol{\mu}_0 = [-10.0,\ -0.5,\ -10.0,\ 0.5]^\top$ and $\mathbf{P}_0$ is given as *diag*[1]([0.25, 0.25, 0.25, 0.25]). The values of the other parameters were $\sigma_u = 0.1$ and $\sigma_v = 0.001$. In the Monte Carlo SPF and GPF, we used $M = 3000$ particles. The initial particles were all drawn from $p(\boldsymbol{x}_0)$. Similarly, when implementing the EKF and UKF, the initial mean and covariance matrix of the state vector were those used in $p(\boldsymbol{x}_0)$. As in the earlier simulation, here too we computed the divergences by averaging them over $K = 100$ different trajectories. In Figs. 10.3 and 10.4, we present the obtained results. Again, the performance of the GPF is better than the

[1]The diag($\mathbf{x}$) operation refers to the formation of the diagonal matrix with vector $\boldsymbol{x}$ along the main diagonal.

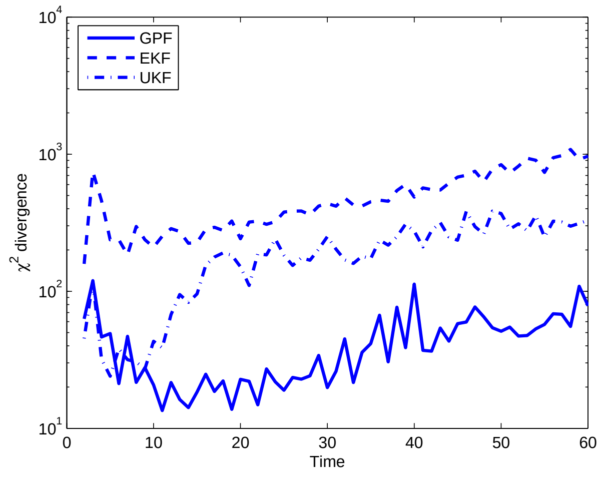


Figure 10.4: χ_t^2-Divergence for the GPF, EKF and UKF filters

other two filters.

10.4 Conclusions

We have provided a novel method for computing the divergences which measure the distance between the posterior distributions obtained with the SPF and the Gaussian-based filters EKF, UKF and the GPF. Through simulation studies, we note that the GPF provides a better approximation to the posterior distribution than the EKF and the UKF.

Chapter 11

Future Work

In this dissertation, we have addressed using Monte Carlo methods, problems related to target tracking and sensor localization. All these problems are addressed under known and unknown probability distributions of the the noise processes. We now highlight some of the possible avenues the work presented here can be extended

- *Design of risk and cost functions in CRPFs:* In Chapter 4, we propose two novel CRPF schemes which differ primarily in the design of proposal distributions for propagation of particles. Another important aspect of CRPFs is the choice of risk and cost functions. We consider simple cost function which are $\propto \frac{1}{|\epsilon|^\kappa}$ where ϵ is the residual and κ is the exponent factor. Typical choices of κ are 1 or 2. With $\kappa = 2$, the quality of the particles (through the risk functions) is magnified than when $\kappa = 1$. Clearly, this may lead to particle degeneracy with multinomial resampling which is partly resolved by the use of proportional resampling. On the other hand when we are estimating the state, the cost functions which are used also have similar structure. There with $\kappa = 2$, more weight is given to particles with are of high quality. Thus clearly different κ may seem optimal for purposes of estimation. An important step ahead in the study of cost reference particle filtering is the usage of design of optimal risk and cost functions.

- *Sensor self-localization:* In Chapter 5, we proposes iterative least squares (LS) and Bayesian methods (BS), Monte Carlo importance sampling (IS) and cost-based (CS) methods for sensor self-localization and we have observed that by a judicious combination of these methods one can obtain improved performance. Another important problem that was partly addressed was procedures for beacon selection. A more rigorous algorithm for combining the beacon selection procedure should be incorporated in the sensor self-localization algorithm. Also ways of reducing computation by reducing the number of particles to be drawn should be constructed. As in the above problem, when distributions of noise processes are unknown selection of cost functions should be incorporated. Also the proposed algorithms should be extended to scenarios when the noise processes are not zero mean but have an unknown and non-zero bias.

- *Multiple Target Tracking using binary sensors:* In this dissertation, we had considered single target tracking using binary sensors. The open road ahead is multiple target tracking. The problem is quite extensively researched but there are several important issues that have not been completely addressed such as multiple target tracking under unknown number of targets or time varying number of targets. We are currently working on these issues too.

- *Tertiary sensors:* We have discussed sequential monte carlo algorithms for binary wireless sensor networks for tracking a single target 6. Another important class of sensors that is the tertiary sensor network. In a model similar to [75], this problem has been partly studied elsewhere where it was observed that due to the implicit memory based measurement scheme the solution was not straightforward. Following approximation it was also noted that some gains can be obtained in estimating the targets position over networks which employ binary sensors. A more rigorous study of tertiary networks on the lines conducted here should be performed.

- *Fusion of random measures:*

  In chapter 7, we presented SPF and CRPF fusion algorithms for target tracking. When the posterior distribution is unimodal, these random measures are adequately approximated with a single Gaussian which has lower transmission requirements. It was also observed that approximations of random measures with a single Gaussian have slightly larger RMSE than approximations with Gaussian mixtures but at the expense of greater transmission requirements. Some of the open issues not addressed here, on this topic are determining the number of components required for an adequate representation of the sample-based distribution. Clearly, this becomes a model order selection problem with simultaneous estimation of model order and the parameters of the model.

- *Asynchronous Sensor Networks* In the proposed framework in 9, the target trajectory is propagated within a time interval as many instants as there are number of sensors and then the overall likelihood of each path is obtained.

