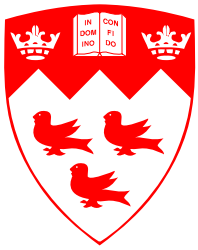
Distributed Particle Filters for Object Tracking in Sensor Networks

Garrick Ing



Department of Electrical & Computer Engineering
McGill University, Montreal,
Quebec, Canada

December 2005

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

Abstract

A particle filter (PF) is a simulation-based algorithm used to solve estimation problems, such as object tracking. The PF works by maintaining a set of "particles" as candidate state descriptions of an object's position. The filter determines how well the set of particles describe the observations and fit the dynamic model, in order to form an object state estimate. The drawback of the basic PF is that the algorithm functions by collecting all data at a fusion centre. This leads to high communication and energy costs in a resource-limited network such as the sensor network. In this thesis, we analyze the PF to determine how it can be modified for efficient use in a sensor network. Our main priority is to keep communication and energy costs low since this increases the network lifetime. We propose two innovative particle filtering algorithms which minimizes the associated costs.

Sommaire

Un filtre de particules (FP) est un algorithme utilisé pour résoudre des problèmes d'estimation, comme le pistage d'un objet. Le FP maintient un ensemble de $\ll$ particules $\gg$ qui représentent chacune un état potentiel de l'objet. Le filtre détermine le niveau d'exactitude de l'ensemble de particules, en se basant sur des observations et le modèle décrivant les dynamiques de l'objet, pour ensuite estimer l'état le plus probable de l'objet. Le désavantage du FP est qu'il requiert que la collecte des observations se fasse à un point central ce qui entraîne une haute consommation d'énergie. Dans ce mémoire, nous déterminons les modifications qu'on doit apporter au FP pour l'utiliser dans un réseau où les ressources sont limitées en gardant les cohatuts de transmissions et la consommation d'énergie au minimum pour prolonger la longévité du réseau. Nous proposons deux nouveaux algorithmes pour le FP qui minimisent ces deux coûts.

Acknowledgments

I would like to thank my supervisor, Professor Mark Coates for his insight, thoughts and help during the course of the experiments and the writing of the thesis. Without his dedications, this thesis would not be possible. I would also like to thank Richard Thommes for taking the time review the thesis to ensure it is grammatically correct. In addition, I would like to thank Frederic Thouin for his help in providing a French language translation of the abstract. I would like to also thank my parents Betty and Jik for their support during the course of my studies. Finally, I would like to thank Amy for her support and patience with me during the course of education.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With every passing day, scientists and engineers continually create new technology to improve the functionality of current devices. Since the invention of the silicon chip in 1961, technological development has advanced at an alarming rate. Silicon chips provided the means for the development of modern computers. Computers, once the size of a warehouse, can now be made to fit in a briefcase or backpack. Better yet, technological advancements have now allowed for miniature devices (or limited functional PC), such as PDAs or cell phones, to be possible. The technological advances seem endless. In 2003, Kris Pister and his research group at the University of California at Berkeley engineered the smart dust [1–3], a tiny microelectromechanical sensor (MEMS) of only a few millimetres in size, capable of detecting many world attributes, such as light intensity and sound.

These "nodes" allow for the possibility of placing devices discretely in any place, out of detection and out of intrusion, to be used for many applications once deemed impossible. For instance, the military can use this technology to detect or track threats, such as enemy vehicles or toxins in the air, in a foreign or unknown terrain by simply "dropping" them in the area using a plane. A business owner can also use the technology to simply manage inventory in his store. A homeowner is able use these devices to detect bacterial levels of the water in his swimming pool.

If a collection of these nodes is used together to perform a specific goal, we can term this structure as a sensor network. A sensor network is thus a collection of tiny devices with the ability to sense certain elements of the

real world [4]. These devices are able to detect, control, or monitor some aspect of an environment. The nodes also have the ability to interact with one another using radio frequency communications. One of the main advantages of sensor networks is their size. Because the nodes are so small, they can be placed virtually anywhere without interfering with their environment. With an onboard sensor, processor, and communication device, they are capable of running a computer network in any place they are needed.

One of the important uses of a sensor network is to track the state of an object. Recently, there has been a lot of attention given to this topic in the research community. Tracking using a sensor network can be useful for many purposes such as tracking an enemy vehicle in a war-zone or ensuring certain personnel do not access restricted areas of a building.

While a sensor network can provide ample opportunity to use technology in ways it has never been used before, there are still many challenges that lie in the way before its full potential can be realized. Some of the main focus of sensor networks research today is in the area of energy consumption and communications costs. Sensor network devices are often battery powered. Unfortunately in many cases, it is difficult to replace the power source after deployment because they are distributed in places which are difficult to reach or in hostile territory. Another difficulty with sensor networks is that many of the algorithms currently used for tracking are communication-intensive. Many algorithms employ a fusion centre to collect and to process data obtained within the network. An algorithm using this type of configuration, called a centralized algorithm, not only drains the energy of the fusion centre due to its constant processing of data, but the neighbouring nodes of this centre will also have to consistently route data to it. Energy consumption will therefore be unbalanced in the network, being mostly distributed near the fusion centre. This means that the energy in the subset of nodes are quickly drained due a heavy workload, making for a shorter lifetime of the network.

Fortunately, the inherent problems of centralized algorithms (high communication cost, and unbalanced, high energy consumption) can be rectified by applying a distributed structure. In distributed algorithms, the goal is to avoid using only one device to perform most or all computational tasks so that the energy problem does not occur.

A popular algorithm used in tracking scenarios is the particle filter. The

particle filter is a simulation based algorithm often used for estimation problems. It is a very powerful procedure since it is able to model attributes of non-Gaussianity and non-linearity, which is found to describe many real world elements. Many tracking or estimation problems are based on the particle filter since it is simple and effective.

1.1 Thesis Contribution

The research into distributed tracking algorithms, with an emphasis on sensor networks and the goal of object tracking, has been very limited. Great attention has not been given to analyzing how the particle filter can be advantageously used in a distributed manner. For example, identifying the most efficient type of information that should be exchanged between nodes in order to maintain the tracking algorithm and determining efficient collaborative signal processing methods to process distributed data to achieve the tracking goal has been only touched briefly in a few papers.

With the advantages of the particle filter, the focus of this thesis is to explore and propose efficient and effective distributed sensor network algorithms based on this filter. In particular, with the limitations of sensor networks, we wish to create distributed algorithms focused on reducing communication and energy consumption costs in order to increase the network lifetime.

The thesis provides an analysis of the particle filter algorithm. We will determine which steps of the process pose a challenge in sensor networks. After this discussion, we propose two different distributed algorithms, each one being established for a different networking scenario. These proposals are based on our analysis of the particle filter in order to maximize their network lifetime. We examine the effectiveness of these algorithms by running simulations in Matlab to determine their tracking accuracy, as well as their communication and computational costs in order to evaluate their energy consumption.

1.2 Thesis Organization

This thesis is divided into 6 chapters. Immediately after the current introduction chapter, Chapter 2 gives a background into tracking algorithms and sensor networks in order to identify the need for a distributed particle filtering

algorithm in sensor networks. Chapter 3 describes the challenges faced with using particle filters in sensor networks. Chapter 4 outlines our first proposed algorithm based on the concept of a parallel distributed particle filter. Chapter 5 then looks at the second proposed algorithm based on locally distributing the particle filter. Finally, Chapter 6 provides concluding remarks to this thesis as well as future avenues of research in this area.

1.3 Published Work

Some of the content presented in this thesis was published in two conferences. These published works discuss some of the preliminary results of the algorithms presented in this thesis.

- G. Ing and M. J. Coates, Parallel Particle Filters for Tracking in Wireless Sensor Networks, in Proc. IEEE Workshop on Signal Processing Advances in Wireless Communications, New York, NY, June 2005.
- M. J. Coates and G. Ing, Sensor Network Particle Filters: Motes as Particles, in Proc. IEEE Workshop on Statistical Signal Processing, Bordeaux, France, July 2005.

Chapter 2

Background

This chapter provides the background necessary to understand the tracking problem in sensor networks. We explore various tracking algorithms and discuss the advantages and disadvantages of each. The particle filter, a popular tracking algorithm, will be shown to be quite beneficial in a sensor network context and a detailed description of the algorithm will be given. Following the discussion on algorithms, we describe what a sensor network is and how it can be beneficial to our society. As research in sensor networks has gained enormous attention recently, we describe sensor network research areas that are being focused on by scientists and engineers today. In this thesis, our interest lies in the collaborative signal processing aspects of sensor networks. Collaborative signal processing is a term used to describe the processing of all sensor data (whether together or separately) in order to achieve its goal, whether it is detection, tracking or some other attribute. We are particularly focused on methods to process sensor measurements taken throughout the network in an efficient manner in order to track a moving object. Finally, this chapter concludes with a literature review on tracking in sensor networks, with a focus on the collaborative signal processing aspects.

2.1 Tracking Algorithms

Many mathematicians look at the tracking problem as an estimation problem. Using available data, an algorithm or set of instructions can be used to estimate the most likely position of an object. In this section, we describe the most popular tracking algorithms.

2.1.1 Problem Statement: Bayesian Tracking

The main focus of this thesis is on a Bayesian tracking scenario. That is, Bayesian models are used. Recall that Bayes' theorem states that the posterior distribution of some signal B given some signal A is equal to the prior probability of B times the likelihood of B given A, divided by a normalizing constant:

$$p(B|A) = \frac{p(A|B)p(B)}{p(A)}. \tag{2.1}$$

Our tracking (estimation) problem focuses on computing the posterior distribution using a likelihood and a prior.

Let us denote $\mathbf{x}_t$ and $\mathbf{y}_t$ as the unobserved system state (e.g. the object's position) and observed measurement signal (e.g. the measured object position), respectively, at time t. We assume the state and signal are both modelled using Markovian, non-linear and non-Gaussian state-space models (although linear and Gaussian systems can be applied). The unobserved state at any given moment in time can be described by

$$\mathbf{x}_t = \mathbf{g}_t(\mathbf{x}_{t-1}, \omega_{t-1}) \tag{2.2}$$

where $\mathbf{g}_t$ is the (possibly) nonlinear system dynamic function and ω_t is the process noise at time t. The observed signal can be described as:

$$\mathbf{y}_t = \mathbf{h}_t(\mathbf{x}_t, \nu_t) \tag{2.3}$$

where $\mathbf{h}_t$ is the (possibly) nonlinear function relating the system state to the measurement and ν_t is the measurement error at time t.

Denote $\mathbf{x}_{0:t} \triangleq \{\mathbf{x}_0, ..., \mathbf{x}_t\}$ and $\mathbf{y}_{1:t} \triangleq \{\mathbf{y}_1, ..., \mathbf{y}_t\}$ as the collection of system states from time 0 to time t and observations from time 1 to time t, respectively. The problem is then to estimate the posterior density $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ of the entire state trajectory conditioned on all the measurements, its marginal distribution (or filtering distribution) $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ and the expectation

$$I(f_t) = \mathbb{E}_{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}[f_t(\mathbf{x}_{0:t})] \triangleq \int f_t(\mathbf{x}_{0:t})p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t} \tag{2.4}$$

where f_t is some function of interest, integrable with respect to some $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. One example function of interest is the conditional mean, whereby $f_t(\mathbf{x}_{0:t}) =$

$\mathbf{x}_{0:t}$.

In our tracking problem, the interest lies primarily in estimating at time t the state $\mathbf{x}_t$ conditioned on the measurements $\mathbf{y}_{1:t}$. This can be achieved by estimating the filtering distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. This density function can be obtained through a recursive computation of the posterior distribution at each time step.

The estimation is performed in two phases. The first phase is the prediction or propagation phase, whereby the current state can be estimated by using the density function $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$. This predictive density can be obtained using the previous posterior density and a transitional prior $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ which is specified by using the system dynamic model of (2.2), as follows:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})d\mathbf{x}_{t-1} \tag{2.5}$$

since

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_t, \mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})d\mathbf{x}_{t-1}. \tag{2.6}$$

The second phase is the update or measurement phase, whereby measurement information is gathered to obtain the posterior density function $p(\mathbf{x}_t|\mathbf{y}_{1:t})$.

Assuming that the measurement $\mathbf{y}_t$ is conditionally independent of the earlier measurements $\mathbf{y}_{1:t-1}$, we just need the probability of the measurement $\mathbf{y}_t$ given $\mathbf{x}_t$ along with the measurement model specified as a likelihood $p(\mathbf{y}_t|\mathbf{x}_t)$. Therefore, the posterior density can then be determined by using Bayes' theorem:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})} \tag{2.7}$$

where the normalization factor is

$$p(\mathbf{y}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1})d\mathbf{x}_t. \tag{2.8}$$

After this phase, the prediction and update processes are repeated in a recursive fashion. This filtering scheme, known as a recursive Bayesian filter, is graphically shown in Figure 2.1.

The only problem with this Bayesian solution is that due to the recursive propagation of the posterior density, analytically determining the solution is

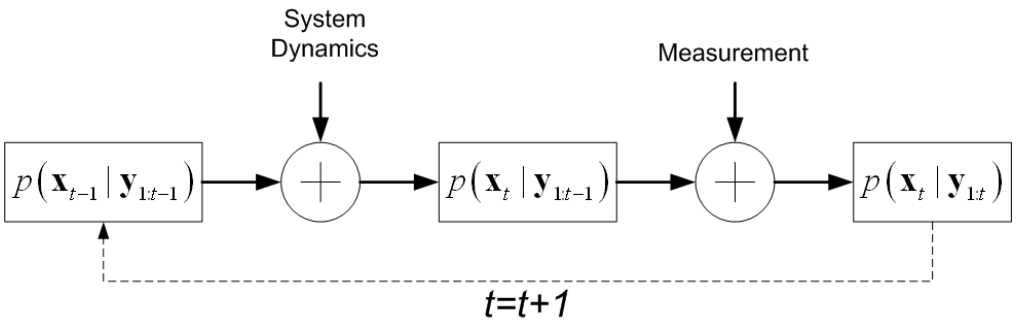


Fig. 2.1 Bayesian filter.

very difficult. This is because many of the posterior densities cannot be exactly formulated using mathematical expressions.

2.1.2 Potential Tracking Algorithms

A popular method for computing the posterior is through the Kalman Filter [5]. The Kalman filter allows for the computation of an exact expression of the sequence of posterior distributions. The drawback is that the data must be modelled by using a linear Gaussian state-space model. However, the advantage is that the posterior density at every time step is assumed to be Gaussian, and therefore it can be parameterized by a mean and covariance.

For example, if $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t})$ is Gaussian, it can be shown that $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ is also Gaussian, provided that the following assumptions are true [6]:

- ω_{t-1} and ν_t are Gaussian
- $\mathbf{g}_t(\mathbf{x}_{t-1}, \omega_{t-1})$ is a linear function of $\mathbf{x}_{t-1}$ and ω_{t-1}
- $\mathbf{h}_t(\mathbf{x}_t, \nu_t)$ is a linear function of $\mathbf{x}_t$ and ν_t

As a result, equations (2.2) and (2.3) can be rewritten as:

$$\mathbf{x}_t = F_t\mathbf{x}_{t-1} + \omega_{t-1} \tag{2.9}$$

$$\mathbf{y}_t = H_t\mathbf{x}_t + \nu_t \tag{2.10}$$

where F_t and H_t are matrices defining linear functions.

From this, the Kalman filter algorithm can be constructed using equations (2.5) and (2.7):

$$p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{x}_{t-1}; m_{t-1|t-1}; P_{t-1|t-1}) \tag{2.11}$$

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{x}_t; m_{t|t-1}; P_{t|t-1}) \tag{2.12}$$

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \mathcal{N}(\mathbf{x}_t; m_{t|t}; P_{t|t}) \tag{2.13}$$

where

$$m_{t|t-1} = F_t m_{t-1|t-1} \tag{2.14}$$

$$P_{t|t-1} = Q_{t-1} + F_t P_{t-1|t-1} F_t^T \tag{2.15}$$

$$m_{t|t} = m_{t|t-1} + K_t(\mathbf{y}_t - H_t m_{t|t-1}) \tag{2.16}$$

$$P_{t|t} = P_{t|t-1} - K_t H_t P_{t|t-1} \tag{2.17}$$

and where $\mathcal{N}(x; m, P)$ is Gaussian with argument x, mean m and covariance P, Q_{t-1} is the covariance of ω_{t-1}, and

$$S_t = H_t P_{t|t-1} H_t^T + R_t \tag{2.18}$$

$$K_t = P_{t|t-1} H_t^T S_t^{-1} \tag{2.19}$$

are the covariance of the innovation term $y_t - H_t m_{t|t-1}$, and the Kalman gain, respectively with R_t as the covariance of ν_t. It should be noted that ω_{t-1} and ν_t have zero mean and are statistically independent.

The solution provided by this Kalman algorithm is an optimal solution to the tracking problem. However, there are many assumptions that must be true in order for this algorithm to be useful. For instance, the Kalman filter requires that linear and Gaussian state-space models be used. Unfortunately, many of problems dealing with the real world phenomena cannot be modelled as such.