This process is computational intensive particularly when there are a large no. of sensors. Methods in which this process of sampling several times could be circumvented but yet accounting for the sensor time offsets will considerably reduce the computation of the algorithm. Also the algorithm requires a good initialization or informative prior knowledge of the sensor offsets. Since the proposed methodology here is similar to that of density assisted particles it suffers from the disadvantages as the latter does such as when the filter gets stuck at local stationary points when simultaneously estimating the dynamic and static parameters. More efficient methods to allow for the samples to perturb from their stationary points should be designed.

Appendix A

Hybrid Cramér-Rao Bounds for Sensor Self-Localization

Sensor s obtains measurements according to (8.1), These set of measurements can be expressed in vector notation as

$$\mathbf{y}_s = \mathbf{f}(\boldsymbol{\ell}) + \mathbf{v}_s \tag{A.1}$$

with $\mathbf{f}(\boldsymbol{\ell}) = [f^1(\cdot) \cdots f^{N_b}(\cdot)]^\top$ and $\mathbf{v}_s = [v_{s,1} \cdots v_{s,N_b}]$. When $\mathbf{v}_s$ is a Gaussian random vector and the prior density is also Gaussian with covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\ell}}$, the HFIM can be written as [60]

$$\mathbf{J} = \mathbb{E}_{\boldsymbol{\ell}} \left[\left\{ \nabla_{\boldsymbol{\ell}} \mathbf{h}^\top(\boldsymbol{\ell}) \, \boldsymbol{\Sigma}_v^{-1} \, \nabla_{\boldsymbol{\ell}}^\top \mathbf{h}^\top(\boldsymbol{\ell}) \right\} \right] + \mathbf{J}_{bb}. \tag{A.2}$$

For the RSS model, the elements of $\nabla_{\boldsymbol{\ell}} \mathbf{h}^{\top}(\boldsymbol{\ell})$ are calculated as

$$
\begin{aligned}
\frac{\partial h^b(\boldsymbol{\ell})}{\partial l_{s,x}} &= -\frac{10\alpha}{\log 10}\left[\frac{(l_{s,x}-l_{b,x})}{(l_{s,x}-l_{b,x})^2+(l_{s,y}-l_{b,y})^2}\right] \\
\frac{\partial h^b(\boldsymbol{\ell})}{\partial l_{s,y}} &= -\frac{10\alpha}{\log 10}\left[\frac{(l_{s,y}-l_{b,y})}{(l_{s,x}-l_{b,x})^2+(l_{s,y}-l_{b,y})^2}\right] \\
\frac{\partial h^b(\boldsymbol{\ell})}{\partial l_{b,x}} &= -\frac{\partial h^b(\boldsymbol{\ell})}{\partial l_{s,x}} \\
\frac{\partial h^b(\boldsymbol{\ell})}{\partial l_{b,y}} &= -\frac{\partial h^b(\boldsymbol{\ell})}{\partial l_{s,y}} \\
\frac{\partial h^b(\boldsymbol{\ell})}{\partial l_{j,x}} &= \frac{\partial h^b(\boldsymbol{\ell})}{\partial l_{j,y}} = 0 \quad (\forall b \neq j)
\end{aligned}
\tag{A.3}
$$

Appendix B

Posterior Cramer Rao Bounds

Here we present the derivation of the PCRBs of tracking a single target. Recall that the dynamics of the state evolution are given by (6.1), where the prior distribution $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ is singular. Let $\boldsymbol{\xi}_t = [\boldsymbol{v}_t\,\tilde{\mathbf{l}}_t]^\top$, where $\tilde{\mathbf{l}}_t = [x_{1,t}\,x_{2,t}\,\Psi_t]^\top$ and $\boldsymbol{v}_t = [\dot{x}_{1,t}\,\dot{x}_{2,t}]^\top$. In [88], a recursive method for determining the PCRBs for such cases is presented, and here we follow that approach. The state evolution as given by (6.1) and (6.24) can also be expressed in block vector notation as

$$\begin{aligned} \boldsymbol{v}_t &= \boldsymbol{v}_{t-1} + \mathbf{F}\mathbf{u}_t \\ \tilde{\mathbf{l}}_t &= \tilde{\mathbf{l}}_{t-1} + \mathbf{G}\left(\boldsymbol{v}_t + \boldsymbol{v}_{t-1}\right) \end{aligned} \tag{B.1}$$

where

$$\mathbf{F} = \begin{bmatrix} T_s & 0 \\ 0 & T_s \end{bmatrix}, \qquad \mathbf{G} = \begin{bmatrix} \frac{T_s}{2} & 0 \\ 0 & \frac{T_s}{2} \\ 0 & 0 \end{bmatrix}.$$

Let the information submatrix of $\boldsymbol{x}_t$ be denoted by $\mathbf{J}_t$ (which in our case is of size 5×5). The recursive computation of the PCRBs can then be written as follows [88]:

$$\begin{aligned} \mathbf{J}_t &= \begin{bmatrix} \mathbf{J}_t^{11} & \mathbf{J}_t^{12} \\ \mathbf{J}_t^{21} & \mathbf{J}_t^{22} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{S}_t^{22} & \mathbf{S}_t^{23} \\ \mathbf{S}_t^{32} & \mathbf{S}_t^{33} \end{bmatrix} - \begin{bmatrix} \mathbf{S}_t^{21} \\ \mathbf{S}_t^{31} \end{bmatrix} \begin{bmatrix} \mathbf{S}_t^{11} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{S}_t^{12} & \mathbf{S}_t^{13} \end{bmatrix} \end{aligned} \tag{B.2}$$

where the block matrices have sizes 2×2 ($\mathbf{J}_t^{11}$, $\mathbf{S}_t^{11}$, $\mathbf{S}_t^{13}$, $\mathbf{S}_t^{31}$, and $\mathbf{S}_t^{33}$), 2×3 ($\mathbf{J}_t^{12}$, $\mathbf{S}_t^{12}$ and $\mathbf{S}_t^{32}$), 3×2 ($\mathbf{J}_t^{21}$, $\mathbf{S}_t^{21}$ and $\mathbf{S}_t^{23}$), and 3×3 ($\mathbf{J}_t^{22}$ and $\mathbf{S}_t^{22}$), and the block matrices $\mathbf{S}_t^{ij}$ are computed according to

$$\begin{aligned} \mathbf{S}_t &= \begin{bmatrix} \mathbf{S}_t^{11} & \mathbf{S}_t^{12} & \mathbf{S}_t^{13} \\ \mathbf{S}_t^{21} & \mathbf{S}_t^{22} & \mathbf{S}_t^{23} \\ \mathbf{S}_t^{31} & \mathbf{S}_t^{32} & \mathbf{S}_t^{33} \end{bmatrix} \\ &= \mathbf{M}^{-\top} \begin{bmatrix} \mathbf{J}_{t-1}^{11} + \mathbf{H}_{t-1}^{11} & \mathbf{J}_{t-1}^{12} + \mathbf{H}_{t-1}^{12} & \mathbf{H}_{t-1}^{13} \\ (\mathbf{J}_{t-1}^{12} + \mathbf{H}_{t-1}^{12})^\top & \mathbf{J}_{t-1}^{22} + \mathbf{H}_{t-1}^{22} & \mathbf{H}_{t-1}^{23} \\ (\mathbf{H}_{t-1}^{13})^\top & (\mathbf{H}_{t-1}^{23})^\top & \mathbf{H}_{t-1}^{33} \end{bmatrix} \mathbf{M}^{-1}. \end{aligned}$$

The 7×7 matrix $\mathbf{M}$, and the matrices $\mathbf{H}_t^{ij}$ are defined by

$$\mathbf{M} = \begin{bmatrix} \mathbf{I}_{2\times2} & \mathbf{0}_{2\times3} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & \mathbf{0}_{2\times3} & \mathbf{I}_{2\times2} \\ \mathbf{G} & \mathbf{I}_{3\times3} & \mathbf{G} \end{bmatrix}$$

and

$$
\begin{aligned}
\mathbf{H}_t^{11} &= E\left[-\Delta_{\boldsymbol{\upsilon}_t}^{\boldsymbol{\upsilon}_t}\log \bar{p}_t\right], & \mathbf{H}_t^{12} &= E\left[-\Delta_{\boldsymbol{\upsilon}_t}^{\mathbf{l}_t}\log \bar{p}_t\right] \\
\mathbf{H}_t^{13} &= E\left[-\Delta_{\boldsymbol{\upsilon}_t}^{\boldsymbol{\upsilon}_{t+1}}\log \bar{p}_t\right], & \mathbf{H}_t^{22} &= E\left[-\Delta_{\mathbf{l}_t}^{\mathbf{l}_t}\log \bar{p}_t\right] \\
\mathbf{H}_t^{23} &= E\left[-\Delta_{\mathbf{l}_t}^{\boldsymbol{\upsilon}_{t+1}}\log \bar{p}_t\right], & \mathbf{H}_t^{33} &= E\left[-\Delta_{\boldsymbol{\upsilon}_{t+1}}^{\boldsymbol{\upsilon}_{t+1}}\log \bar{p}_t\right]
\end{aligned}
$$

with

$$
\bar{p}_t = p\left(\boldsymbol{\upsilon}_{t+1} \mid \boldsymbol{\upsilon}_t\right) p\left(\mathbf{z}_{t+1} \mid \boldsymbol{\xi}_t, \boldsymbol{\upsilon}_{t+1}\right) \tag{B.3}
$$

and Δ being the Laplacian operator.

The computation of the $\mathbf{H}_t^{ij}$, $i, j = 1, 2, 3$ proceeds as follows:

1. $\mathbf{H}_t^{11}$: We use the following identities:

$$
\begin{aligned}
\mathbf{H}_t^{11} &= E\left[-\Delta_{\boldsymbol{\upsilon}_t}^{\boldsymbol{\upsilon}_t}\log \bar{p}_t\right] = \mathbf{H}_{t,a}^{11} + \mathbf{H}_{t,b}^{11} \\
\mathbf{H}_{t,a}^{11} &= E\left[-\Delta_{\boldsymbol{\upsilon}_t}^{\boldsymbol{\upsilon}_t}\log p\left(\boldsymbol{\upsilon}_{t+1} \mid \boldsymbol{\upsilon}_t\right)\right] \\
\mathbf{H}_{t,b}^{11} &= E\left[-\Delta_{\boldsymbol{\upsilon}_t}^{\boldsymbol{\upsilon}_t}\log p\left(\mathbf{z}_{t+1} \mid \boldsymbol{\upsilon}_t, \mathbf{l}_t, \boldsymbol{\upsilon}_{t+1}\right)\right]
\end{aligned}
$$

   where the expectation is over $\{\boldsymbol{\upsilon}_t, \mathbf{l}_t, \boldsymbol{\upsilon}_{t+1}, \mathbf{z}_{t+1}\}$.