As a result, an approximation to the Kalman filter can be used to address these phenomena. This is called the Extended Kalman Filter (EKF) [7]. The EKF works by considering a linear Taylor series approximation of the system function and also an approximation for the observation function. Consequently, equations (2.9) and (2.10) can no longer be used since these functions

are not linear. As a result, we can perform linearization by approximating $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ with a Gaussian

$$p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) \approx \mathcal{N}(\mathbf{x}_{t-1}; m_{t-1|t-1}; P_{t-1|t-1}) \tag{2.20}$$

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) \approx \mathcal{N}(\mathbf{x}_t; m_{t|t-1}; P_{t|t-1}) \tag{2.21}$$

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \approx \mathcal{N}(\mathbf{x}_t; m_{t|t}; P_{t|t}) \tag{2.22}$$

where

$$m_{t|t-1} = f_t'(m_{t-1|t-1}) \tag{2.23}$$

$$P_{t|t-1} = Q_{t-1} + \hat{F}_t P_{t-1|t-1} \hat{F}_t^T \tag{2.24}$$

$$m_{t|t} = m_{t|t-1} + K_t(\mathbf{y}_t - h_t'(m_{t|t-1})) \tag{2.25}$$

$$P_{t|t} = P_{t|t-1} - K_t \hat{H}_t P_{t|t-1}. \tag{2.26}$$

$f_t'(\cdot)$ and $h_t'(\cdot)$ are nonlinear functions, and $\hat{F}_t$ and $\hat{H}_t$ are local linearization of these nonlinear functions which can be represented by:

$$\hat{F}_t = \left.\frac{df_t'(x)}{dx}\right|_{x=m_{t-1|t-1}} \tag{2.27}$$

$$\hat{H}_t = \left.\frac{dh_t'(x)}{dx}\right|_{x=m_{t|t-1}} \tag{2.28}$$

$$S_t = \hat{H}_t P_{t|t-1} \hat{H}_t^T + R_t \tag{2.29}$$

$$K_t = P_{t|t-1} \hat{H}_t^T S_t^{-1}. \tag{2.30}$$

The EKF uses the first term in the Taylor series expansion to linearize the nonlinear function. It is also possible to use the EKF with more terms in the Taylor series expansion, but this introduces additional complexity.

Unfortunately, while the EKF eliminates the linearization restriction associated with the standard Kalman filter, the EKF performs poorly with large bias measurement noise [8], has divergence issues [9], and lacks robustness [10]. Other alternative algorithms for this tracking problem are grid-based [11, 12] and Gaussian-sum filters [13] but they also have their limitations.

Sequential Monte Carlo (SMC) methods, such as the particle filter, provide an excellent alternative. These are numerical methods based on simulation and

are convenient to use now that computation power is easily available. There are no restrictions on the type of system and observation functions, and the noise does not have to be restricted to the Gaussian-type either. The only drawback to SMC methods is that they are generally more computationally demanding [14].

2.1.3 Monte Carlo Methods

A Monte Carlo method is an analytical technique for solving a mathematical or physical problem using random samples (or pseudo-random samples) [15]. Often, exact closed-form or deterministic solutions cannot be obtained due to the complexity of the problem. Monte Carlo methods are used to determine the most likely state of an element. They involve the execution of a large number of simulation trials using random samples of this element. The distribution of the results obtained from these trials is then used to determine the most likely state of the element. The advantage of Monte Carlo methods is that they are able to model non-linear and non-Gaussian state space models, which was not possible with the Kalman filter.

Sequential Monte Carlo Methods

The problem with Monte Carlo methods is that every time new information is obtained, the algorithm must recompute everything from scratch. This means that all data collected since beginning of "time" must be processed again.

In many real-time signal processing applications, data is observed periodically. Thus, an algorithm with data processed sequentially is less computationally demanding since only the new data has to be considered. This is also termed as *on-line* computation. An added benefit to online computation is that there is no need to save an enormous amount of data (i.e. starting with the beginning of "time"). This provides substantial savings in terms of memory cost. As a result, a methodology which can process data sequentially can be very beneficial.

Consequently, Sequential Monte Carlo methods are a convenient approach to computing posterior distributions since they combine the benefit of a Monte Carlo method with sequential computation. A classical use of a Sequential Monte Carlo method is to track a state process. Blake and Isard were among

the first to develop an algorithm using a SMC method to track human movement using an algorithm they termed Condensation [16]. Using this algorithm, the authors were able to track the movements of a dancing girl and a moving hand. Sequential Monte Carlo methods have also been used to track an object [17], to perform localization [18] and robot navigation [19], and for many others tasks.

The particle filter is a popular Sequential Monte Carlo method. It is a powerful method for predicting the state of an element in the environment based on past and current observations. The particle filter maintains a set of "particles" (candidate state descriptions). The algorithm determines how well these particles describe the observations and fit the dynamic model to create state estimates. Algorithms based on SMC methods also come under many other names, such as bootstrap filters [9], condensation [16], and survival of the fittest [20].

We now develop the mathematics behind the particle filter, based on [21].

Perfect Monte Carlo Sampling

In the simplest case, we can represent a posterior distribution using a set of samples or particles. This can be achieved by taking N independent and identically distributed (i.i.d.) random samples $\mathbf{x}_{0:t}^{(t)}$. These samples are drawn from the probability $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. As a result, we can obtain an empirical estimate of this distribution by:

$$\begin{aligned} p(d\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) &= \frac{1}{N}\sum_{i=1}^{N}\delta(\mathbf{x}_{0:t}-\mathbf{x}_{0:t}^{(i)}) \\ &= \frac{1}{N}\sum_{i=1}^{N}\delta_{\mathbf{x}_{0:t}^{(i)}}(d\mathbf{x}_{0:t}) \end{aligned} \tag{2.31}$$

where $\delta_{\mathbf{x}_{0:t}^{(i)}}(d\mathbf{x}_{0:t})$ denotes the delta-Dirac mass located at $\mathbf{x}_{0:t}^{(i)}$.

It is thus easy to approximate the expectation of the form

$$\widehat{I}(f_t) = \int f_t(\mathbf{x}_{0:t})p(d\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \frac{1}{N}\sum_{i=1}^{N} f_t(\mathbf{x}_{0:t}^{(i)}). \tag{2.32}$$

While forming estimates using Monte Carlo sampling is simple, in theory, it is often impractical. In many cases, it is not possible to sample from the posterior distribution $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, or it is very inefficient (computationally demanding) to do so. Alternatively, Markov chain Monte Carlo (MCMC) methods can be used to sample from these distributions [22, 23]. Unfortunately, using MCMC algorithms for recursive estimation (of the propagation of the posterior density) is problematic since these algorithms are iterative.

Bayesian Importance Sampling

It is often difficult to sample from the desired distribution directly. As an alternative to drawing samples directly from the posterior distribution, samples can be drawn from a known proposal distribution $\pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. This means that (2.32) can be rewritten as:

$$\begin{aligned}\widehat{I}(f_t) &= \int f_t(\mathbf{x}_{0:t})\frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{\pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}\pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t} \\ &= \int f_t(\mathbf{x}_{0:t})w^*(\mathbf{x}_{0:t})\pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t}\end{aligned} \tag{2.33}$$

where $w^*(\mathbf{x}_{0:t})$ is the 'true' importance weight

$$\begin{aligned}w^*(\mathbf{x}_{0:t}) &= \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{\pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} \\ &= \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{p(\mathbf{y}_{1:t})\pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}.\end{aligned} \tag{2.34}$$

As a result, if N i.i.d. samples $\{\mathbf{x}_{0:t}^{(i)}, i = 1, ..., N\}$ are selected from the distribution $\pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, a good estimate of the state distribution is

$$I(f_t(\mathbf{x}_{0:t})) = \frac{1}{N}\sum_{i=1}^{N} f_t(\mathbf{x}_{0:t}^{(i)})w^*(\mathbf{x}_{0:t}^{(i)}). \tag{2.35}$$

The importance sampling weight permits the sampling from another distribution to be possible as the importance weight can act as a conversion factor to the original distribution of interest. The difficulty of using the state estimate of (2.35) is that in most circumstances, this estimate requires an evaluation of

a normalizing constant $p(\mathbf{y}_{1:t})$. Define the unnormalized weight

$$w(\mathbf{x}_{0:t}) = \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{\pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} \propto w^*(\mathbf{x}_{0:t}). \tag{2.36}$$

Therefore,

$$\begin{aligned} I(f_t) &= \int f(\mathbf{x}_{0:t}) \frac{w(\mathbf{x}_{0:t})}{p(\mathbf{y}_{1:t})} \pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t} \\ &= \frac{\int f(\mathbf{x}_{0:t}) w(\mathbf{x}_{0:t}) \pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t}}{\int w(\mathbf{x}_{0:t}) \pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t}} \end{aligned} \tag{2.37}$$

and

$$\begin{aligned} \widehat{I}(f_t) &= \frac{\frac{1}{N}\sum_{i=1}^{N} f_t(\mathbf{x}_{0:t}^{(i)}) w(\mathbf{x}_{0:t}^{(i)})}{\frac{1}{N}\sum_{i=1}^{N} w(\mathbf{x}_{0:t}^{(i)})} \\ &= \sum_{i=1}^{N} f_t(\mathbf{x}_{0:t}^{(i)}) \tilde{w}(\mathbf{x}_{0:t}^{(i)}) \end{aligned} \tag{2.38}$$

where

$$\tilde{w}_t^{(i)} = \frac{w(\mathbf{x}_{0:t}^{(i)})}{\sum_{i=1}^{N} w(\mathbf{x}_{0:t}^{(i)})} \tag{2.39}$$

are the normalized importance weights. The 'true' importance weights have now been replaced by $w_t^{*(i)} = N\tilde{w}_t^{(i)}$.

This integration method can be interpreted as a sampling method. The posterior distribution $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ can now be approximated by

$$\widehat{p}(d\mathbf{x}_{0:t}|\mathbf{y}_{0:t}) = \sum_{i=1}^{N} \tilde{w}_t^{(i)} \delta_{\mathbf{x}_{0:t}^{(i)}}(d\mathbf{x}_{0:t}) \tag{2.40}$$

and the expectation can be represented by

$$\widehat{I}(f_t) = \int f_t(\mathbf{x}_{0:t}) \widehat{p}(d\mathbf{x}_{0:t}|\mathbf{y}_{1:t}). \tag{2.41}$$

This method of using a known proposal distribution to sample from is called importance sampling. As can be seen, importance sampling is a powerful method to use when the posterior distribution is not easy to evaluate. However, the downside is that importance sampling cannot handle recursive or on-line

estimations. This is due to the fact that the data $\mathbf{y}_{1:t}$ has to be collected before estimating $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. The importance weights of the entire space have to be recomputed each time new data $\mathbf{y}_{t+1}$ becomes available. As a result, the complexity of this computation greatly increases as time passes.

Sequential Importance Sampling (SIS)

In order to reduce the complexity of the algorithm, we can factorize the proposal distribution

$$\pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \pi(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})\pi(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t}). \tag{2.42}$$

Recursively iterating this equation,

$$\pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \pi(\mathbf{x}_0)\prod_{k=1}^{t}\pi(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{y}_{1:k}). \tag{2.43}$$

Consequently, the importance weights can also be recursively evaluated by

$$\begin{aligned}
\tilde{w}_t &\propto \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{\pi(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t})\pi(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})} \\
&\propto \tilde{w}_{t-1}\frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})}{p(\mathbf{y}_{1:t-1}|\mathbf{x}_{0:t-1})}\frac{p(\mathbf{x}_{0:t})}{p(\mathbf{x}_{0:t-1})}\frac{1}{\pi(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t})} \\
&\propto \tilde{w}_{t-1}^{(i)}\frac{p(\mathbf{y}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{\pi(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})}.
\end{aligned} \tag{2.44}$$

In order for sequential importance sampling to take place, one of the most important factors to consider is the choice of the proposal distribution. A common choice is the prior distribution

$$\pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = p(\mathbf{x}_{0:t}) = p(\mathbf{x}_0)\prod_{k=1}^{t}p(\mathbf{x}_k|\mathbf{x}_{k-1}). \tag{2.45}$$

The prior is often used because it is convenient. Another choice for the proposal distribution, which has been shown to be optimal, is to minimize the variance of w_t [24]. In this thesis, we restrict ourselves to using the prior as our proposal distribution for simplicity and because other proposal distributions require more joint information which increases the complexity of the algorithm.

Sequential Importance Resampling (SIR)

While sequential importance sampling eliminates the recursion problem, it still suffers from the degeneracy problem. Degeneracy refers to the fact that after a few iterations, all but one particle will have negligible weights. This occurs because the variance of importance weights increases stochastically over time [24]. Possible solutions to overcome this effect include using a very large sample size N or optimizing the choice of the importance density. The former approach can be too computationally demanding, and the latter requires the difficult identification of an optimal importance density. A selection process or *resampling* is an alternative solution to this problem. In resampling, samples that have very low importance weights are eliminated and replaced by replicas of samples with high importance weights.

There are a variety of resampling schemes. Each scheme has a different performance, most notably in terms of complexity and the induced variance of the particle set. Some notable techniques are residual sampling [25], systematic sampling [26], and mixture of SIS and SIR (only resample when necessary) [24, 27].

Unfortunately, even when resampling schemes are used, degeneracy may still be a problem. Samples may eventually collapse to a single point if, during the resampling stage, samples with high importance weights are duplicated an extremely large number of times. Again, there have been numerous proposals to rectify the problem. Notable techniques include auxiliary particle filters [28], local linearization using EKF [24, 28] or the unscented Kalman filter [29] (another variation of the Kalman filter) to estimate the importance distribution and MCMC methods [30–33].

2.1.4 Basic Particle Filter Algorithm

In summary, the particle filter keeps track a set of candidate state descriptions (particles). Each of these particles is associated with a weight. When a new measurement becomes available, each particle's state trajectory is augmented based on the dynamic model (and possibly also with the measurement). This stage is called the *propagation* step. The weights are then updated according to how well they describe the dynamic model and the likelihood of it generating the current measurement. This is known as the *update* step. The set of

particles along with their associated weights which approximates the filtering distribution can then be used to form the current state estimate. This is the *estimation* step. Finally, a *resampling* step is normally applied to eliminate particles with low weights and duplicate particles with high weights. This allows us to have a set of particles that focus on the likely region of the state-space of concern. A high level of the particle filter is described in Figure 2.2.

1. Initialization: $t = 0$;
   - For $i = 1$ to N: Sample $x_0^{(i)} \sim p(x_0)$;
   - Set $t = 1$;
2. Importance sampling step:
   - Propagation – For $i = 1$ to N: Sample $\tilde{x}_t^{(i)} \sim p(x_t|x_{0:t-1}^{(i)})$;
   - For $i = 1$ to N: Set $\tilde{x}_{0:t}^{(i)} = (x_{0:t-1}^{(i)}, x_t^{(i)})$;
   - Update – For $i = 1$ to N: Evaluate the importance weights $w_t^{(i)} = p(y_t|\tilde{x}_t^{(i)})$;
   - Normalize importance weights $\tilde{w}_t^{(i)} = w_t^{(i)} / \sum_{j=1}^{N} w_t^{(j)}$.
3. Estimation – The estimate $\hat{x}_{0:t}$ can be determined using the particles $\tilde{x}_{0:t}^{(i)}$ and weights $\tilde{w}_t^{(i)}$.
4. Resampling step:
   - Resample with replacement N particles $(x_{0:t}^{(i)}; i = 1, ..., N)$ from $(\tilde{x}_{0:t}^{(i)}; i = 1, ..., N)$ according to the normalized importance weights $\tilde{w}_t^{(i)}$;
   - Set $t = t + 1$ and
   - Proceed to the importance sampling step when the next measurement is obtained.

Fig. 2.2 High-level algorithm description of the particle filter algorithm.

2.2 Sensor Networks

A sensor network is a collection of tiny devices (called nodes or motes) that can be deployed in large numbers to monitor some element in the environment.

The nodes are connected together via wireless communication with the ability to detect, or monitor some aspect of an environment. Each node in a sensor network is self-powered and contains its own processing power. Unfortunately, each node has limited energy, memory, and computation power. Each sensor network node is usually equipped with a radio transceiver, a small microcontroller and a power source, such as a battery. Sensor networks are usually self-sufficient devices, with self-organizing capabilities since often there is no human intervention after they are deployed.

Arguably, sensor networks were first used in the military during the Cold War. The Sound Surveillance System (SOSUS) [34] was constructed with the purpose of tracking submarines deep underwater by attempting to detect their faint acoustic signals. There are many applications for these powerful networks including governmental, commercial and personal purposes [4, 35–37]. Their uses include (but are not limited to) measuring temperature, sound, pressure, motion, and pollutants. For example, a sensor network could be used:

- to detect or track threats, such as enemy vehicles or toxins in the air by the military,
- to monitor bacterial levels in a swimming pool,
- to detect the onset of a tornado, or
- to manage inventory in a store.

The use of sensor networks is advantageous for many reasons. For instance, since nodes are very tiny in size, they virtually do not disturb the environment in any way. They can be placed in an environment discretely and without anyone's knowledge for intelligence purposes. Furthermore, sensor networks can be deployed in places that are difficult to get to, or unsafe for humans presence. A plane can fly-by an area of interest and "drop" the sensor network.

Currently, Crossbow [38] is the most popular commercial company selling sensor network devices. Crossbow's best selling sensor network products are the MICA series development kits. An example of a kit is shown in Figure 2.3. These kits typically include a base station with several motes and sensor acquisition devices. These motes run under the TinyOS system [39] for embedded sensor networks, developed at the University of California at Berkeley.