2. $\mathbf{H}_t^{12}$: For computing $\mathbf{H}_t^{12}$, we need the following expressions:

$$
\begin{aligned}
\mathbf{H}_t^{12} &= E\left[-\Delta_{\boldsymbol{\upsilon}_t}^{\mathbf{l}_t}\log \bar{p}_t\right] = \mathbf{H}_{t,a}^{12} + \mathbf{H}_{t,b}^{12} \\
\mathbf{H}_{t,a}^{12} &= E\left[-\Delta_{\boldsymbol{\upsilon}_t}^{\mathbf{l}_t}\log p\left(\boldsymbol{\upsilon}_{t+1} \mid \boldsymbol{\upsilon}_t\right)\right] = \mathbf{0} \\
\mathbf{H}_{t,b}^{12} &= E\left[-\Delta_{\boldsymbol{\upsilon}_t}^{\mathbf{l}_t}\log p\left(\mathbf{z}_{t+1} \mid \boldsymbol{\upsilon}_t, \mathbf{l}_t, \boldsymbol{\upsilon}_{t+1}\right)\right]
\end{aligned}
$$

3. $\mathbf{H}_t^{13}$: We can show that

$$\begin{aligned}\mathbf{H}_t^{13} &= E\left[-\Delta_{\boldsymbol{\upsilon}_t}^{\boldsymbol{\upsilon}_{t+1}} \log \bar{p}_t\right] \\ &= E\left[-\Delta_{\boldsymbol{\upsilon}_t}^{\boldsymbol{\upsilon}_{t+1}} \log p\left(\boldsymbol{\upsilon}_{t+1} \mid \boldsymbol{\upsilon}_t\right)\right] + E\left[-\Delta_{\boldsymbol{\upsilon}_t}^{\boldsymbol{\upsilon}_{t+1}} \log p\left(\mathbf{z}_{t+1} \mid \boldsymbol{\upsilon}_t, \mathbf{l}_t, \boldsymbol{\upsilon}_{t+1}\right)\right].\end{aligned} \tag{B.4}$$

4. $\mathbf{H}_t^{22}$: For computing $\mathbf{H}_t^{22}$, we use

$$\begin{aligned}\mathbf{H}_t^{22} &= E\left[-\Delta_{\mathbf{l}_t}^{\mathbf{l}_t} \log \bar{p}_t\right] \\ &= E\left[-\Delta_{\mathbf{l}_t}^{\mathbf{l}_t} \log p\left(\mathbf{z}_{t+1} \mid \boldsymbol{\upsilon}_t, \mathbf{l}_t, \boldsymbol{\upsilon}_{t+1}\right)\right].\end{aligned}$$

5. $\mathbf{H}_t^{23}$: The process is similar as before. We have

$$\begin{aligned}\mathbf{H}_t^{23} &= E\left[-\Delta_{\mathbf{l}_t}^{\boldsymbol{\upsilon}_{t+1}} \log \bar{p}_t\right] \\ &= E\left[-\Delta_{\mathbf{l}_t}^{\boldsymbol{\upsilon}_{t+1}} \log p\left(\mathbf{z}_{t+1} \mid \boldsymbol{\upsilon}_t, \mathbf{l}_t, \boldsymbol{\upsilon}_{t+1}\right)\right].\end{aligned}$$

6. $\mathbf{H}_t^{33}$: We use the identities

$$\begin{aligned}\mathbf{H}_t^{33} &= E\left[-\Delta_{\boldsymbol{\upsilon}_{t+1}}^{\boldsymbol{\upsilon}_{t+1}} \log \bar{p}_t\right] \\ &= E\left[-\Delta_{\boldsymbol{\upsilon}_{t+1}}^{\boldsymbol{\upsilon}_{t+1}} \log p\left(\boldsymbol{\upsilon}_{t+1} \mid \boldsymbol{\upsilon}_t\right)\right] + E\left[-\Delta_{\boldsymbol{\upsilon}_{t+1}}^{\boldsymbol{\upsilon}_{t+1}} \log p\left(\mathbf{z}_{t+1} \mid \boldsymbol{\upsilon}_t, \mathbf{l}_t, \boldsymbol{\upsilon}_{t+1}\right)\right].\end{aligned}$$

Appendix C

Conditions for the validity of the Gaussian approximations

Let $p(\boldsymbol{\theta})$ be a smooth density with $\boldsymbol{\theta} \in \mathbb{R}^n$. Expanding the logarithm of the density around its mean $\hat{\boldsymbol{\theta}}$ using Taylors series and neglecting the higher order derivative terms (> 2) we have

$$\ln(p(\boldsymbol{\theta})) = -\frac{1}{2}(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}) + \mathbf{g}(\hat{\boldsymbol{\theta}})^\top(\hat{\boldsymbol{\theta}}) + \mathbf{g}(\hat{\boldsymbol{\theta}})^\top \boldsymbol{\Sigma}\mathbf{g}(\hat{\boldsymbol{\theta}}) + \ln(p(\hat{\boldsymbol{\theta}}) + \epsilon(\boldsymbol{\theta})$$

where we express, $\tilde{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}} + \boldsymbol{\Sigma}\mathbf{g}(\hat{\boldsymbol{\theta}})$, $\mathbf{g}(\hat{\boldsymbol{\theta}}) = \frac{\partial \ln(p(\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}} |_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}$ and $\boldsymbol{\Sigma}^{-1} = -\frac{\partial^2 \ln(p_t(\boldsymbol{\theta}))}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top} |_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}$. Thus we have

$$p(\boldsymbol{\theta}) \propto \exp\left[-\frac{1}{2}(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})\right] \propto \mathcal{N}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\Sigma}). \tag{C.1}$$

Similarly we have for a product of N densities

$$p_N(\boldsymbol{\theta}) = \prod_n p_n(\boldsymbol{\theta}) \propto \prod_n \mathcal{N}_n(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}_n, \boldsymbol{\Sigma}_n) \propto \mathcal{N}(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}_f, \boldsymbol{\Sigma}_f) \tag{C.2}$$

$$\begin{aligned}
\boldsymbol{\Sigma}_f^{-1} &= \boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1} \cdots \boldsymbol{\Sigma}_N^{-1} \\
\tilde{\boldsymbol{\theta}}_f &= \boldsymbol{\Sigma}_f(\boldsymbol{\Sigma}_1^{-1}\tilde{\boldsymbol{\theta}}_1 + \cdots \boldsymbol{\Sigma}_N^{-1}\tilde{\boldsymbol{\theta}}_N) \\
\tilde{\boldsymbol{\theta}}_f &= \boldsymbol{\Sigma}(\boldsymbol{\Sigma}_1^{-1}\hat{\boldsymbol{\theta}}_1 + \cdots \boldsymbol{\Sigma}_N^{-1}\hat{\boldsymbol{\theta}}_N) + \boldsymbol{\Sigma}(\mathbf{g}(\hat{\boldsymbol{\theta}}_1) + \mathbf{g}(\hat{\boldsymbol{\theta}}_2) + \cdots \mathbf{g}(\hat{\boldsymbol{\theta}}_N)) \\
\boldsymbol{\Sigma}^{-1}\tilde{\boldsymbol{\theta}}_f &= \boldsymbol{\Sigma}^{-1}\hat{\boldsymbol{\theta}}_f + (\mathbf{g}(\hat{\boldsymbol{\theta}}_1) + \mathbf{g}(\hat{\boldsymbol{\theta}}_2) + \cdots \mathbf{g}(\hat{\boldsymbol{\theta}}_N)) = \boldsymbol{\Sigma}^{-1}\hat{\boldsymbol{\theta}}_f + \Delta\mathbf{g}(\hat{\boldsymbol{\theta}}). \qquad (C.3)
\end{aligned}$$

A ratio of two distributions can be expressed as

$$\frac{p_1(\boldsymbol{\theta})}{p_2(\boldsymbol{\theta})} \propto \mathcal{N}_1(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}_s, \boldsymbol{\Sigma}_s) \tag{C.4}$$

where $\boldsymbol{\Sigma}_s^{-1} = \boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\Sigma}_2^{-1}$ and $\boldsymbol{\Sigma}_s^{-1}\tilde{\boldsymbol{\theta}}_s = \boldsymbol{\Sigma}_1^{-1}\tilde{\boldsymbol{\theta}}_1 - \boldsymbol{\Sigma}_2^{-1}\tilde{\boldsymbol{\theta}}_2$. Thus we have the following form for a ratio of a product of densities using equations (C.2) and (C.4)

$$\frac{\prod_k p_{a,k}(\boldsymbol{\theta})}{\prod_k p_{b,k}(\boldsymbol{\theta})} \propto \frac{\mathcal{N}(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}_a, \boldsymbol{\Sigma}_a)}{\mathcal{N}(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}_b, \boldsymbol{\Sigma}_b)} \propto \mathcal{N}(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}_{ab}, \boldsymbol{\Sigma}_{ab}) \tag{C.5}$$

$$\boldsymbol{\Sigma}_{ab}^{-1} = \boldsymbol{\Sigma}_a^{-1} - \boldsymbol{\Sigma}_b^{-1}$$

$$\begin{aligned}
\boldsymbol{\Sigma}_{ab}^{-1}\tilde{\boldsymbol{\theta}}_{ab} &= \boldsymbol{\Sigma}_a^{-1}\tilde{\boldsymbol{\theta}}_a - \boldsymbol{\Sigma}_b^{-1}\tilde{\boldsymbol{\theta}}_b = \boldsymbol{\Sigma}_a^{-1}\hat{\boldsymbol{\theta}}_a - \boldsymbol{\Sigma}_b^{-1}\hat{\boldsymbol{\theta}}_b + \Delta\mathbf{g}(\hat{\boldsymbol{\theta}}_a) - \Delta\mathbf{g}(\hat{\boldsymbol{\theta}}_b) \\
&= \boldsymbol{\Sigma}_{ab}^{-1}\hat{\boldsymbol{\theta}}_{ab} + \Delta\mathbf{g}(\hat{\boldsymbol{\theta}}_{ab})
\end{aligned}$$

When these distributions are approximated by a Gaussian distribution around the true mean $\hat{\boldsymbol{\theta}}$ of the distribution, instead of $\tilde{\boldsymbol{\theta}}$ it can be shown that the ratio of these products is offset by an exponential product of $\exp[-2\Delta\mathbf{g}(\hat{\boldsymbol{\theta}}_{ab})^{\top}\boldsymbol{\theta}]$.

Bibliography

[1] J. Fraden, *Handbook of Modern Sensors: Physics, Designs, and Applications*, AIP Press, 2nd edition, 1997.

[2] R. R. Brooks and S. S. Iyengar, *Multi-sensor fusion: Fundamentals and applications with software*, Prentice Hall, 1998.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks (Amsterdam, Netherlands: 1999)*, vol. 38, no. 4, pp. 393–422, Mar. 2002.

[4] M. Tubaishat and S. Madria, "Sensor networks : An overview," *IEEE Potentials*, pp. 20–23, April/May 2003.

[5] P. Rentala, R. Musunuri, S. Gandham, and U. Saxena, "Survey of sensor networks," Utdcs-10-03., University of Texas at Dallas,, 2003.

[6] G. J. Pottie and W. J. Kaiser, "Embedding the Internet: Wireless integrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 551–558, May 2000.

[7] "Habitat monitoring on great duck island," http://www.greatduckisland.net.

[8] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA-02)*, New York, Sept. 28 2002, pp. 88–97, ACM Press.

[9] "Sensor web for the study of glaciers," http://envisense.org/glacsweb.htm.

[10] J. Burrell, T. Brooke, and R. Beck, "Vineyard computing: Sensor networks in agricultural production," *IEEE Pervasive Computing*, vol. 3, no. 1, pp. 38–45, January-March 2004.

[11] "Smart bridge," IEEE Spectrum, July 2004.

[12] D. Li and Yu-Hen Hu, "Energy-based collaborative source localization using acoustic microsensor array," *EURASIP Journal on Applied Signal Processing, Special Issue on Sensor Networks*, vol. 2003, no. 4, pp. 321–337, March 2003.