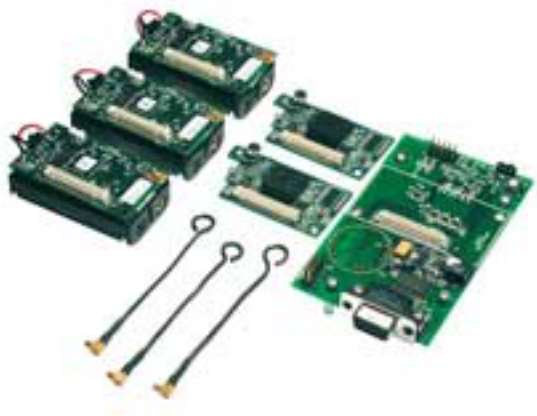


Fig. 2.3 Crossbow's MICA2 basic sensor network kit. This kit contains three MICA2 Processor/Radio Boards (motes), two MTS300 Sensor Boards (Light, Temperature, Acoustic, and Sounder), and one MIB510 Programming and Serial Interface Board (base station). Source: Crossbow (http://www.xbow.com)

The MICA series motes are about an inch or two long on the widest dimension, depending on the model. Another popular sensor network product comes under the name of smart dust [1–3]. These motes are designed to be the size of a particle dust containing all the features required in a sensor network, such as detection, communication and computational capabilities. However, the smart dust is still highly developmental and is not yet available to the public. Despite the availability of sensor network products, basic construction of a sensor network is not difficult. The main components of this network are microprocessors, sensors, and wireless communication components.

2.3 Collaborative Signal Processing in Sensor Networks

Some challenges faced with sensor network research today are to ensure high accuracy, low latency, low energy consumption, low ratio of active sensors, and fast computational time in any application. A common practice in sensor network applications has been to process collected data at a central server. This type of detection algorithms are known as centralized algorithms. Centralized algorithms are generally simpler to execute as processing data at one location can reduce the computational complexity of an algorithm. However, there are various problems associated with this type of methodology. The first problem

with centralization is that the network is relying on one processor to perform the task of the network. This introduces the single point of failure problem. Secondly, in real-time applications, sending data from a node to a central processor may take too long. The central node may reside many hops away, introducing latency and synchronization issues into network design. This also leads to an imbalance of the workload in the network. Nodes closer to the fusion centre exert more energy performing routing, while the fusion centre performs most of the computations. Lastly, centralized algorithms utilize a great deal of resources to transmit the data. These methods usually keep data processing procedures simple, while neglecting on communication costs. However, the energy consumption corresponding to computational work usually is lower than the consumption related to communication and therefore avoiding high data transmission is advantageous in this type of a setup. To avoid or at least reduce the major problems inherent with centralized algorithms, distributed processing over the sensor network can be used to alleviate the costs.

The development of sensor networks with distributed algorithms has largely focused on three research areas which deal with these factors [4]. The first is sensor management. Sensor management research involves the optimization of sensor resources to collect data. Since the location of deployment of sensor nodes is not usually exactly predetermined, sensors have to be able to locate neighbouring nodes in order to be able to form a network. This leads us to the second area of importance, the communication between all the distributed nodes. Nodes generally have very limited communication range and power. Therefore, the communication protocols used to transmit data between nodes in the sensor network must be efficient and a minimum due to its limitations. Finally, the data processing in the network is another key research area. This is the area where algorithms are developed to analyze the sensor data in order to achieve its tracking or detection goal. One particular topic of interest in this research field is collaborative signal processing (CSP). CSP deals with the processing of sensor data separately or distributively (i.e. at different nodes) in order to achieve a global solution. That is, rather than processing all data at a single location, computational methods are devised to obtain a similar solution by distributing the calculations among several nodes. For example, this allows for *distributed algorithms* to be performed with the absence of a

data fusion centre.

The primary focus of this thesis is on the collaborative signal processing aspects of tracking using a sensor network. That is, the focus is on creating distributed algorithms which minimizes energy consumption and network communication relating to sensor data fusion or collection. We assume that sensor management and communication issues such as network protocols are already efficient enough for our purposes. We are particularly interested in using a sensor network to track an object, using a powerful scheme of the particle filter, as described earlier. In [19], the particle filters was shown to be more effective for position, navigation and tracking, compared to other algorithms, such as the classical Kalman filter.

2.4 Literature Review

Distributed algorithms for sensor network have recently gained much attention by sensor network researchers. One of the first developments of a distributed particle filter algorithm was in [40]. The paper proposes two methods to use a distributed system for tracking. The first algorithm is based on a likelihood factorization. In this method, parametric approximations to the likelihood factors are formed using the particles of the filter and their associated likelihood as training data. The idea is that the model parameters can now be exchanged between nodes instead of the particles itself. The second algorithm is based on an adaptive data-encoding scheme. The distributed particle filter algorithm works by maintaining particle filters at a set of nodes dispersed throughout the network. The unique aspect of the distributed particle filters is that the local particle filter at each of these nodes is used to efficiently encode the local measurements. In order to encode data at the next measurement instant, the local particle filter at the current time step is propagated (blindly) according to the dynamic model. An expected measurement value can be obtained from each particle state. The expected values are then used to quantize the actual measurement by performing the Lloyd-Max algorithm [41,42]. Straightforward labelling of the data bins is then used to encode the local data for transmission to the other class A nodes. In order for this quantization and encoding scheme to be successful, it is crucial that all local particle filters match in the sense that the set of particles are identical. This can be achieved by initializing

the particle filters using the same random seed and ensuring that they all propagate based on the same distributed, quantized measurements. Note that these nodes must have some knowledge about the measurement process at each sensor node, for example, sensor position and calibration. This information can be communicated with the encoded data if necessary. The scheme performs very well because the particle filter should be able to provide the location of where the object is likely to be.

Other various CSP methods have also been examined in the tracking scenario for sensor networks. In [43], a *location-centric* approach is taken whereby a cell, consisting of a collection of nodes close to the object to be tracked, is formed to collect data about the target. A cell manager then has the responsibility of creating the next cell to maintain surveillance of the object. While this minimizes the communication cost in the network by selecting the necessary nodes to track the object, the disadvantage is the overhead in cell coordination and cell selection for the next time step. In [44], a *direct diffusion* concept is introduced for tracking. The proposal brought forth in this paper is that data communication should be minimized between the data source and data sink by setting up a gradient along each path, which describes the strength of the "interest" of the neighbouring nodes on particular data attributes available. Again, a disadvantage still exists in this scenario as there is overhead cost to setup this direct diffusion.

Researchers from the Palo Alto Research Center (PARC) examined collaborative signal processing methods in [45–47]. They explored a leader node methodology termed *information driven sensor query* for sensor collaboration. The general idea is that a "leader node" is the only device responsible for taking any measurements. That is, only one node in the network is active at any given moment in time. A group is formed about the leader node in preparation of a handoff of leadership. Specifically as described in [46], the leader node is used to track the object at every time step using a particle filter. A node is elected leader based on the information utility and cost of its neighbouring sensors. Once selected, a hand-off of information to a new leader node is required whenever the leader changes. This involves transmitting the particle filter in the form of raw particle values and weights or training and communicating a parametric approximation. While this scheme does reduce the amount of energy consumption if the rate of the particle handoff is high, the problem is

that if a query is made on the location of the object, it may be difficult to determine where the active node is to retrieve its estimate. The scheme also adds communication cost in order to manage the states in which a node may be in (i.e. as a leader, or being idle waiting for a handoff from the leader). The authors also have investigated techniques for approximating and transmitting the distribution of particles in an extension of this work.

On another front, the authors of [48] adopt a similar approach to the distributed particle filter of [40]. The novelty brought forth is that an EM algorithm is used to train a Gaussian mixture, to approximate the particle representation. As a result, the mixture parameters can be exchanged instead of particle weights. This approximation allows for a significant reduction in data communication between nodes, especially if a large number of particles is used. In [49], each node has its own local particle filters. Using a query-response system, neighbours send each other the most relevant information. In the scheme, the query node transmits a small subset of particles with the entire state trajectory, describing the most informative information contained within its local memory. In response, the queried node evaluates its own local memory in conjunction with the newly received information and exchanges its most informative and unshared data. Therefore, only important information is exchanged and communication is kept to a minimum. Shin et al. [50] proposed a distributed method whereby nodes use their local data to formulate an estimate of the state. Data is only transmitted when other users of the system requests for them. A belief matrix is used to keep tab of the target.

Particle filters in the context of sensor networks have also been analyzed from other angles. For example, in [51], the authors examined the particle filter as a problem of lossy encoding of a non-parametric density. The main issue they consider is how to measure the loss or error of regenerating an equivalent particle set from a distribution represented by the actual particle set. They note that a good measurement of error controls takes into consideration both the difference between the actual and regenerated distributions, but also the errors which could arise after in the future due to the approximations used. They explored three possible measures of error – the maximum log-error, the Kullback-Leibler divergence, and the integrated absolute error. In [52], the authors examine the problem of distributed particle filters with a focus on the resampling step, and an emphasis on increasing speed and reduc-

ing complexity. In their approach, called the *resampling with nonproportional allocation*, resampling is done at two levels. First, the procedure is invoked locally to perform resampling of the particles on each local node. Then, an inter-nodal resampling procedure has to be performed in order to normalize the weights of all the particles in the network. This step requires particles to be exchanged between local nodes, meaning weights and constants need to be transmitted. This procedure can be particularly expensive if the number of particle exchanges is high.

Chapter 3

Particle Filters in Sensor Networks

The basic particle filter, as described previously, is an excellent algorithm to use for tracking an object. However, when the particle filter is used in the context of a sensor network, communication and energy cost becomes an important factor in the performance of an algorithm. Particularly in a sensor network where nodes are possibly spread across a vast area, using a particle filter at a fusion centre, where all measurements are collected for processing, becomes costly in terms of communication and energy costs. In this chapter, the primary goal is to examine the particle filter in the context of a sensor network to determine if any of the steps in this procedure can be modified to run more cost efficiently, while maintaining tracking efficiency. Our focus is to avoid the use of a fusion centre and to use a more collaborative signal processing approach such that data can be processed close to the data source, reducing energy costs associated with transmitting information. Energy costs are determined by analyzing both communication and computational costs. The main concerns of the particle filter we analyze are the approaches toward maintaining a particle filter in the network, the transfer of particle filter information, the maintenance of an accurate particle set, and the efficiency of the estimation and transmission steps of the algorithm.

3.1 Approaches to Maintaining a Particle Filter in a Sensor Network

Recall the basic particle filter algorithm described in Section 2.1.4. This algorithm is simple to use when the particle filter is maintained in a single node. This is because once the measurement data is received by this node, the computation can be straightforwardly performed at this location without the need for extra steps or complexity. We call methods which employ a fusion centre to collect measurement data for processing as *centralized* algorithms.

While the complexity of centralized algorithms is low, there are various problems associated with these methods, as discussed in the previous chapter. We will repeat them again to recall the problems. First, despite the fact that the only communication exchange necessary in the network is to transmit sensor data to the fusion centre, this communication cost can be extremely high. This is because individual nodes with sensor measurements may be located many hops away from the centre, and the data must be relayed through many nodes. Not only does this increase the transmission cost in the network, but this also consumes a lot of the limited energy power of the nodes. In addition, this topology creates an unbalance distribution of the energy consumption in the network, particularly near the region of the fusion centre. The fusion centre needs to exert energy at every time step to maintain the particle filter while the nodes located near the centre consumes an immense amount of resources to route data. In a sensor network, this is extremely undesirable as each node only has limited energy resources. Draining energy quickly can therefore lead to a very short network lifetime. Furthermore, as the particle filter resides on one designated device, a major problem occurs if this node fails. Consequently, using a fusion centre to collect the sensor data and maintain a particle filter is not a very good choice in a sensor network architecture due to the unbalanced energy consumption and high communications costs. As a result, centralized schemes should be avoided and a distributed architecture in which a particle filter can be maintained without the need of designated centre to collect all measurement data should be considered.

We consider four distributed scenarios to maintain the particle filter for tracking:

- Particle filter resides at a different node each time step
- Particle filter resides at multiple nodes (simultaneously)
- Distinct particle filter resides at individual nodes
- Particle filter is dispersed over several nodes

3.1.1 Particle Filter Resides at a Different Node Each Time Step

One method to address the uneven distribution of the energy costs in the sensor network is to use an approach based on the centralized architecture. As opposed to using a single fusion centre, the concept of a "leader node" [46] can be utilized. The leader node has the responsibility of maintaining the particle filter at each time instant, identical to the task performed by the fusion centre in the centralized case. The difference with this concept is that the leader node is not always the same device at every time step. Ideally, if this leader node is strategically positioned at a location close to the object's predicted position, this solution is very beneficial in a sensor network context because energy consumption is no longer centred on a single node. Instead, the energy consumption is distributed along the path of the object. This means that during the entire tracking process, more nodes are involved in maintaining the filter. Energy consumption is thus distributed and the entire network can last longer. Another benefit to this method is that we can avoid the single point of failure problem since the particle filter is maintained over multiple nodes during the course of operation, although only one node is active and has the most up-to-date information at any given time. If a leader fails, it is possible to activate another leader node to continue the tracking process by obtaining information from the previous leader.

In order for this approach to work, particle filter information must be handed off to the new leader when activated. Unfortunately, this information transfer has additional communication cost associated with it that was unnecessary in the centralized case. A further disadvantage arises if the data dimension of the filter is large. This would mean that more information would have to be exchange to the new leader. However, if the data dimension is low and this information exchange procedure is performed efficiently, the communication cost of particle filtering information handoff can cost less than by

routing data from the sensors to a fusion centre. At the very least, if the cost of this leader node approach is equal to that of the centralized case, the former case is still more beneficial since the cost will be distributed over a larger group of nodes meaning the network lifetime is extended. Particle filter information handoff will be analyzed later in this chapter.

The only downside to this method is when a query is made for retrieving the particle filter results. Since only one node has this information, and this node changes periodically, a search must be performed to find the node. This could potentially be costly if the query rate is high.

3.1.2 Particle Filter Resides at Multiple Nodes

As an extension to the leader node concept, we can adopt a proactive, rather than reactive methodology in handling particle filtering information. Instead of only handing off information to new leader node when activated, we can allow all possible leader nodes to maintain an identical copy of the particle filter. This possibility was analyzed in [40]. If information exchange can occur at every time step, or even periodically, all nodes can be updated with the latest information and ready to perform any particle filtering duties immediately when required. As an additional benefit, should any leader node fail at any particular time during the course of its operation, the other nodes can be immediately ready to continue the tracking tasks.

Obviously, this scheme requires greater communications and computational costs as compared to the leader node method to maintain since multiple nodes have to maintain an updated particle filter. However, if query is made to obtain the current estimate formulated by the particle filter, any node in the network could respond immediately. There is no need to search for a specific node, as was the case in the leader node.

3.1.3 Distinct Particle Filters Reside at Individual Nodes

Another filtering scenario we can consider is one where a different particle filter is maintained at individual nodes. In such a method, each node would use its own sensor data to form an estimate by itself (which is exclusive to the other nodes). This avoids the need to update each node with every other node's sensor measurements or particle filter information. Communication costs will

then be significantly lower. In addition, the hassle of ensuring that all nodes maintain identical particle filtering information is eliminated, as was the case in the previous proposal.

The drawback to solution is that nodes that are located far away from the object would use its own local sensor which may not give accurate sensor readings. This means that some nodes maintain an inaccurate particle filter, which waste energy resources. Another complication in this scenario is that each node computes a different estimation of an object's location due to different sensor readings and different particle set maintained at each node. There is no way for an external query to determine which node has the most accurate tracking estimation. Finally, the computational costs are higher as compared to the centralized case since multiple filters are maintained simultaneously.

At first glance, it appears that there may be no real advantage in using this proposal since each node computes a different estimation, with nodes located far away from the object giving inaccurate results. However, in the case when a query is made to a node close to the object's position, a reasonable estimation can obtain which could be adequate enough for certain applications. In addition, the entire process of maintaining a particle filter would cost very little in terms of communication costs since there is no need to exchange sensor data.

By itself, the method is probably undesirable. However, this method can possibly be advantageous if a mechanism can be put into place where only nodes close to the objects are activated to maintain a particle filter. Of course, newly activated nodes will have to be updated somehow with latest particle filter information.

3.1.4 Particle Filter is Dispersed over Several Nodes

Another form of maintaining a particle filter in a sensor network is to distribute the particles over several nodes. That is, the entire set of particles can be divided into small subsets, each residing on a different node. The advantage is that it can be more manageable to place particles on various nodes due to the memory limitations of a single node, especially if a large set of particles is maintained.

Using a large number of particles in the particle filter is desirable to give

the best possible representation of the object's state. Recall that particles are candidate state descriptions of the object's position. The greater the number of particles is used in the filter, the more accurate the estimate obtained should be. This is the typical trade-off issue between accuracy and computational cost.

Despite the advantage of using larger particle set, this adds a complication in maintaining the particle filter. The traditional centralized particle filter requires the consolidation of data at one node in order to compute the estimate. Applying this action to this method would defeat the purpose of distributing the data over several nodes. Therefore, another methodology to perform the particle filter has to be used. For example, the individual nodes maintaining subsets of particles can run the particle filter independently using its own particle subset only. This way, there is no communication cost required due to exchanging particle filter information or measurement data.

In summary, while the choice of a local or distributed particle filter can significantly affect the communication costs of the tracking system in a sensor network, this factor alone is not enough to determine which methodology is best for the application. An important issue to consider is how to transmit particle filtering information to another node, should a distributed method be used. Whether the network architecture calls for the use of a leader node or for several nodes to maintain some type of a particle filter, the means of communication between various nodes to transfer particle filtering information is crucial to the overall costs, if necessary.

3.2 Particle Filtering Information Transfer

The whole basis of a particle filter is the particle set. The particle set, composed of particles each with an associated state (or value) and weight, makeup the particle filtering information that is processed in order to formulate the estimate of the object. When nodes exchange particle information, the key is for the receiving node(s) to be able to regenerate these particle states and weights.

The particle filter information can be transmitted to other nodes in a variety of ways, with each method differing in communication costs and particle filter information reconstruction accuracy. The most straightforward way to

transfer filtering information is to send the exact particles states and weights. This method is by far the most costly approach, even if compression is used, since the number of particles tends to be high.

Transmitting a subset of particles, such as those with the largest weights, is also possible and is discussed in [49]. Recall that an estimation of the object's position is determined using a weighted average of the particle states. Theoretically speaking, if we neglect the low weight particles in our estimation, it is still possible to obtain a reasonably accurate result. Depending on the number of large weight particle information sent, the procedure may be beneficial but this is a tradeoff between accuracy and communications cost. However, a balance of the two can provide an excellent alternative to the transmission of the entire particle set. Nevertheless, this procedure can also be beneficial if used in conjunction with the resampling procedure. Recall that resampling is invoked in the particle filter to eliminate low-weight particles in order to lower the variance of the particle set. These particles are replaced by replicas of the higher weighted particles. Thus, a transmission of a subset of particles with high weights and recreating the remaining particle using resampling is another feasible option.

Another method to transmit particle filtering information is to use a parametric representation of the particle set. The distribution of particles states can be parameterized and the value of the parameters of the distribution can be exchanged. For example, if the particles can be represented by distance value from the object, the distances along with their weights can be modelled in terms of a distance measurement distribution function. For instance, should a Gaussian mixture model be used to describe the particles, the mean and variance of the Gaussian distribution only has to be transmitted [48]. The node receiving these parameters just has to reconstruct the distribution based on the values received and reproduce a set of particles with the same distribution. While the exact particles would not be replicated, the particle set can still be effective if the overall approximation error to the particle values is low. Potentially, this scheme could require only a few bits of transfer per parameter which makes it attractive to use.

In a case where multiple nodes are simultaneously maintaining identical particle filters, it is not necessary to transmit particle states and weights, or parametric values representing the particle distribution. Instead, sensor

measurement data can be sent to these nodes to update the particle filter. With compression, the cost of transmitting this data can be comparable to that of a parametric representation. The advantage to this is that the exact particle filtering information can be reconstructed at every single node without a high cost. Thus any node will be able to give an identical estimation of the object's state should a query be made.

In brief, considering how to exchange particle filter information is an important factor to consider in a distributed particle filter methodology. However, it should be noted that the particle filter algorithm is based on the manipulation of the set of particles. This also has to be given attention when determining the type of algorithm to use to track an object in a sensor network.

3.3 Issue of Maintaining an Accurate Particle Set

As particles are propagated in time in a filter, they become less of an accurate representation of the object's state due measurement noise which increases variance of the set of particle values. As time progresses, the representation gets less useful and can eventually lose track of the object if some measure is not taken to ensure the validity of each particle. This issue is important to consider, whether a centralized or distributed particle filter is being used, since it deals with the nature of the particle filter itself. To address the issue of maintaining an accurate particle set, several methods can be considered.

3.3.1 Replacing the Particle Set

In a situation where different particle sets are maintained at each node, or where particle sets are distributed among various nodes, particle set replacement can be considered. For instance, the particle filter can periodically determine whether its particle set is still effective at estimating the object's position by computing the expected log-likelihood [53]. If the test shows that the particles are a poor representation of the object's position, this signifies that continuing to track the object with this particle set will produce unacceptable results. As a result, the entire particle set at this node can be replaced by a set from another node.

A crucial factor to consider when using this method is to determine the ex-

act parameters to when particle set replacement should occur. The balance of this value in conjunction to the communications cost associated with particle set transfer is of great importance in a sensor networks context. The particle transfer method used can be one of those that was just discussed in the previous section – that is sending the raw particle values or sending a parametric representation of the particle set.

3.3.2 Resampling

Alternatively, when the set of particles is no longer a good representation of the object's position filter, it is due to the fact that the particles have a large variance due to system noise. This means that only a few particles have weights that are substantial to estimate the object's position. This situation can be corrected by using a resampling procedure, whereby a subset of the current particles which no longer provides useful information is replaced. This procedure can be done at every time step, or periodically.

As discussed in Section 2.1, resampling works by eliminating particles with low weight and replicating those of higher importance (i.e. larger weights). The particular issue in dealing with resampling in a distributed particle filter in a sensor network is to determine how to normalize the weights of all particles in the system in order to perform a proper comparison of all weights. In the centralized case, the weights are all available at the fusion centre and therefore resampling can be performed straightforwardly. In the distributed case, there are two resampling scenarios to examine.

In the first scenario, we consider resampling at a node where all the particles are available without any extra communication necessary. This is the case when a leader node approach is used or when multiple nodes maintain a particle filter. Normalization in these schemes can be achieved with no problem as this *local* resampling can be performed simply by utilizing the weights of particles at the node itself. No extra communication or computations is required.

In the second scenario, we consider the case where particles are dispersed over multiple nodes. A local resampling procedure, as just described, can be invoked to keep the variance of the particles low at each node. However, when the local normalizing factors become large, a *global* resampling should be performed on the entire particle set maintained in the network in order to ensure

that the variance of the overall set is low. Global resampling is performed using both particle weights and local normalization constants. Local normalization factors are needed because the local resampling procedures altered the weights of the particles, and thus a direct comparison of the normalized weights is unfair.

Global resampling can be performed using several methods. One approach is to have all nodes broadcast their particle weights and normalization constants so that each individual node may be able to perform the resampling. This can be very costly in terms of communication and computational costs of the entire system. A variation to this method is to use a set of intermediary nodes to perform the collection, where each node reports to its intermediary node. The intermediary node can then communicates with each other in order to aggregate the weights. Another extension is to build a binary tree structure with all nodes acting as elements of the tree and a summation can be performed by a parent node on its children [54].

An additional method to achieve global resampling when particles are disperse over multiple nodes is by using a central transceiver. The concern with this proposal is that the communication cost associated with transmitting information to a transceiver can be high. Obviously, a central transceiver should only be considered if the information exchange can be performed in an efficient manner. Note though, that there is still a concern of a single point of failure in this case.

In short, the issue of maintaining an accurate particle set is necessary to consider in order to ensure the overall effectiveness of the particle set in the filter. Whether a particle set replacement or resampling is used, these are important factor in a particle filter framework. Particular in our context, this has to be considered as it affects the communication costs associated with the algorithm.

3.3.3 Object Estimation

In estimating the object's state, a weighted average of the particle values is computed. Similar to the global resampling problem, this estimation requires the collection all particle information. In the case of the leader node approach or where multiple nodes maintain particle filters, estimation can be performed

with no additional communications costs. In the case when the particles are distributed over multiple nodes, a consolidation of particle information is required to formulate an estimate. The methods available to achieve this are identical to those proposed for the global resampling procedure. In fact, the only difference between global resampling and estimation problem is only the values required to perform their respective operation. Therefore, the methods needed to achieve either procedure are the same meaning that a global transceiver or the use of intermediary nodes to aggregate the values can be used.

3.4 Transmission Efficiency

In the previous sections, we focused the discussion on ways to minimize data transmission in the network. However, the transmission cost of each step of the algorithm should not be the only factor we should consider. Another aspect to look at is efficiency of the transmission process itself. For example, in order to ensure transmitting data in the network is efficient, data compression or quantization of the data to be communicated should always be utilized. The level of compression has to depend on the accuracy required in decoding the data. In addition, communication protocols should also be considered, such as whether handshaking is required, and what type of packets should be used to send the information efficiently. Each type of solution proposed can use various types of compression algorithm, and as a result, the benefits of each can significantly change depending on the type used.

Compression can be performed in many fashions. One of the simplest method is to use quantization. For example, in scalar quantization, an entire space can be divided into sections. Compression is achieved by using the representation of the label associated with the region the value belongs to [55]. In order for quantization to be useful, the number of labels should be significantly less than the number of potential values in the state space.

Depending on the type of quantization required, there are also a slew of other methods such as uniform quantization, predictive quantization, tree-structured quantization, just to name a few [56]. There are also other algorithms to consider such as Lloyd-Max [41, 42], minimum weight spanning tree [54], and entropy encoding [57].

3.5 Summary of Particle Filtering Issues in Sensor Networks

In this chapter, we considered the issue of using a particle filter in a sensor network, with a focus on the communication and energy costs. We wish to use the powerful particle filter in a tracking algorithm within the context of a sensor network, ensuring that network lifetime is maximized (by minimizing the associated costs), while maintaining tracking accuracy. Sensor networks have limited energy resources and therefore we wish to have an efficient algorithm. We considered various particle filtering issues in sensor networks, such as the approaches to maintaining a particle filter, the transfer of particle filtering information, the maintenance of an accurate particle set, and the efficiency of the estimation and transmission processes.

Pertaining to the approaches of maintaining a particle filter, we indicated that the centralized form of the particle filter in a sensor network is computationally simple. However in this scenario, there is a high communication cost associated with transmitting data from the sensor to the fusion centre. In addition, this type of approach has an effect of unevenly distributing energy costs in the network, with the most located in the area near the fusion centre. This is due to the energy requirement for relaying sensor data to the fusion centre, and for the fusion centre to perform the particle filter computations. As an alternative, we looked at a few distributed methods.

The first proposal given was a leader node approach, similar to the centralized fusion concept. A leader node maintains the particle filter at every time step, but the device designated as leader periodically changes. It is preferable that the leader is located near the object's predicted position in order to reduce the amount of communication associated with relaying measurement data. This approach distributes energy consumption to the nodes near the object's path, prolonging the network lifetime. Communication costs are also reduced.

Another scenario discussed is a proactive leader node methodology, whereby potential leader nodes are updated with particle filtering information periodically. Since these potential nodes may eventually require the information, this saves on time in making the transfer. Furthermore, should a query be made about the object's estimated position, all potential nodes are able to respond

immediately. If communication can be made to be efficient, there is lots of potential for this method.

The third scenario described involves maintaining distinct particle filters among various nodes. In this case, each node would be able to use its own sensor(s) to take measurements. This data can then be used to maintain the local particle filter. The advantage with this scheme is that there is no need for communication among the nodes. However, the disadvantage is that sensor data obtained from nodes located far away from the object is not reliable, and thus this is only advantageous for nodes residing close to the object. In addition, each node will compute a different estimate, leading to questions about which node to query to get the best results. However, the scheme can be advantageous if a mechanism can be put into place where only nodes close to the objects are activated to maintain a particle filter in this manner.

The last scenario described involves distributing the set of particles among several nodes. This allowed for a larger particle set to be maintained in the network, since each node has only limited memory and computational power. The disadvantage is that the data from the individual nodes have to be consolidated together in order to perform a particle filter. This can resolved simply if individual nodes have the means to process a particle filter exclusive to the other particle sets in other nodes, so that the particle filter information and measurement data do not have to be consolidated at one node.

Examining these particle filter scenarios based solely on the (distributed) location of where the particle filter resides is not enough to determine whether or not that method is beneficial. This is because there are many other issues to consider in order to determine their effectiveness of an algorithm employing one of these methods. Particle filter information transfer is also important to consider as this can potentially dictate whether a proposed distributed algorithm is cost effective or not. There are three main methods proposed for particle filtering information transfer. The first is to send a subset of the particle information – that is of those particles with the highest weights – and using resampling to replace the remaining particles. This method could be potentially costly as the number of particle information that should be transmitted tends to be high. Another method is to parameterize the distribution of the particle set and transmitting only the parameters necessary to describe the distribution. If the dimension of the particles is low, the transfer of par-

ticle information can be performed using very few bits. The disadvantage of this method is that the exact particle set will not be regenerated, but is a good alternative if this level of accuracy is not required. The last proposal discussed was to send compressed versions of the measurement data in the case when multiple nodes are maintaining a common particle filter. This method is possible since all nodes have identical particle filter information and only the measurement values are needed for updating. The communications cost associated with this method is comparable to the parametric methodology, and has an added advantage of allowing for the exact recreation of the particle set at each node.

Another important issue to consider when using a particle filter in a sensor network is maintaining an accurate particle set. As was explained earlier, the set of particles will have a large variance as it is propagated in time due to the noise in the system. Depending on the case, this can be rectified by either discarding the particle set in the local node and replacing it with the set from another node, or by using a resampling procedure. The resampling procedure in itself provides various difficulties in some cases if the particles are not localized at one node. The weights of all particles, as well as local normalization constants have to all be considered in the case of a global resampling. This can be done using a central receiver, or by using a scheme like a minimum weight spanning tree. This also holds the same for the estimation procedure after obtaining the particles and their associated weights. If the particles all reside at one node, only local resampling is needed and this can be performed simply with no extra information exchange.

Finally, we also need to consider the actual transmission costs. Much focus was put into lowering the amount of transmitted data necessary for the algorithm to succeed. In fact, the overall energy cost varies depending on how well a compression algorithm performs to send the various types of data required.

Determining an optimal distributed scenario for tracking in a sensor network therefore requires the consideration of many factors. The effectiveness of each proposed solution for each factor greatly depends on the type of tracking scenario considered and the circumstances surrounding the sensor network system. In this chapter, we have built up a framework to devise a distributed particle filter algorithm. In the next two chapters, we will propose two different

algorithms that can be used depending on the sensor network architecture and scenario. The first proposal algorithm which will be presented is a case when sensor measurements are made available at multiple nodes. It is important in this case to consider an efficient data fusion-like methodology which has a high data compression ratio for transmission. In the second proposal algorithm, a local particle filter is used to act on local data and can perform tracking duties without the need for a global data fusion centre.

Chapter 4

Parallel Distributed Particle Filter

As discussed in the previous chapter, employing a fusion centre to maintain a particle filter in a network is highly inefficient. This type of methodology requires the use of an intense amount of communication to function, which in turns consumes a high amount of energy and reduces the overall lifetime of a sensor network. In this chapter, we propose a distributed particle filter algorithm that will alleviate the high communications costs based on a simple networking scenario where sensor measurements are obtained from various nodes in a sensor network.

4.1 Proposed Solution

A typical sensor network is comprised of processing nodes and sensors. Each sensor is usually associated with a single node and therefore located in a position close to this node. In a situation where a sensor network is used to track an object, it is often the case that the sensors located closest to the object's predicted position are activated to take measurements. These active sensors may belong to different nodes and therefore a mechanism must be in place to utilize all the sensor data to formulate an estimate.

The most computationally simple solution to maintaining a particle filter is to employ a fusion centre, where this unit collects all the measurement data and processes it. As has been discussed, the measurement data may be ob-

tained at a location far away from the centre, and therefore the communication costs associated with relaying this data will be high. Alternatively, the inherent problem with this structure can be rectified by employing a leader node approach. The leader node concept permits this fusion centre to be a node located near the object. Therefore, communication costs can be dramatically lowered since the measurements should be obtained and relayed to a location near the leader. The problem with this latter proposal is that if an external query was to be made on the location of the object, a search would have to be conducted to determine the location of the leader node. Furthermore, in order to ensure that the new leader node is given the updated particle filter information, extra costs will be incurred.

As a result, we propose a structure that extends this model and eliminates the search for the leader node problem on a query. We propose that particle filters are maintained at all nodes. At first reaction, this suggestion may seem to be a worst solution than a centralized method since the communication costs can be dramatically increased due to the maintenance of extra particle filters that were previously unnecessary. However, a proposal can be given in which this architecture can be maintained while lowering the communication costs as compared to a fusion centre structure.

First, it is often the case that a query is periodically made for the estimated location of an object in a tracking system. As a result, it is not necessary to maintain an updated particle filter at every time instant. We can therefore utilize a scheme we term *vectorization*, where measurement data over several time steps are collected and sent together, thus reducing on the communication costs by eliminating the need to send extra packet headers. The details of this method will be described later.

Second, we can employ a scheme to ensure that the update of particle filter information is done in an efficient manner. As analyzed in the previous chapter, there are various methods to address this update. For this case, the only method we can consider is transmitting the measurement data itself since all measurements are not located at one node. Using any other scheme involving the particle sets would require the manipulation of multiple particle sets, which would be communicationally and computationally more demanding. We also want to do this to ensure that each node maintains identical particle filters, so that the estimations at all nodes are the same. This is a further advantage

as compared to the leader node approach. In order to transfer particle information to the next leader node ensuring that the exact values are maintained, the only method to achieve this is to transmit the exact particle values and weights, which is extremely costly.

In a sensor network, it is likely that only a few nodes are active at any given time. Since nodes are only turned on whenever at least one of its associated sensors are active taking measurements, only a small number of nodes should be expecting measurements, as the concentration of working sensors is located near the object. In the case where a query for the object's estimated location has to be made during the intervening time before all nodes are updated with all sensor measurements, it is quite possible that each active node may have enough active sensors to track an object by itself within a reasonable accuracy rate. Therefore, a particle filter can be used on these local sensor measurements to perform an estimation result if necessary.

Since each active node is maintaining an additional particle filter based on local sensor measurements, we can also use this filter to encode the measurement data without too much extra computational costs. As a result, another innovation we propose is a new quantization and encoding technique for the measurement data using the particle filter, which will be described in detail later.

With this proposed framework, we can outline an efficient procedure which functions in the manner described in this section. The algorithm will be performed in two layers. The first layer works by having active nodes located near the object's position collect measurement data for a period of time while maintaining a *local* particle filter using its associated active sensors. Then periodically, the measurement data collected from all nodes are exchanged, in order for all nodes to maintain their *global* particle filters. Before we describe the complete algorithm, let us discuss the two innovation procedures designed to reduce communication costs – our quantization and encoding procedure and our vectorization scheme.