[13] R. Moses, D. Krishnamurthy, and R. Patterson, "A self-localization method for wireless sensor networks," *EURASIP Journal on Applied Signal Processing, Special Issue on Sensor Networks*, vol. 2003, no. 4, pp. 348–358, March 2003.

[14] Nirupama Bulusu, *Self-Configuring Localization Systems*, Ph.D. thesis, University of California at Los Angeles,, October 2002.

[15] X. Sheng and Yu-Hen Hu, "Sequential acoustic energy based source localization using particle filter in a distributed sensor network," IEEE International Conference on Speech, Acoustics, and Signal Processing, May 17-21 2004, pp. 972–975.

[16] X. Sheng, *Collaborative Energy Based Source Localization and Tracking in Wireless Sensor Network System*, Preliminary report, University of Wisconsin-Madison, College of Engineering, 2003.

[17] F. Zhao, J. Shin, and J. Reich, "Information-driven dynamic sensor collaboration," *Signal Processing Magazine, IEEE*, vol. 19, no. 2, pp. 61–72, March 2002.

[18] M. Coates, "Distributed particle filters for sensor networks," in *Proceedings of the third international symposium on Information processing in sensor networks (IPSN-04)*, New York, Apr. 26–27 2004, pp. 99–107, ACM Press.

[19] D. Blatt and A. O. Hero, "Distributed maximum likelihood estimation for sensor networks," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, 2004.

[20] P. K. Varshney, *Distributed Detection and Data Fusion*, Springer-Verlag, 1997.

[21] D. Li, K.D. Wong, Yu-Hen Hu, and A. M.Sayeed, "Detection, classification, and tracking of targets," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 17–29, March 2002.

[22] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*, Springer, New York, 1999.

[23] D. O. Anderson and J. B. Moore, *Optimal Filtering*, Prentice-Hall, Inc., Englewood Cliffs N. J., 1979.

[24] S. J. Julier and J. K. Uhlmann., "A new extension of the Kalman filter to nonlinear systems," in *In Proceedings of AeroSense: The 11th International Symposium on Aerospace and Defence Sensing, Simulation and Controls*, 1997.

[25] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC.*, 1-4 Oct. 2000, pp. 153–158.

[26] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEE Proceedings-F*, vol. 140, no. 2, pp. 107–113, Apr. 1993.

[27] 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*, vol. 50, no. 2, pp. 174–188, Feb 2002.

[28] J. Carpenter, P. Clifford, and P. Fernhead, "An improved particle filter for non-linear problems," *IEEE Proceedings on radar and sonar navigation*, vol. 146, no. 1, pp. 2–7, 1999.

[29] R. Douc and O. Cappe, "Comparision of resampling schemes for particle filtering," in *4th International Symposium on Image and Signal Processing and Analysis (ISPA)*, 2005.

[30] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*, Springer-Verlag, 2001.

[31] Genshiro Kitagawa, "A self-organizing state space model," *Journal of the American Statistical Association*, vol. 93, no. 443, pp. 1203–1215, Sept. 1998.

[32] J. Kotecha and P. M. Djurić, "Gaussian particle filtering," *IEEE Transactions on Signal Processing*, vol. 51, no. 10, pp. 2592–2601, October 2003.

[33] A. Doucet, S. J. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statistics and Computing*, pp. 197–208, 2000.

[34] J. Míguez, M. F. Bugallo, and P. M. Djurić, "A new class of particle filters for random dynamical systems with unknown statistics," *EURASIP Journal on Applied Signal Processing*, vol. 15, pp. 2278–2294, 2004.

[35] Mónica F. Bugallo, Joaquín Míguez, and P M Djurić, "Positioning by cost reference particle filters: Study of various implementations," in *The Proceedings of the International Conference on Computer as a tool (EUROCON)*, Belgrade (Serbia), 2005, EUROCON.

[36] W. J. J. Rey, *Introduction to Robust and Quasi–Robust Statistical Methods*, Springer–Verlag, 1983.

[37] M. West, "Approximating posterior distributions by mixtures," *Journal of the Royal Statistical Society (Ser. B)*, vol. 55, no. 2, pp. 409–422, 1993.

[38] N. Patwari, A. O. Hero-III, M. Perkins, N. S. Correal, and R. J. O'Dea, "Relative location estimation in wireless sensor networks," *IEEE Transactions on Signal Processing, Special Issue on Signal Processing in Networks*, vol. 51, no. 8, pp. 2137–2148, Nov 2003.

[39] P. M. Djurić, M. Vemula, and M. F. Bugallo, "Signal processing by particle filtering for binary sensor networks," in *IEEE Digital Signal Processing Workshop*, 2004.

[40] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low cost outdoor localization for very small devices," *IEEE Personal Communications Magazine*, vol. 7, no. 5, pp. 28–34, October 2000.

[41] B. M. Sadler, R. J. Kozick, and L. Tong, "Multimodal sensor localization using a mobile access point," in *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP-05)*, Philadelphia, Mar. 18–23 2005, pp. 753–756.

[42] D. Niculescu and B. Nath, "Ad hoc positioning system (aps)," *Proceedings of GLOBECOM San Antonio*, 2001.

[43] A. Savvides, H. Park, and M. B. Srivastava, "The bits and flops of the n-hop multilateration primitive for node localization problems," in *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA-02)*, New York, Sept. 28 2002, pp. 112–121, ACM Press.

[44] K. Langendoen and N. Reijers, "Distributed localization in wireless sensor networks: a quantitative comparison," *Computer Networks*, vol. 43, no. 4, pp. 499–518, 2003.

[45] M. L. Sichitiu and V. Ramadurai, "Localization of wireless sensor networks with a mobile beacon," in *Proceedings of the First IEEE Conference on Mobile Ad-hoc and Sensor Systems (MASS 2004)*, (Fort Lauderdale, FL), Oct. 2004.