4.2 Quantization and Encoding

Data transmitted in a sensor network is generally quantized to some extent; the simplest form of quantization is probably accomplished by dividing the entire

measurement space into small regions and transmitting the label of the region where the data lies. Nevertheless, this scheme may still require a substantial number of bits to represent the data. The distributed particle filter in [40] uses a similar procedure whereby a Lloyd-Max algorithm is trained according to propagated particle filters to quantize the measurement to achieve a higher compression ratio. However, the Lloyd-Max algorithm is computationally expensive. Here we propose a much simpler quantization method employing an efficient encoding scheme to achieve similar compression.

Our proposed quantization process commences by blindly propagating the particle filter from the previous time instant and calculating the expected measurement for each propagated particle. By blindly propagating, we are performing a propagation of the particle to the next time step by using only the dynamic model, without consideration of any measurements. Imposing this latter limitation can have a negative impact on the performance of the particle filter since the particles are not augmented toward the object's actual location. However, reasonable tracking accuracy is still maintained if it is used only for a short duration of time.

The next step is then to divide the range of the expected measurements into bins of equal size and form a histogram based on these measurements. These steps are illustrated in Figure 4.1. We construct a Huffman tree [58] using the histogram to develop the codebook for encoding the data measurements. The measurements can then be encoded by using the Huffman tree codeword representing the bin associated with the data. If the propagated particles are a good representation of the state, then the measurement should lie in a densely-populated bin and the codeword should consist of very few bits.

In the decoding stage, the quantized measurement values can be reconstructed by recreating the same Huffman tree because each node has the same particle filter conditions – and thus an identical copy of the same particle representation before quantization commences. The key in this encoding technique is that each node must have the same copy of the particle representation. This can simply be achieved by initializing all nodes with the same particle representation. As a result, this encoding step can greatly reduce communication costs in a system.

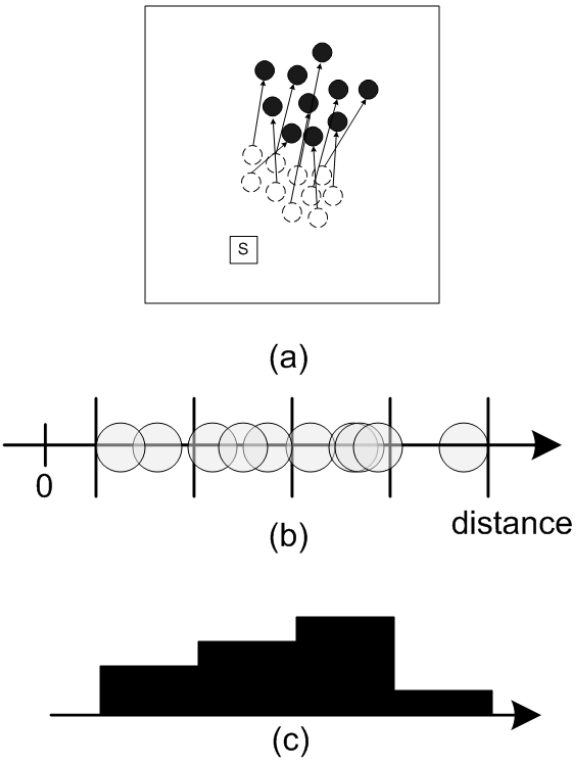


Fig. 4.1 Example of the proposed quantization and encoding stage at a node. (a) Original particles (circles) representing object position are propagated using the dynamic model. Shaded circles are the propagated particles. (b) The expected values (distance value) of the propagated particles are split into bins of equal size. (c) A histogram of the particle distance values is constructed.

4.3 Vectorization

Since obtaining the most accurate estimation is not necessary needed at every time instant, the global particle filter can only be occasionally performed. To update the global particle filter, the measurement data has to be disseminated to the other nodes.

To keep things simple, the same set of nodes and sensors will be activated to collect measurements during a "vector" interval. The set of sensors performing the measurement tasks will be the set of sensors which is closest to the object's predicted position of the beginning of the "vector" time interval.

Communications costs are dramatically reduced in this scheme because the communication costs associated to packet overhead is eliminated. As an

example, suppose each node collects data for five time steps before distributing its vector of compressed values to the other nodes. During the intervening time steps, only local nodes will have up-to-date *local* estimates and these will vary between nodes, as they are formed using only local measurements. The quantization of the data is performed by using a blindly propagated particle filter. When the five time instances have elapsed, the vector of measurements is shared with other nodes in the network and each of these nodes can run its particle filter on the data received. At this point, the *global* estimate of the state can be formed at all nodes. In this example, four time steps worth of packet overhead is eliminated which can provide substantial savings since the ratio of overhead bits to data bits is usually high.

4.4 Algorithm

With the main issues of how the tracking algorithm functions, and how communication between nodes is performed, we can now describe our distributed particle filter scheme in detail. In order for this method to function, the initialization process is crucial. The global particle filters in each node must produce identical results in order to allow for identical estimation and also to allow for the performance of the quantization and encoding/decoding procedure. This can be achieved if all nodes are initialized to the same particle filtering condition, meaning that each node is initialized to the same random seeding so that particles initially generated will be identical. In addition, this allows for identical particle propagation in all nodes since identical random numbers will continually be generated.

Once the seeds have been set, each node will create some particles from an initial distribution, with each particle having a weight of one. All nodes in the network will then be activated. Each node will get all its associated sensors to take a measurement of the position of the object. We assume that each node knows the location of all sensors in the network, whether it is their own sensors or the sensors belonging to other nodes. The system will utilize only a small subset of sensors at any given time. The subset will consist of sensors (for example, eight sensors) that are located closest to the object's predicted position. Therefore, each node has to determine if any of its sensors are among the closest sensors to the object. If so, that sensor and its associated node will

remain active. All other sensors and nodes will be put into a sleep mode for the time being, until it is required.

Next, the particles on each active node are propagated using the state dynamic model. The quantization and encoding procedure based on the propagated particles will be used and the encoded value will be saved. The particle filter then acts on these quantized measurements (i.e. bin value representation of the measurement) to generate an estimate of the object's position. We call this estimate the *local* estimate, since it is formed using local measurements only.

When a vector's length of time passes, the collection of saved encoded data will be transmitted to all other nodes. The other nodes will decode all information received by the exact same procedure for the encoding to generate the Huffman tree codebook. Once all data has been successfully recovered, the particle filter is run on *all* the measurements collected in the network. A *global* estimate can straightforwardly be generated since all the particles reside at the same node. Additionally, a resampling procedure can be used to ensure the variance of the particle set is low. Here, local resampling can simply be invoked.

The process is then repeated with the selection of a new subset of sensors closest to the object's projected position. Propagation, encoding and vectorization are then performed. A local estimate can be calculated if necessary. Finally, decoding is performed after the vectorized data has been distributed to other nodes and a global estimate can be formed.

In our specific scenario, even though it can be generalized, the distributed particle filter algorithm deals with the problem of estimating the state for a multi-dimensional signal (4-dimensions in our case) using a Markovian state-space model that is (potentially) non-linear and non-Gaussian. The unobserved global state $\{\mathbf{x}_t; t \in \mathbb{N}\}$ is modelled as a Markov process with initial distribution $p(\mathbf{x}_0)$ and a transition probability $p(\mathbf{x}_t|\mathbf{x}_{t-1})$. The observations $\{\mathbf{y}_t; t \in \mathbb{N}^*\}$ are assumed to be conditionally independent in time given the process $\mathbf{x}_t$. The system state and observations up to time t are denoted by $\mathbf{x}_{0:t} \triangleq \{\mathbf{x}_0, ..., \mathbf{x}_t\}$ and $\mathbf{y}_{1:t} \triangleq \{\mathbf{y}_1, ..., \mathbf{y}_t\}$, respectively. The measurements $\mathbf{y}_t$ are recorded by K sensors, and we use $\mathbf{y}_t^k$ to denote the subset of observations made by the k-th sensor. A high level algorithm is shown in Figure 4.2. We call our proposed algorithm the *Parallel Distributed Particle Filter* (PDPF).

1. Initialization, $t = 0$

   - Initialize the particle filter of each sensor $k = 1, ..., K$ by equating the random seeds to ensure they all match.
   - For each sensor $k = 1, ..., K$
     - For $i = 1, ..., N$ particles, sample $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)$.

2. Quantization and encoding at the nodes:

   - Set $t \leftarrow t + 1$.
   - For the length of the vector $v = 1, ..., V$

     (a) Quantization:
     - For each sensor $k = 1, .., K$
       - For $i = 1, .., N$, sample $\tilde{\mathbf{x}}_t^{(i)} \sim p(\mathbf{x}_t | \mathbf{x}_{0:t-1}^{(i)})$.
       - Calculate expected values of measurements $g_t^{(i)} = E(y_t^{(k)} | \tilde{\mathbf{x}}_t^{(i)})$.
       - Create a histogram of the $g_t^{(i)}$ values with h equal bins encompassing the range of values $[\min(\mathbf{g}_t), \max(\mathbf{g}_t)]$.
       - Use the histogram to form a Huffman tree H_t^k and encode the quantized measurements $\widetilde{\mathbf{y}}_t^k$.

     (b) Local Estimation:
     - Form a *local* estimate of the object's state using a standard particle filter acting only on the local measurements. That is, a weighted sum of the particles will be used to estimate the object's position. The weights can be calculated using the local measurements.

3. Network Communication:

   - For each $k = \{1, ..., K\}$, send the vectorized measurements $(\mathbf{y}^k_{t-V:t})' = \{(\mathbf{y}^k_{t-V})', (\mathbf{y}^k_{t-V+1})', ..., (\mathbf{y}^k_t)'\}$ to all other $K-1$ sensors.

4. Global Estimate:

   - For $t' = t - V, ..., t$
     (a) For each active node k, create the Huffman tree $H^k_{t'}$ to reconstruct the quantized data $\widetilde{\mathbf{y}}^k_{t-V:t}$.
     (b) Using $\{\widetilde{\mathbf{y}}^k_{t-V:t}, k = 1, ..., K\}$ as the set of measurements obtained for time interval $t - V : t$, apply a standard particle filtering algorithm to generate the *global* state estimates:
        i. Importance sampling
           – For $i = 1, ..., N$, sample $\tilde{\mathbf{x}}^{(i)}_{t'} \sim \pi(\mathbf{x}_{t'}|\mathbf{x}^{(i)}_{0:t'-1}, \mathbf{y}'_{0:t'})$, and set $\tilde{\mathbf{x}}^{(i)}_{0:t'} = (\mathbf{x}^{(i)}_{0:t'-1}, \tilde{\mathbf{x}}^{(i)}_{t'})$.
           – For $i = 1, ..., N$, evaluate the (approximate) importance weights $\tilde{w}^{(i)}_{t'} = \frac{p(\mathbf{y}'_{t'}|\tilde{\mathbf{x}}^{(i)}_{t'})p(\tilde{\mathbf{x}}^{(i)}_{t'}|\mathbf{x}^{(i)}_{t'-1})}{\pi(\tilde{\mathbf{x}}^{(i)}_{t'}|\mathbf{x}^{(i)}_{0:t'-1}, \mathbf{y}'_{0:t'})}$.
           – Normalize the importance weights.
        ii. Estimation
           – Form a *global* estimate of the object's state using a standard particle filter acting on all measurements obtained.
        iii. Selection
           – For $k = 1, ..., K$, resample with replacement the N particles $\{\mathbf{x}^{(i)}_{0:t'}; i = 1, ..., N\}$ from the set $\{\tilde{\mathbf{x}}^{(i)}_{0:t'}; i = 1, ..., N\}$ according to the importance weights.
        iv. Set $t' \leftarrow t' + 1$.

5. Go to step 2.

Fig. 4.2 High-level algorithm description of the parallel distributed particle filter approach.

4.5 Simulation Example

In this section, we put the proposed distributed particle filter algorithm to the test. We will first describe the simulation conditions including the network architecture and object dynamics used in our experiments. We then discuss how this simulation can be performed using physical hardware.

4.5.1 Network Architecture and Object & Measurement Dynamics

While our algorithm can be applied to a generalized case, we constructed an example networking scenario. In our setup, an object travels through the sensor network of size 128×128 metres. The dynamic system of the object's movement uses a jump-state Markov model [24], described by an initial distribution $p(u_0, \theta_0, \mathbf{x}_0)$ and update equations

$$u_t \sim p(u_t|u_{t-1}), \tag{4.1}$$

$$\theta_t = \theta_{t-1} + c(u_t) + v_t, \tag{4.2}$$

$$x_t = x_{t-1} + m[\cos\theta_t, \sin\theta_t], \tag{4.3}$$

where $u_t \in \{0, 1, 2\}$ represents the time varying state of the object; that is continuing straight, making a 0.3 radian left turn or making a 0.3 radian right turn, respectively. $c(u_t)$ represents the angle of turn in radians. The angle of the motion is represented by θ_t, which has a Gaussian innovation noise v_t. The object's position is x_t and has a constant velocity m.

The observation equation for node v with position g_v is

$$r_t^v = \max(\|x_t - g_v\|(1 + s_t), 0), \tag{4.4}$$

where s_t is a zero-mean Gaussian noise with variance $\sigma_s^2 = 0.02$. This states that the accuracy of the sensor measurements is proportional to the relative position of the object in reference to the sensor. This property can be found in many ultrasonic sensors, as exemplified in [59, 60].

The object's initial position is determined by using a Gaussian distribution centred at $[2, 2]$ with diagonal covariance entries set to 1. The initial angle of the object's motion is also determined by using a Gaussian centred at $\pi/4$

with a variance of 0.01. The object moves at a constant velocity of $m = 0.5$ metres/second and the innovation noise has a variance of 0.001. The state probability matrix is

$$p(u_t|u_{t-1}) = \begin{pmatrix} 0.75 & 0.65 & 0.65 \\ 0.125 & 0.3 & 0.05 \\ 0.125 & 0.05 & 0.3 \end{pmatrix}.$$

The sensor network architecture we consider consists of two different types of nodes. Class B nodes (or sensors), when active, are responsible for taking measurements related to the object's position. Class A nodes, which have more processing power and energy, are responsible for running the particle filters to track the object. Each class A node is associated with a set of "children" class B sensors and collects data from these children, if any are active, equally spaced out from each other. There are 128 uniformly distributed class B sensors dispersed to take distance measurements. Each of these class B sensors is associated with exactly one class A node, the one which is located closest to its position. Figure 4.3 shows an example of one such setup. For the purposes of this thesis, we will term class A nodes as "nodes", while referring to class B nodes as "sensors".

Under these simulation conditions, there are eight active sensors taking measurements at every time step, for a duration of 500 time steps. The choice of the eight sensors used in the network is determined by selecting the closest sensors from the predicted object's position. This prediction is made one time step before of taking the measurement.

The experimentation we conducted uses $S = 20$ different realizations of a sensor field and object path. In each study, the algorithm was applied with $REP = 5$ different random seeds for initialization. Trials were conducted for $N = \{35, 50, 100, 200, 500, 1000\}$ particles. Recall that the number of particles refers to the number of state trajectories (or particles) that are candidate representations of the system state. From this setup, there were a total of 100 trials performed for each of the particle sets used.

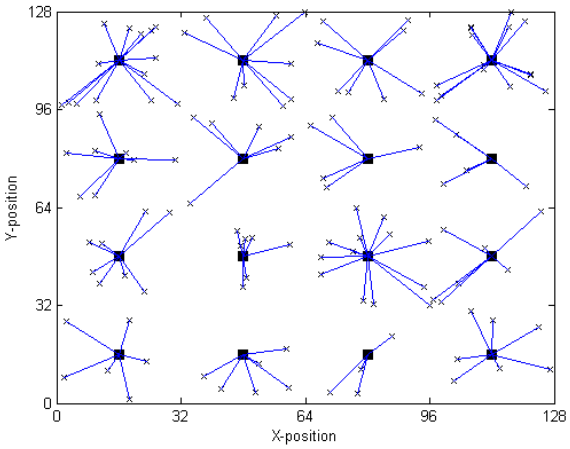


Fig. 4.3 Sample tracking area. The x's are class B sensors, each which are associated with a class A node, denoted as a square.

4.5.2 Hardware Implementation

To construct such a system, only a few components are required. Class A nodes can be constructed using a small processor chip and a RF wireless transmitter. Class B sensors can be constructed with a small processor, a wireless transmitter and a sensor (e.g. ultrasonic range finders). Class A nodes tend to have a more powerful processing chip than the class B sensors since they must maintain a particle filter. The processor for the class B sensors is only responsible for transmission of measurement data that it collects and therefore does not need a powerful chip.

Communication from both types of classes is performed using different frequencies. The number of frequencies used will be enough to ensure that there is no interference with one another.

4.6 Tracking Accuracy

To determine the effectiveness of the tracking algorithm, we need to analyze the experimental test results. We make this analysis in three phases. The first will look at the tracking results of the case where a centralized particle filter is used in this specific simulation scenario. In the centralized particle filter, all sensor measurements are sent to a common node, where a particle filter is maintained. The second analysis will then look into results obtained from

performing the distributed approach only using the described quantization and encoding procedure. That is, data from the various active class A nodes will be exchanged at every time step. This is essentially the algorithm using a vector length of one. Following that, a complete analysis of the algorithm will be conducted which includes the vectorization procedure.

The tracking accuracy is measured in terms of the mean-squared error (MSE) values. The MSE is computed by finding the mean-square error between the true object position and the particle filter-based estimate of the object's position. The value is then averaged over a duration of T seconds, with S realizations of REP trials of this algorithm. Mathematically, the MSE is defined as

$$MSE = \frac{1}{S}\sum_{s=1}^{S}\left[\frac{1}{REP}\sum_{r=1}^{REP}\left[\frac{1}{T}\sum_{t=1}^{T}\|x_{t,s} - \hat{x}_{t,r,s}\|^2\right]\right] \tag{4.5}$$

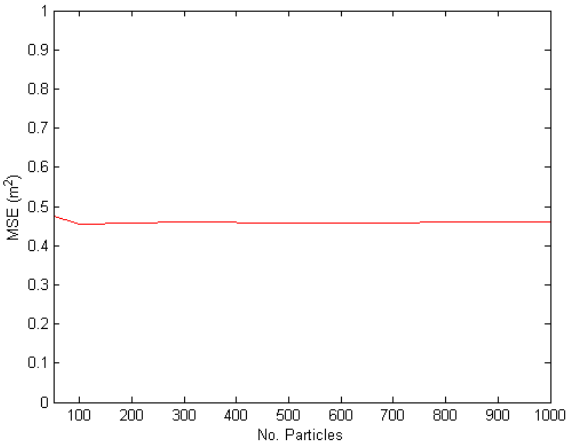
where $x_{t,s}$ is the vector position of the object at time t and $\hat{x}_{t,r,s}$ represents the particle filter estimate.

4.6.1 Centralized Particle Filter

In our centralized network, we assume lossless encoding was used in order to preserve the exact data collected by the sensor nodes. Figure 4.4 shows the MSE results of our trials in the centralized algorithm. In the range of particles that was conducted, the MSE values averaged around $0.45m^2$. The MSE value of 0.45 indicates that the average error in the measurement is approximately a step size of the object movement. The minor variation of these MSE values as shown in Figure 4.4 (b) can be accounted for by the randomness attribute of the particle filter. As a result, for the remainder of the comparison of our distributed algorithm, we will use this MSE value as a basis of comparison to determine the effectiveness of the distributed particle filter algorithm.

4.6.2 Distributed Particle Filter with Quantization

Next, we observe the results when we use the distributed particle filter as described (without vectorization), looking at the experimental results of the global particle filtering estimates. Figure 4.5 shows the experimental results of the tracking algorithm using the various numbers of quantization bins. Recall



(a)

No. of Particles	35	50	100	200	300	500	1000
MSE (m^2)	0.5075	0.4745	0.4539	0.4569	0.4598	0.4564	0.4612

(b)

Fig. 4.4 Plot and Table of MSE of the centralized tracking algorithm trial runs using various numbers of particles.

that the number of quantization bins refers to the number of regions that the current local particle space is divided into in order to create a histogram of particle distances for measurement encoding. Accuracy similar to the centralization scheme is achieved when at least 200 particles are used in the filter with 8, 16, or 32 quantization bins. At least 500 particles are required for the 4 quantization bins case.

Although it can be observed that in certain cases, the distributed algorithm produces better MSE results than the centralized case, it is erroneous to conclude that the distributed particle filter is a more accurate algorithm. The reason the distributed case can produce better results is due to the fact that the particle filter uses many random numbers. Depending on the numbers generated, the results can be affected for better or for worse.

The discrepancy between the MSE results of the distributed and centralized cases can be attributed to the loss in accuracy due to the quantization stage. Quantization effectively rounds the measurements values to the nearest bin representation. As a result, the more quantization bins that are used, the more accurate the encoded measurement values become because each bin will

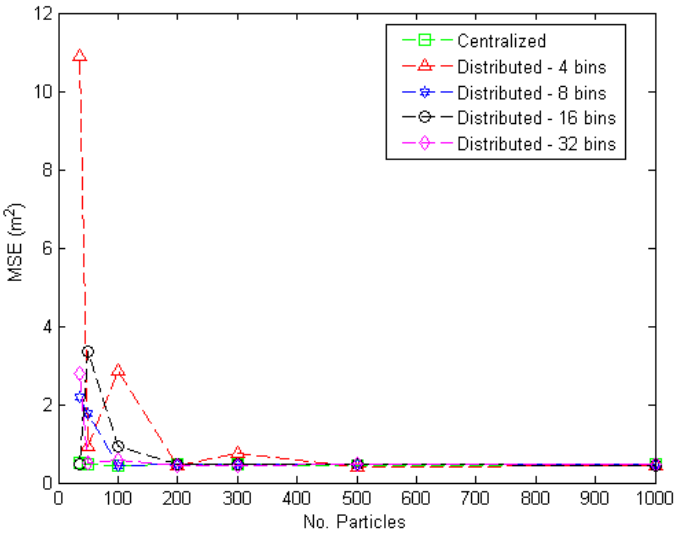


Fig. 4.5 Comparison of MSE results for the centralized case and various distributed cases with the proposed quantization and encoding scheme. Reasonable accuracy is obtained for when at least 200 particles for 8, 16, 32 quantization bins tested and at least 500 particles for 4 quantization bins.

represent a smaller region. As the number of bins increases toward infinity, the MSE results should converge towards those of the centralized case. This is because having an infinite number of bins is effectively equal to not quantization the measurements. Unfortunately, the results of Figure 4.5 do not confirm this statement at low particle numbers. This is probably due to the fact that since the MSE is an average value, the results in the plots do not necessarily give an accurate representation on how each individual trial performed. A single trial might have completely lost track of the object, therefore producing an extremely high MSE value. This can alter the MSE to give a false sense of the performance of individual trials. Therefore, it may be more beneficial to evaluate how successful the algorithm performs every time it is executed.

For the purposes of these experimentations, we define a lost track to be the trials which have a mean square error (MSE) of 15 or greater. This number is rather arbitrary, but does for the most part give an accurate representation of a track being lost. In some of the trial runs, the object's track may have been considered temporarily lost meaning that the filter loses track of the object for a brief moment in time before relocating it. Here, we will define a

temporarily lost track as inaccurately tracking the position of the object for a short period of time. Mathematically, we define temporarily lost tracks as trial runs which have a MSE $\in [5, 15)$. Again, for most cases, this range provides a good indication whether objects have been temporarily lost. For the purpose of illustrating how well the algorithm works, these definitions are adequate enough.

No. Particles	No. bins				Centralized
	4	8	16	32	
35	3/1	3/0	0/0	3/1	0/0
50	1/0	3/0	1/0	0/0	0/0
100	1/0	0/0	1/0	0/1	0/0
200	0/0	0/0	0/0	0/0	0/0
300	1/0	0/0	0/0	0/0	0/0
500	0/0	0/0	0/0	0/0	0/0
1000	0/0	0/0	0/0	0/0	0/0

Table 4.1 Percentage of lost tracks/temporarily lost tracks in the distributed particle filter algorithm with the proposed quantization and encoding scheme using various numbers of quantization bins and particles.

Table 4.1 shows the percentage of lost tracks and temporarily lost tracks that were obtained in our trial runs using various sets of particles and quantization bins. It is with this table that the spikes in Figure 4.5 can be better explained. The points on the figure which represent trial runs which average greater than approximately 0.5 MSE contain runs with either lost track(s) or temporarily lost track(s). Unfortunately, the height of the spikes does not necessarily indicate the frequency of the number of lost runs. For example, the distributed case with 4 quantization bins at 100 particles has a spike to about an MSE of 3, but this is accounted for by only 1 lost track. At 50 particles in the distributed case with 8 quantization bins, there are 3 lost tracks, yet the MSE value is under 2. It should be noted that a lost track can have varied MSE results as some lost tracks can have values that are extremely high (> 1000) and some can have lower values such as 20.

Before jumping to conclusions whether MSE values of 2, 5 or even 10 are acceptable, one needs to consider the practical use of this object tracking algorithm. Depending on its function, it may be acceptable to allow slightly

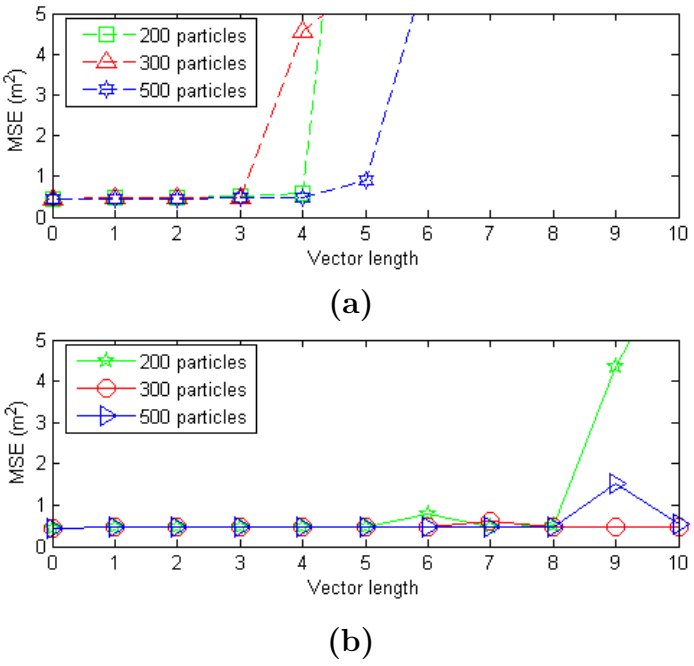


Fig. 4.6 MSE results of the parallel distributed particle filter tracking algorithm using various numbers of particles (a) MSE results using 8 quantization bins (b) MSE results using 16 quantization bins. At 16 quantization bins, the algorithm is effective for vector lengths up to 8 for 200 particles and 10 for 300 particles.

higher MSE values, such as some tracks where the object is temporarily lost.

4.6.3 Parallel Distributed Particle Filter

Next, we analyze the effect the vectorization scheme has on the tracking algorithm. Recall that the vectorization scheme saves the quantized measurement values for a few time steps before transmitting them to all other nodes. This means that the same set of active sensors will have to take measurements for the entire duration of the vectorization process. As a result, it is not necessarily true the set of active sensors are always the closest sensors to the object.

Figure 4.6 shows the MSE values of the trial runs using a various numbers of particles with 8 (subfigure (a)) and 16 (subfigure (b)) quantization bins. It is obvious that the algorithm performs better using 16 quantization bins. With this bin scenario, reasonably accurate results are obtained for vector lengths up to at least 8 for the various number of particle sets used in the experiment.

Vector Length	No. Particles/Bins					
	200/8	200/16	300/8	300/16	500/8	500/16
1	0/0	0/0	0/0	0/0	0/0	0/0
2	0/0	0/0	0/0	0/0	0/0	0/0
3	0/0	0/0	0/0	0/0	0/0	0/0
4	0/1	0/0	1/0	0/0	0/0	0/0
5	3/0	0/0	3/0	0/0	2/0	0/0
6	3/0	1/1	2/1	1/1	0/0	0/0
7	4/2	0/0	3/1	0/1	2/0	0/0
8	6/2	0/0	6/0	0/0	6/1	0/0
9	8/1	1/0	10/0	0/0	8/0	1/0
10	8/1	1/0	4/0	0/0	5/4	0/0

Table 4.2 Number of lost tracks/temporarily lost tracks for the parallel distributed particle filter trial runs using various quantization bins and vector lengths.

In fact, when using 300 particles, accurate results were obtained for vector lengths up to 10. There is also very minimal loss of objects for all trials with the various particle sets used at this quantization level of 16 bins. The lost track results are outlined in Table 4.2.

The MSE results of our PDPF algorithm are comparable to the centralized case, which is indicated on the plots as having vector length of 0. Vector length 1 is just quantization/encoding without any vectorization. These results indicate that in an application where an estimate is not needed at every time step, vectorization can be used without incurring a substantial penalty in tracking accuracy. Of course, these results are specific to the reported tracking scenario. In the reported simulation, the set of active class B sensors are only changed after communication is exchanged between class A nodes. This means that measurements become more unreliable as the vector length increases, because the object is likely to be further away from the sensors.

When comparing the relative importance between the number of particles and the number of quantization bins used in the filter, it seems apparent that varying the number of particles has a greater impact on the results. Figure 4.7 displays the tracking error results for a specific tracking scenario using 500 particles and 16 quantization bins. An analysis is made between the centralized particle filter, the global distributed particle filter and the local

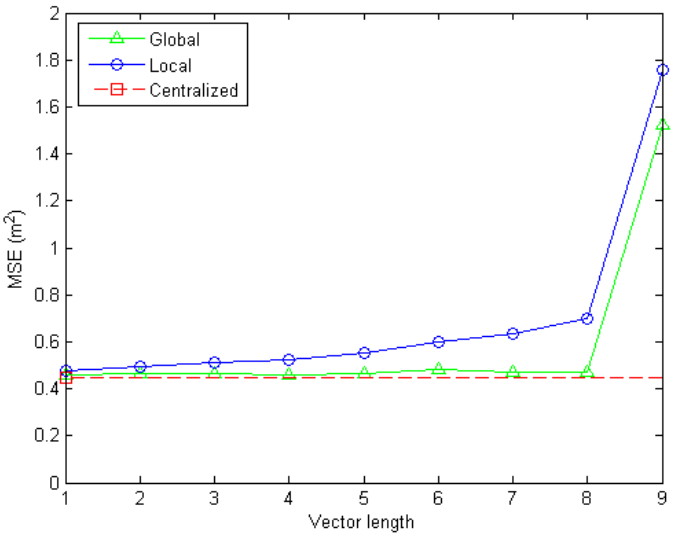


Fig. 4.7 Comparison of MSE results for the global and local parallel distributed particle filter, and for the centralized particle filter for various vector lengths using 500 particles and 16 quantization bins. Under these conditionals, the global particle filter produces very similar results to the centralized case for vector length up to 8.

distributed particle filter estimates. Note that for comparative purposes, the local MSE values are formed by computing the average MSE values of all active nodes. In the simulations, the MSE of local estimates are only slightly larger than the global estimates. This can be accounted for by the fact that each of those active local particle filter utilized less than eight sensor node, thus getting an estimate which is not as accurate as compared to the other two. Nevertheless, should an object state estimation be required during the period when a vector's length time has not elapsed, a local estimate can be still be used as a temporary value without too much loss in accuracy.

4.7 Communication Costs

The results of the previous section indicate that our distributed algorithm produced comparable tracking accuracy to the centralized scheme. Recall though, that the goal of our algorithm is to minimize the communication cost while maintaining reasonable accuracy. Now that we have determined that

the scheme can produce relatively accurate tracking results for various simulated scenarios, we next look at the communication costs needed to run this algorithm.

The communication costs are evaluated in terms of the number of bits transmitted into the network by active class A nodes. We neglect the cost required to obtain data from class B sensors because this cost is equal for both the centralized and distributed case. Note that in our analysis, we neglect the costs associated with relaying data because the average number of relays for both the centralized and distributed case in our experimental setup is similar. In fact in our specific sensor network scenario, it is reasonable to assume that the transmitted data can be broadcasted, and therefore not requiring any relaying of information by any nodes.

4.7.1 Centralized Particle Filter

In the centralized scheme, the number of bits transmitted into the network by active class A nodes is determined by the size of the packets that are sent out. The overhead for these packets is four bytes long - one byte for each of the source and destination addresses, one byte to describe how to read the payload (e.g. determine how many measurements are encoded) and one byte for the CRC. On average, it was determined that 2.5 class A nodes are active at any given moment. There are a total of eight measurement data transmitted, each represented by 16 bits. As a result, the average number of bits transmitted per time step in this scenario is 208 bits.

4.7.2 Distributed Particle Filter with Quantization

The overhead cost associated with the distributed particle filter with quantization is different as compared to the centralized case. An extra byte is required to help describe the variable length of the quantized/encoded measurement values. The header structure therefore consists of a one-byte source address, a one-byte destination address, a two-byte field to describe how to read the payload and quantized value, and finally a one-byte CRC field. Figure 4.8 (a) shows the effectiveness of the quantization/encoding scheme in terms of the average number of transmitted bits sent by active class A nodes into the network at any given time step. The figure shows that the more quantization

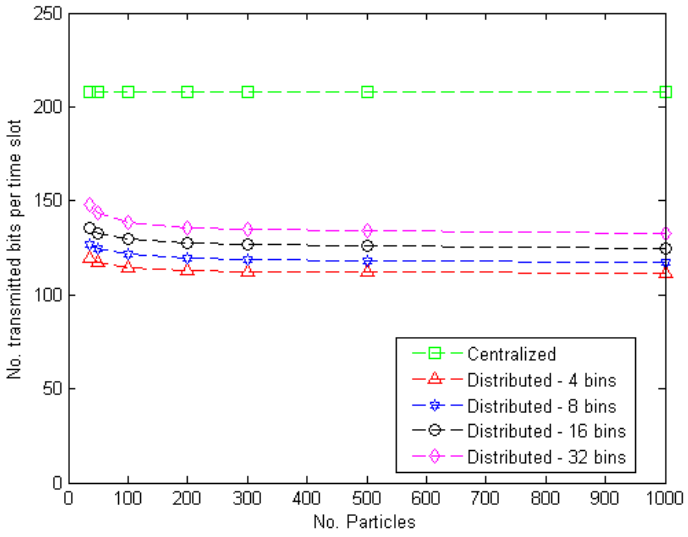
bins that are used, the greater the number of bits is transmitted. The use of a larger number of quantization bins appears to increase the average cost by about one bit per measurement, as shown in Figure 4.8 (b). This can be explained by considering the Huffman tree size, since increasing to the next bin level (i.e. increasing the value to the next power of 2) effectively adds one more level to the Huffman binary tree. Figure 4.8 (c) shows the change of transmitted bits as the number of particles is varied at 16 quantization bins. As the number of particles used increases, a smaller number of bits is required for transmission. We can speculate that this is due to the fact that as more particles are used, a more accurate set of particles is obtained. This in turn translates to a Huffman encoding that uses a smaller number of bits since the measurement data is most likely in a bin of high density.