[46] A. Galstyan, B. Krishnamachari, K. Lerman, and S. Pattem, "Distributed online localization in sensor networks using a moving target," in *Proceedings of the*

Third International Symposium on Information Processing in Sensor Networks (IPSN-04), New York, Apr. 26–27 2004, pp. 61–70, ACM Press.

[47] P. M. Djurić, M. Vemula, M. F. Bugallo, and J. Miguez, "Non-cooperative localization of binary sensors," in *IEEE Workshop on Statistical Signal Processing (SSP-05)*, Jul 17-20 2005.

[48] Pubudu N. Pathirana, Nirupama Bulusu, Andrey V. Savkin, and Sanjay K. Jha, "Node localization using mobile robots in delay-tolerant sensor networks," *IEEE Trans. Mob. Comput*, vol. 4, no. 3, pp. 285–296, 2005.

[49] N. Patwari, J. N. Ash, S. Kyperountas, Alfred O. Hero III, Randolph L. Moses, and Neiyer S. Correal, "Locating the nodes: cooperative localization in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 54–69, July 2005.

[50] Andreas Savvides, L. Girod, Mani B. Srivastava, and Deborah Estrin, *Localization in Sensor Networks*, Wireless Sensor Networks. Kluwer, Norwell MA, 2004.

[51] D. Fox, J. Hightower, H. Kauz, L. Liao, and D. Patterson, "Bayesian techniques for location estimation," in *In Proc. Workshop on Location-aware Computing, UBICOMP Conf., Seattle, WA, October 2003.*, 2003.

[52] W.H. Foy, "Position-location solutions by Taylor-series estimation," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. AES-12, no. 2, pp. 187–194, March 1976.

[53] Andreas Savvides, Chih-Chieh Han, and Mani B. Srivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *MOBICOM*, 2001, pp. 166–179.

[54] V. Ramadurai and M.l L. Sichitiu, "Localization in wireless sensor networks: A probabilistic approach.," in *International Conference on Wireless Networks*, 2003, pp. 275–281.

[55] A. T. Ihler, J. W. Fisher, R. L. Moses, and A. S. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 809–819, 2005.

[56] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer-Verlag, NY, 1999.

[57] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, Prentice Hall PTR, Upper Sadle River, NJ 07458, 1993.

[58] A. H. Sayed and T. Kailath, "A state-space approach to adaptive RLS filtering," *Signal Processing Magazine, IEEE*, vol. 11, no. 3, pp. 18–60, July 1994.

[59] J. Liu and M. West, "Combined parameter and state estimation in simulation-based filtering," in *Sequential Monte Carlo Methods in Practice. New York*, J. F. G. De Freitas A. Doucet and N. J. Gordon, Eds. 2000, Springer-Verlag, New York.

[60] H. L. Van Trees, *Detection, Estimation, and Modulation Theory*, John Wiley & Sons, 1968.

[61] Y. Rockah and P. Schultheiss, "Array shape calibration using sources in unknown locations–part i: Far-field sources," *IEEE Transactions on Acoustics, Speech, and Signal Processing* ,, vol. 35, no. 3, pp. 286–299, Mar 1987.

[62] H. Messer, "The hybrid Cramer-Rao Lower Bound - From practice to theory," in *Sensor Array and Multichannel Processing*, July 12-14, 2006, pp. 304–307.

[63] H. Wang, K. Yao, G. Pottie, and D. Estrin, "Entropy-based sensor selection heuristic for target localization," in *Proceedings of the third international symposium on Information processing in sensor networks (IPSN-04)*, New York, Apr. 2004, pp. 36–45, ACM Press.

[64] L. M. Kaplan, "Local node selection for localization in a distributed sensor network," *IEEE Transactions on Aerospace and Electronic Systems,*, vol. 42, no. 1, pp. 136–146, Jan. 2006.

[65] B. Chen, W. B. Heinzelman, M. Liu, and A. T. Campbell, "Editorial of Special issue on wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2005, no. 4, pp. 459–461, 2005.

[66] C.-Y. Chong and S. P. Kumar, "Sensor networks: Evolution opportunities and challenges," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247–1256, 2003.

[67] S. Kumar, F. Zhao, and D. Shepherd, "Collaborative signal and information processing in microsensor networks," *IEEE Signal Processing Magazine*, pp. 13–14, March 2002.

[68] F. Zhao and L. Guibas, *Wireless Sensor Networks*, New York: Morgan Kaufman Publishers, 2004.

[69] A. Arora and et al, "A line in the sand: A wireless sensor network for target detection, classification, and tracking," *Computer Networks*, pp. 605–634, 2004.

[70] M. Pitt and N. Shepard, "Filtering via simulation: Auxiliary particle filters," *Journal of the American Statistical Association*, vol. 94, no. 446, pp. 590–599, June 1999.

[71] D. Crisan and A. Doucet, "A survey of convergence results on particle filtering," *IEEE Transactions Signal Processing*, vol. 50, no. 3, pp. 736–746, March 2002.

[72] P. Del Moral, *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*, Springer-Verlag New York Inc., 2004.

[73] J. Míguez, M. F. Bugallo, and P. M. Djurić, "Erratum on 'a new class of particle filters for random dynamical systems with unknown statistics," *EURASIP Journal on Applied Signal Processing*, 2006, Article ID 78708.

[74] K. Mechitov, S. Sundresh, Y. Kwon, and G. Agha, "Cooperative tracking with binary-detection sensor networks," in *1st ACM International Conference on Embedded Networked Sensor Systems*, 2003, pp. 332–333.

[75] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus, "Tracking a moving object with a binary sensor network," in *the Proceeings of the First International Conference on Embedded Networked Sensor Systems*, Los Angeles, CA, 2003, pp. 150–161.