4.7.3 Parallel Distributed Particle Filter

When vectorization is considered, the packet overhead cost again slightly differs from distributed particle filter case since an additional one byte overhead is needed to describe the length of the vector of the packet. This means that the packet overhead will consist of a six-byte header – one byte source address, one byte destination address, three-byte field to help decode the payload, and finally a byte for the CRC field.

We observe in Figure 4.9 with the condition of 500 particles and 16 quantization bins that vectorization generates a substantial savings in the average number of bits transmitted. As the vector length increases, the bits savings also increases. For example at vector length 8, the savings increases to approximately at 75% as compared to the centralized case. This is due to the fact that communication between class A nodes occurs only once every period T. With this period of T, there are $b(T-1)$ packet overhead bits saved, where b is the number of bits per header. Effectively, we are saying that only one header is needed for the entire vector length.

In the centralized case, we assume the bulk of communication required in the network is from each individual class A node to the central fusion node. If there are m active class A node that transmit n quantized measurements bits, the communication cost is $O(m(n+b))$ per time step. In the distributed case with quantization and vectorization, the required number of transmitted



(a)

	4 bins	8 bins	16 bins	32 bins
500 particles	112.0	118.1	125.8	133.8

(b)

	No. of Particles						
	35	50	100	200	300	500	1000
16 bins	135.4	133.0	129.5	127.4	126.6	125.8	125.0

(c)

Fig. 4.8 (a) Plot of the average number of transmitted bits per time slot in the network from class A nodes using the distributed particle filter using only the proposed quantization and encoding scheme. (b) Table of average number of bits transmitted per time slot at 500 particles with varying number of quantization bins. (c) Table of average number of bits transmitted per time slot at 16 quantization bins with varying number of particles.

bits is $O(m(n + b/T))$. As T grows, the number of transmitted bits tends to $O(mn)$.

4.8 Computational Costs

The parallel distributed particle filtering algorithm presented here is inherently more computationally expensive than the centralized approach because a particle filter must be maintained at multiple nodes. The maintenance of

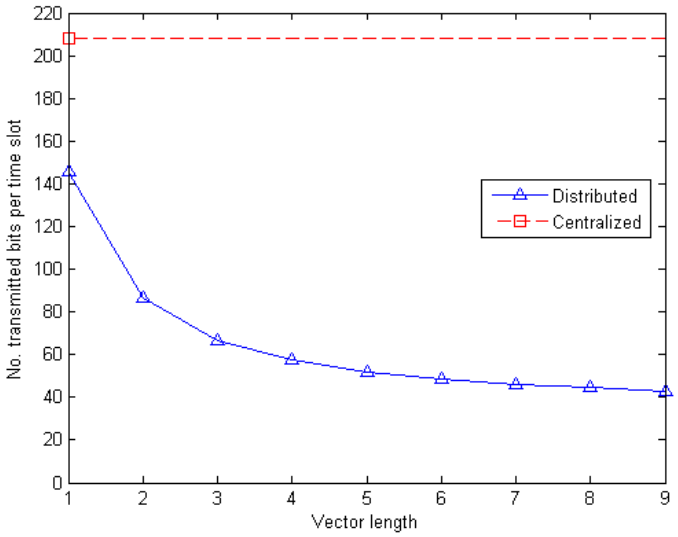


Fig. 4.9 Plot of the average number of transmitted bits per time slot in the network from class A nodes using the parallel distributed particle filter algorithm for various vector lengths for 500 particles and 16 quantization bins. The centralized scheme assumes 16-bit values are sent.

each of these particle filters also requires more computation, because there is the extra quantization and encoding procedure. In our experiments, the distributed particle filter at any class A node required roughly between $1.25 - 1.5$ times the computation of a standard particle filter. This includes the encoding procedure, which required approximately $\frac{1}{4}$ of the time of the particle filter.

Overall, we can assess that the computational costs of the PDPF is higher than that of a centralized algorithm, but is worthwhile since the communication costs have been substantially lowered. It should also be noted that energy costs related to computation is significantly less than the costs associated with communicating information in a network. In the context of a sensor network, this is very important as the network lifetime can be increased with the modifications presented.

4.9 Comparison with Other Schemes

We saw that the distributed particle filter scheme requires less energy than a centralized particle filtering approach to perform their respective particle

filtering operations. Not only is our distributed particle filter scheme more favourable than the centralized particle filter in this respects, but the algorithm is also communicationally more advantageous compared with other schemes proposed in literature. For example, in the distributed method with a changing leader node, as described in [46], their algorithm has a reduced communication cost compared to a centralized particle filtering algorithm as the leader node will poll a set of fairly local sensors to the object. This scheme is quite comparable to the information exchange required in our vectorized particle filtering scheme. However, the advantages of this scheme are quickly eliminated when particle information is exchanged to the new node.

In the query-response distributed method in [49], each query and each response requires the communication of p particles. This means at any given time, the $O(q \cdot b_m)$ bits are transmitted, where q is the number total number of query-response made and b_m is the number of bits required to send the p particle information. We can assume that $m \ll q$ since the query-response communication requires communications between pairs of nodes while the class A nodes broadcasts their information to other nodes. The number of bits that must be used to send the p particles would most likely be greater than the length of the vector. This consequently indicates our proposed vectorized algorithm requires lower bandwidth, since a sufficient number of particles are needed to represent the state.

The scheme of [48] shares at each time step the parameters of a Gaussian mixture approximation of the particle set. This requires $O(cm(n_g + b))$ bits, where c represents the number of Gaussians in the mixture, and n_g represents the number of bits required to encode the mixture parameters. Whether this approach involves a smaller communication overhead depends on whether state dimension is small compared to data dimension (per time step) and whether the Gaussian mixture approximation is reasonably efficient. In our simulation scenario, the dimensions are comparable (8 measurements, 4 state components), so transmitting compressed, encoded data is much more efficient. However, if the data were images from a camera, then modelling the distribution of the state using a mixture of Gaussians or another approach would probably be preferable to the data encoding procedure we have proposed here, because the data dimension is so large.

4.10 Variation: PDPF with One Active Class A Node

As a variation to our PDPF algorithm, we can modify our algorithm to further minimize the communication costs if only one class A node is active at any given moment. Recall that in our proposed PDPF algorithm, the closest class B sensor to the object's project position is activated to take measurements. The class A nodes associated with these selected class B sensor are consequently activated. It is possible that three or four of the class A nodes are activate at any given time. This can be a waste of energy and resources. Our proposal here is to activate the eight associated class B sensor belonging to that one particular class A node that is activated so that there isn't the need to use multiple class A nodes.

4.10.1 Algorithm

In this setup, there is no longer a need to maintain a local and global particle filter since measurements are only taken at one local node at each time step. When the measurement data is quantized, encoded, and vectorized, this information can be distributed to all other nodes to update their filtering information. A high level algorithm is shown in Figure 4.10.

1. Initialization, $t = 0$
   - Initialize the particle filter of each sensor $k = 1, ..., K$ by equating the random seeds to ensure they all match.
   - For each sensor $k = 1, ..., K$
     - For $i = 1, ..., N$ particles, sample $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)$.

2. Particle filtering process at leader node:

   - Set $t \leftarrow t+1$.
   - For the length of the vector $v = 1, ..., V$ and closest sensor k:
     (a) Quantization and encoding:
       – For $i = 1, .., N$, sample $\tilde{\mathbf{x}}_t^{(i)} \sim p(\mathbf{x}_t | \mathbf{x}_{0:t-1}^{(i)})$.
       – Calculate expected values of measurements $g_t^{(i)} = E(y_t^{(k)} | \tilde{\mathbf{x}}_t^{(i)})$.
       – Create a histogram of the $g_t^{(i)}$ values with h equal bins encompassing the range of values $[\min(\mathbf{g}_t), \max(\mathbf{g}_t)]$, where $\mathbf{g}_t = [g_t^{(i)}, i = 1, ..., N]$.
       – Use the histogram to form a Huffman tree H_t^k and encode the quantized measurements $\widetilde{\mathbf{y}}_t^k$.
     (b) Particle Filter
       i. Importance Sampling
         – For $i = 1, ..., N$, sample $\tilde{\mathbf{x}}_t^{(i)} \sim \pi(\mathbf{x}_t | \mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}'_{0:t})$, and set $\tilde{\mathbf{x}}_{0:t}^{(i)} = (\mathbf{x}_{0:t-1}^{(i)}, \tilde{\mathbf{x}}_t^{(i)})$.
         – For $i = 1, ..., N$, evaluate the (approximate) importance weights $\tilde{w}_t^{(i)} = \frac{p(\mathbf{y}'_t | \tilde{\mathbf{x}}_t^{(i)}) p(\tilde{\mathbf{x}}_t^{(i)} | \mathbf{x}_{t-1}^{(i)})}{\pi(\tilde{\mathbf{x}}_t^{(i)} | \mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}'_{0:t})}$.
         – Normalize the importance weights.
       ii. Estimation
         – Form an estimate of the object's state using a standard particle filter acting on all measurements obtained with their associated weights.
       iii. Selection
         – For $k = 1, ..., K$, resample with replacement particles $\{\mathbf{x}_{0:t}^{(i)}; i = 1, ..., N\}$ from the set $\{\tilde{\mathbf{x}}_{0:t}^{(i)}; i = 1, ..., N\}$ according to importance weights.

3. Network Communication:

   - For each $k = \{1, ..., K\}$, send the vectorized measurements $(\mathbf{y}_{t-V:t}^k)' = \{(\mathbf{y}_{t-V}^k)', (\mathbf{y}_{t-V+1}^k)', ..., (\mathbf{y}_t^k)'\}$ to other *K-1* sensors.
   - Each node k can now decode the data and compute the exact estimate using Step 3.

4. Go to step 2.

Fig. 4.10 High-level algorithm description of the distributed particle filter that uses the leader node approach.

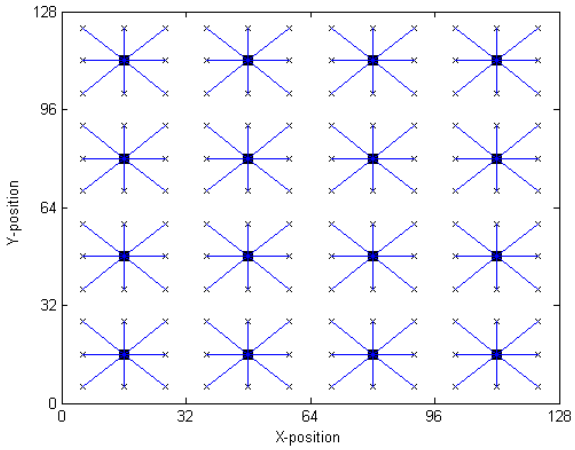


Fig. 4.11 Sample tracking area using a prearranged class B placement. The x's are class B sensors which are associated with the class A nodes, denoted as squares.

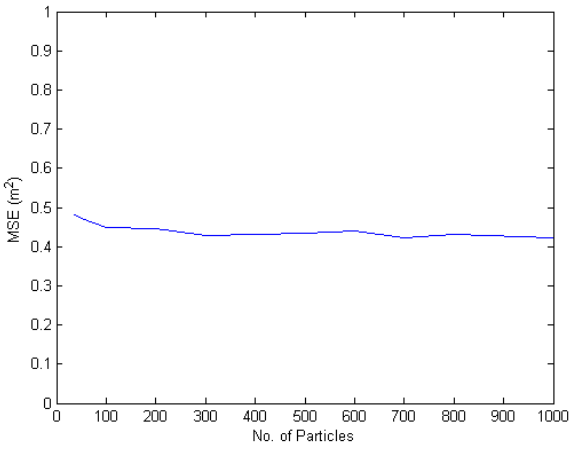
4.10.2 Simulations

When the class B sensors were distributed in a random fashion, (i.e. the same conditions as were used in the PDPF), this new modified algorithm did not give comparable results. This is due to the fact that since each class B sensor is associated with the closest class A node, we cannot guarantee that exactly eight sensors are associated with each class A node. Since nodes had access to fewer measurements, the tracking performance was poorer. In order to perform a fair comparison, we adopt a sensor configuration with all class B arranged uniformly to provide the best coverage of the region, as previously seen in Figure 4.11. With this configuration, eight sensors are associated with each class A node and eight measurements are available at each time step.

Tracking Accuracy

Again, we will perform an analysis on this algorithm similar to that with the PDPF. However, since the node configuration is now different, we need to recompute the MSE values of the centralized case. The results are shown in Figure 4.12. Compared to the random configuration of class B sensors, this arrangement has slightly lower MSE values. This is most likely accounted for by the fact that every position in the sensor network has very similar

sensor coverage which provides for a more consistent result. The consistency translates into slightly better MSE values because there is less chance that a sensor with less accurate readings (i.e. in a position farther away from the object) is used to make measurements.



(a)

No. of Particles	35	50	100	200	300	500	1000
MSE (m^2)	0.4805	0.4717	0.4502	0.4470	0.4282	0.4348	0.4240

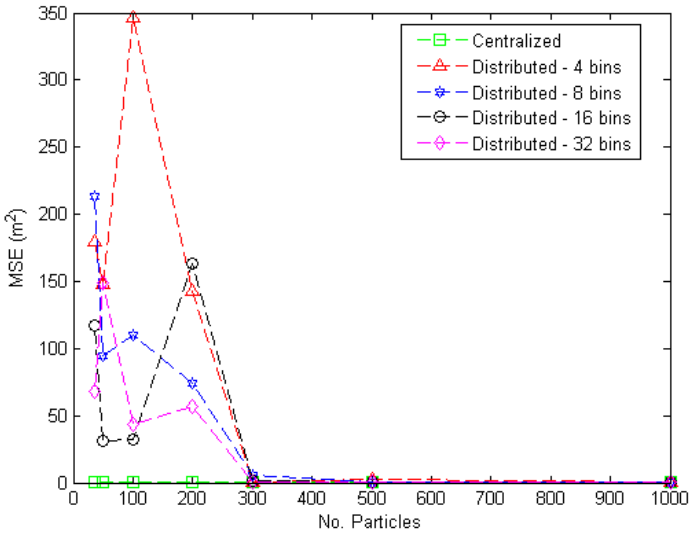
(b)

Fig. 4.12 Plot and Table of MSE of the centralized tracking algorithm trial runs using various numbers of particles using the prearranged class B sensor placement.

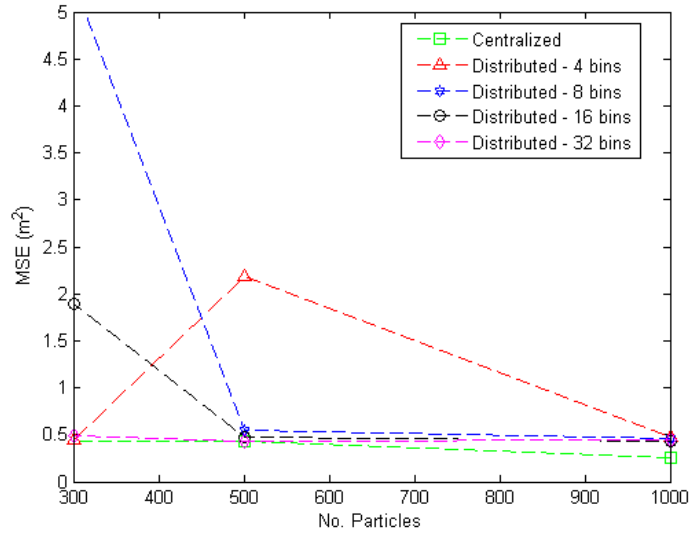
Distributed Particle Filter with One Active Node

When comparing the one active class A node distributed algorithm with quantization and encoding scheme with that of a centralized network, we note erratic results when less than 300 particles are used. With at least 500 particles, the algorithm was able to produce very comparable results to that of the centralized case. Figure 4.13 shows the results of the investigation.

The percentage of lost tracks (Table 4.3) was also significantly higher when the number of particles is low. However, when at least 500 particles are used, very good results are produced.



(a)



(b)

Fig. 4.13 Comparison of MSE results for the centralized case and the leader node distributed approach with the proposed quantization and encoding using various numbers of quantization levels. Reasonable accuracy is obtained for at least 300 particles when 32 quantization bins are used and at least 500 particles for 4, 8 and 16 quantization bins. (a) Shows the MSE results for the various number of particle sets used. (b) Shows a zoomed in version of plot (a).

No. Particles	No. bins				Centralized
	4	8	16	32	
35	11/5	10/5	8/3	10/2	0/0
50	8/4	8/6	8/7	10/4	0/0
100	10/2	9/5	9/4	8/5	0/0
200	8/2	9/3	8/7	8/4	0/0
300	0/0	1/0	1/0	0/0	0/0
500	1/0	0/1	0/0	0/0	0/0
1000	0/0	0/0	0/0	0/0	0/0

Table 4.3 Percentage of lost tracks/temporarily lost tracks in the leader node distributed approach with the proposed quantization and encoding scheme using various numbers of bins and particles.

Vectorized Distributed Particle Filter with One Active Node

When the vectorization scheme is added to the distributed algorithm, results indicate that vector lengths of at least 9 can be used with reasonable accuracy with 500 particles and 8 bins or higher. With an increase in vector length, the general trend is for the MSE value to rise, albeit not substantially in the range of the plot shown. The increase in MSE could be accounted by the object moving away from the sensor nodes, for the most part. This movement results in sensor measurements having a greater noise factor and thus poorer performance. Table 4.4 shows the number of lost track in this scenario.