[76] W. Kim, K. Mechitov, J.-Y. Choi, and S. Ham, "On target tracking with binary proximity sensors," in *The Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks, (IPSN)*, 2005, pp. 301–308.

[77] S. Oh and S. Sastry, "Tracking on a graph," in *the Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks (IPSN)*, Los Angeles, CA, 2005, pp. 195–202.

[78] R. Niu and P. Varshney, "Target location estimation in wireless sensor networks using binary data," in *38th Annual Conference on Information Sciences and Systems*, Princeton, NJ, 2004.

[79] L. Y. Wang, J.-F. Zhang, and G. G. Yin, "System identification using binary sensors," *IEEE Transactions on Automatic Control*, vol. 48, pp. 1892–1907, 2003.

[80] J.-F. Chamberland and V. V. Veeravali, "Decentralized detection in sensor networks," *IEEE Transactions on Signal Processing*, pp. 407–416, 2003.

[81] F. Gustaffson, F. Gunnarsson, N. Bergman, U. Forssel, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filtering for positioning, navigation, and tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 425–437, 2002.

[82] N. Patwari, A. O. Hero, M. Perkins, N. S. Correal, and R. J. O'Dea, "Relative location estimation in wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 51, no. 5, pp. 2137–2148, 2003.

[83] X. Sheng and Y-H. Hu, "Maximum likelihood multiple-source localization using acoustic energy measurements with wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 53, pp. 44–53, 2005.

[84] N. Chopin, "A sequential particle filter for static models," *Biometrika*, vol. 89, no. 3, pp. 539–552, 2002.

[85] P.M.Djurić, M.F.Bugallo, and J. Míguez, "Density assisted particle filters for state and parameter estimation," *Proceedings of the IEEE 29th International Conference on Acoustics, Speech and Signal Processing*, May 2004.

[86] G. Storvik, "Particle filters for state-space models with the presence of unknown static parameters," *IEEE Transactions Signal Processing*, vol. 50, no. 2, pp. 281–289, February 2002.

[87] J. Kotecha and P. M. Djurić, "Gaussian particle filtering," *IEEE Transactions on Signal Processing*, vol. 51, no. 10, pp. 2592–2601, 2003.

[88] P. Tichavský, C. H. Muravchik, and A. Nehorai, "Posterior Cramér-Rao bounds for discrete-time nonlinear filtering," *IEEE Transactions on Signal Processing*, vol. 46, pp. 1386–1396, 1998.

[89] P K Varshney, "Multisensor data fusion," *Electronics and Communication Engineering Journal*, vol. 9, no. 6, pp. 245–253, Dec. 1997.

[90] R. C. Luo, C. C. Yih, and K. L. Su, "Multisensor fusion integration: Approaches applications, and future research direction," *IEEE Sensors Journal*, vol. 2, no. 2, pp. 107–119, 2002.

[91] M. Rosencrantz, G. J. Gordon, and S. Thrun, "Decentralized sensor fusion with distributed particle filters," in *UAI '03, Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence, August 7-10 2003, Acapulco, Mexico*, 2003, pp. 493–500.

[92] S. Challa, M. Palaniswami, and A. Shilton, "Distributed data fusion using support vector machines," in *Proceedings of the Fifth International Conference on Information FUSION 2002*, 2002, vol. 2, pp. 881– 885.

[93] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association*, Academic Press, New York, 1988.

[94] G. J. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*, John Wiley and Sons, 1997.

[95] M. Yarvis and Wei Ye, "Tiered architectures in sensor networks," in *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*, M. Ilyas and I. Mahgoub, Eds., chapter 13. CRC Press LLC., 2004.

[96] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Hawaii International Conference on System Sciences*, 2000.

[97] W. Chen, J. C. Hou, and L. Sha, "Dynamic clustering for acoustic target tracking in wireless sensor networks," in *International Conference on Network Protocols*, 2003, pp. 284–294.

[98] X. Sheng and Y. H. Hu, "Energy based acoustic source localization," in *Information Processing in Sensor Networks*, 2003, pp. 285–300.

[99] M. Vemula and P. M. Djurić, "Multisensor fusion for target tracking using sequential Monte Carlo methods," in *IEEE Workshop on Statistical Signal Processing*, 2005.

[100] F. Sivrikaya and B. Yener, “Time synchronization in sensor networks,” *IEEE Network*, vol. 18, no. 4, pp. 45– 50, 2004.

[101] L. Doherty, B. A. Warneke, B. Boser, and K. S. J. Pister, “Energy and performance considerations for smart dust,” in *International Journal of Parallel and Distributed Sensor Networks*, Dec 2001.

[102] X. Lin, Y. Bar-Shalom, and T. Kirubarajan, “Multisensor-multitarget bias estimation for general asynchronous sensors,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 3, pp. 899– 921, 2005.

[103] Yifeng Zhou, “A kalman filter based registration approach for asynchronous sensors in multiple sensor fusion applications,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP ’04)*, may 2004, vol. 2, pp. 293–6.

[104] T. Li, A. Ekpenyong, and Y.F. Huang, “A location system using asynchronous distributed sensors,” in *INFOCOM*, 2004.

[105] A. Doucet and V. B. Tadic, “Parameter estimation in general state-space models using particle methods,” *Annals of the Institute of Statistical Mathematics*, vol. 55, no. 2, pp. 409–422, June 2003.

[106] G. Hendeby and R. Karlsson, “Target tracking performance evaluation — A general software environment for filtering,” in *IEEE Aerospace Conference, Big Sky, MT, USA, Proceedings*, 2007.

[107] G. L. Gilardoni, “Accuracy of posterior approximations via χ^2 and harmonic divergences,” *Journal of Statistical Planning and Inference*, vol. 128, no. 1, pp. 475–487, 2005.