In Figure 4.15, the centralized MSE results are compared with the distributed scheme of 500 particles with 8 and 16 quantization bins. The distributed algorithms both produced slightly larger MSE value than the original centralized particle filter in these cases. This is to be expected as the quantization effect can reduce the accuracy level. Despite the discrepancy, the MSE is not substantially different in the case of 16 quantization bins up to vector length of 9.

4.10.3 Communication Costs

The results of the previous section indicate that our distributed algorithm produces comparable tracking accuracy to the centralized scheme. Let us now briefly analyze the communication costs of this one active class A node case.

When considering only the quantization and encoding stage, this distrib-

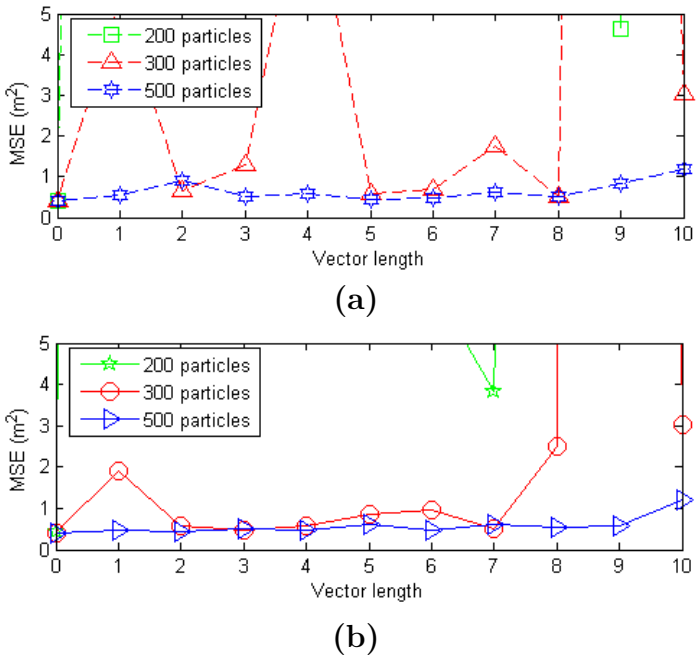


Fig. 4.14 MSE results of the leader node approach with quantization and vectorization using various numbers of particles (a) MSE results using 8 quantization bins (b) MSE results using 16 quantization bins. With 500 particles, the algorithm achieves reasonable results at both 8 and 16 quantization bins.

uted algorithm compares to the centralized case as shown in Figure 4.16. As expected, there is substantial savings in terms of communication costs, as also seen with the regular PDPF. The difference is that the number of transmitted bits per time slot in the PDPF was slightly higher as there were on average 2.5 class A nodes active and transmitting at any given time instant. In this case, only one class A node is active and this accounts for the savings.

When we consider the vectorization scheme, we observe in Figure 4.17 that vectorization generates a substantial savings in the average number of bits transmitted, similar to that displayed in the original PDPF but lower in value due to only one class A node being active at any given time.

4.10.4 Computational Costs

The algorithm here will have similar computational costs to that of the PDPF. The main saving is due to the fact that there is no need to maintain two particle

Vector Length	No. Particles/Bins					
	200/8	200/16	300/8	300/16	500/8	500/16
1	9/0	8/2	1/0	1/0	0/0	0/0
2	8/1	7/3	0/1	0/1	0/0	0/0
3	7/2	9/2	1/1	1/0	0/0	0/0
4	8/2	8/3	2/0	0/0	0/1	0/0
5	8/2	8/1	0/1	1/0	0/0	0/1
6	6/3	7/4	0/2	1/0	0/0	0/0
7	6/3	6/5	2/1	0/0	0/1	0/1
8	8/1	8/1	0/0	1/0	0/0	0/0
9	8/4	9/4	8/4	7/4	0/3	0/0
10	9/4	9/5	6/7	6/5	2/0	2/0

Table 4.4 Number of lost tracks/temporarily lost tracks for the leader node approach with quantization and vectorization using various quantization bins and vector lengths.

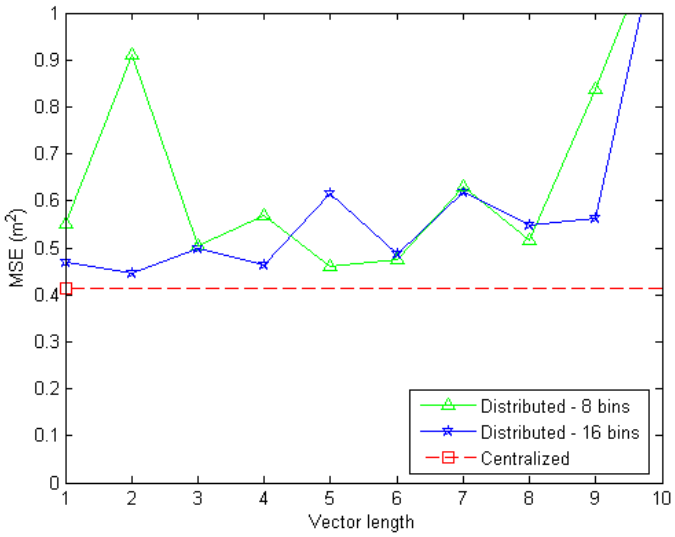
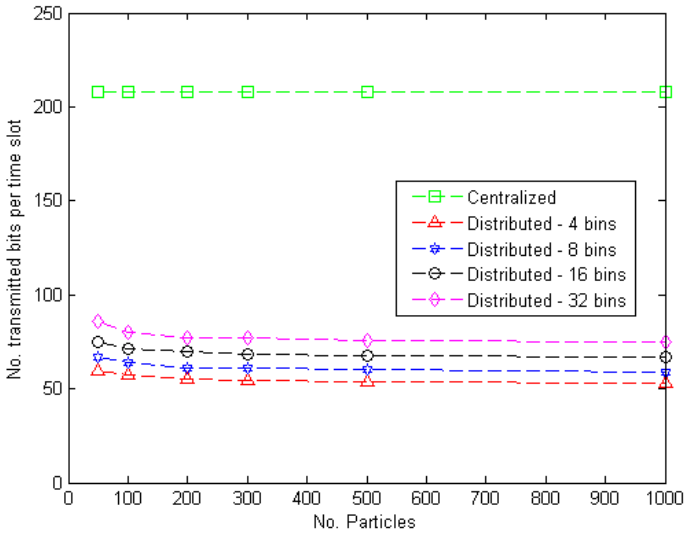


Fig. 4.15 Comparison of MSE results for the leader node approach with quantization and vectorization and the centralized particle filter for various vector lengths using 500 particles with 8 and 16 quantization bins. We notice that the leader node approach produces similar results to that of the centralized algorithm but not as accurate. As the vector length increases for either distributed experiment, the accuracy decreases.



(a)

	4 bins	8 bins	16 bins	32 bins
500 particles	63.7	60.0	67.7	75.7

(b)

	No. of Particles						
	35	50	100	200	300	500	1000
16 bins	78.1	74.6	71.1	69.9	68.6	67.7	66.7

(c)

Fig. 4.16 (a) Plot of the average number of transmitted bits per time slot in the network from class A nodes using the leader node approach with quantization and vectorization. (b) Table of average number of bits transmitted per time slot at 300 particles with varying number of quantization bins. (c) Table of average number of bits transmitted per time slot at 300 particles with varying number of particles.

filters for active nodes. Since only one class A node is active at any given moment, that node uses the one particle filter it has for both the encoding and to update the particle filter. An explicit local particle filter is not needed as a result to handle a subset of measurements.

As a network in general, the computational cost is lowered since only one class A node is maintaining a real-time particle filter, compared to 2.5 nodes in the PDPF case.

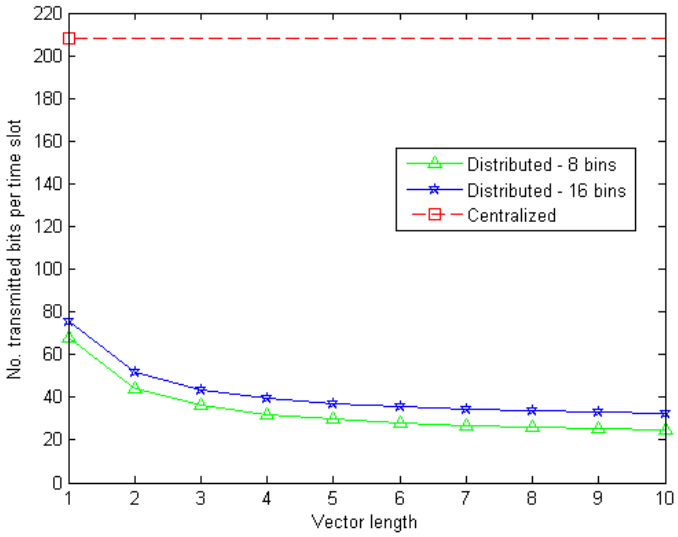


Fig. 4.17 Plot of the average number of transmitted bits per time slot in the network from class A nodes using the leader node approach with quantization and vectorization for various vector lengths for 500 particles and 16 quantization bins. The centralized scheme assumes 16-bit values are sent.

4.11 Overall Assessment of PDPF

Given the simulation results, the parallel distributed particle filter described here significantly reduces the communication costs. The quantization/encoding scheme that was proposed was effective at reducing the costs to send the data to other nodes. The scheme actually reduces the number of bits required to represent one measurement to anywhere from one to five bits in our scenario. The vectorization scheme even provided more savings in communication costs. With a large vector length, savings of 75% or higher could be achieved.

While the global MSE results are slightly higher in value in comparison to a centralized particle filtering scheme, the tradeoff with the communication costs is worthwhile. Communication costs have been dramatically reduced. If a query is made on the location of the object during the intervening time when only local estimates are calculated, this estimate is still adequate enough to give a reasonable accurate result. However, the goal is to use the algorithm when the query rate is equal to the vector length.

The only drawback to this algorithm is that the computation of the algo-

rithm may be slightly higher than that of the centralized case, since multiple particle filters must be computed and maintained. However, because of the energy costs were distributed throughout the network so as not to drain the power of any specific nodes in the sensor network, the concentration of energy consumption was along the path of the object locations throughout the trial. This allows for the network to last longer as the energy of the same nodes are not being consistently drained. Furthermore, costs associated with communications tend to be higher than costs associated with computational work, and therefore it is quite advantageous to have such a scheme.

With the variation of the PDPF having only one active class A node at any given time, the algorithm seems to produce better results than the original PDPF. However, this is only the case if eight class B sensors are associated each class A node. Therefore, it is not really fair to make a direct comparison of the results shown. Nevertheless, it is a viable option to consider depending on the application of the tracking algorithm. The difficulty is ensure that the class B sensors have a distribution as described here. As indicated previously, often the deployment of the nodes and sensors cannot be exactly determined due to the conditions of the region of interest.

The algorithm presented nevertheless produced significant innovation in terms of reducing communication costs and decentralizing the algorithm that makes it worthwhile and promising distributed algorithm. The two key novelties of our quantization/encoding method and vectorization scheme provided this savings. Note that the reporting results are specific to the networking architecture and simulation conditions presented, however the overall concept should still perform well in other similar context.

Chapter 5

Locally Distributed Particle Filter

In the previous chapter, we proposed a method whereby particle filters reside on multiple nodes, each maintaining a common particle filter in order to track an object. However, despite the success, the parallel distributed particle filter's major burdens come in the form of data fusion of the measurements and the computational costs within each mote. Data fusion is necessary since the basic particle filter algorithm requires the sensor measurements and particle filter information in order to track an object.

In this chapter, we examine another form of a distributed algorithm using an alternative method described in Chapter 3 to minimize communication costs. The proposal we consider is to maintain subsets of particles at multiple nodes. This can lead to allowing a particle filter to reside on each node, functioning independently by using only its own particles and sensor measurements. In such a scheme, the high computational costs associated with processing all of the particles is distributed amongst many nodes, therefore reducing the energy burden of a single mote. To help keep costs even further down, a scheme should be devise for the filter to only use local sensor measurements since a particle filter can run independently of other nodes. With this latter step, there would no longer be a need to collect all sensor measurements at one location. The sensor network can therefore be active for a longer period of time, increasing its lifetime as compared to a setup using a centralized scheme. In this chapter, we explore the issue of distributing particles amongst

several nodes in order to devise a detailed particle filtering scheme which can reduce the costs of running a filter in a sensor network.

5.1 Proposed Solution

In order to devise a particle filtering scheme that would run based on the distribution of particle subsets amongst several nodes, we need to analyze the steps of the particle filter which require a collection of particle filter information and/or measurement data at one location. According to the four step procedure of the particle filter, as described in Section 2.1.4, steps 2, 3 and 4 have to be reviewed.

In the Importance Sampling step (step 2), the *update* stage requires the availability of all particle filter information and measurements values in the networks to evaluate the weight of each particle. The weight is computed using the likelihood of that particle state being generated from the measurement data collected in the system. In the situation where we wish to avoid the fusion of measurement data at one node to perform this calculate, the likelihood function can be approximated using a pseudolikelihood function [61]. The idea to the pseudolikelihood is that a function acting like the likelihood can be obtained without the explicit knowledge of the entire measurement set of the system. Instead, an implicit knowledge of the measurement values of non-local measurements could be utilized. In order for this to work, the networking conditions and system models surrounding the object tracking scenario in the sensor network must right. As a result, the update step can be modified to run independently of the non-local measurement if a suitable likelihood function can be obtained.

In step 3, object estimation is performed. The object estimation value is computed by using a weighted sum of all particle values. While the particles can be independently maintained using local particle filters at each node, the entire set of particles in the network is still required to formulate an estimate. In order for this to occur, as discussed in Chapter 3, there are a few methods that can be employed. For example, a tree-structure can be used to perform a weighted summation of the particles values and weights. Alternatively, a global transceiver can also be used to perform this function. The best solution for this task, however, depends on the exact networking conditions.

In step 4, resampling is needed to ensure that the variance of the particle sets is not high. In this setup, as the subsets of particles are maintained at individual nodes, a local resampling procedure should be invoked to keep the communication costs low. Local resampling is performed using only the local particles and their associated weights and therefore has no added transmission costs. This is sufficient in most cases since the particle sets are maintained independently of other nodes. However, a global resampling step may be needed periodically to ensure the entire set of particles in the network does not have a high variance. The set of particles in the network are collectively used to compute the object's state estimation and therefore it is also important that the variance of the entire particle set is not high in order to properly track the object. Since each local resampling procedure modifies the weights of its local particles, their normalization constants have to be considered in order to perform a global resampling. Taking into account the normalization constants used in each individual node will allow the proper weights of the particles in relation to other particles maintained at other nodes to be computed. Thus, global resampling requires the aggregation of the normalization factors and particle weights from each individual node. Aggregation can be performed using any of the methodology proposed for estimation, since the general steps for both procedures are similar.

In summary, we have described a framework for creating a distributed particle filtering algorithm based on maintaining subsets of particles at individual nodes to reduce communication costs due to the exchange of particle information. We now outline a tracking algorithm in a sensor network based on the proposed modifications to the particle filter.

5.2 Algorithm

In the initialization stage at time t_0, all motes are activated. Each mote is allotted the same number of particles with each particle given the mote's position, x_0, as its initial value. The weights of all particles are set to one.

At each time step, every active node then propagates their particles. As each mote is responsible for a specific range of state values (i.e. specific particle positions) in the sensor network, the mote determines whether any of their propagated particles have states (i.e. position values) that has left their

range of responsibility. If so, the mote responsible for the range in question is activated and the particle is consequently sent to that mote. Any motes that are no longer responsible for any of the particles maintained in the network are shut off.

In the update step, measurements are taken by the sensors of the active nodes. The likelihood (or pseudolikelihood) is then computed in order to evaluate the weights of the particles. Note that the likelihood function of the local particle filters must produce results that are similar to that of a global likelihood using entire sensor measurement data collected in the network, so that maintaining independent particle filters is valid. Estimation then can be performed using some aggregation process. Finally, local resampling can be invoked at each individual node, reacting only on particles within each local node. Periodically, global resampling is performed in order to keep the variance of the entire particle set of the network low.

A brief high level summary of the locally distributed particle filter algorithm is shown in Figure 5.1.

5.3 Simulation Example

We present an example network scenario which can utilize this algorithm. The network architecture and system conditions are first discussed. The type of hardware that is required to run this sensor network then is explained. A specific algorithm is then present describing how to devise a likelihood function which meets the criteria of this algorithm, as well as a discussion on how estimation and resampling is performed.

5.3.1 Network Architecture and Object & Measurement Dynamics

In this sensor network architecture, we only consider one type of node. This node has all the computational power, sensory and communication devices needed onboard. The observations are non-linear functions of the object state. The object dynamics and measurements will be similar to those seen in the previous chapter. The object dynamics can be described by a jump Markov model [24], as in the PDPF, by the initiation distribution $p(u_0, \theta_0, \mathbf{x}_0)$ and the update equations

1. Initialization, $t = 0$
   - For each mote $k = 1, ...K$, activate and initialize some particles with value of the mote position. Each particle is given a weight of one.
   - Set $t \leftarrow 1$.
2. Propagation:
   - For each active mote, propagate its particles $\tilde{\mathbf{x}}_t^{(i)} \sim p(\mathbf{x}_t|\mathbf{x}_{0:t-1}^{(i)})$.
   - For each propagated particle, determine if the value of the particle has left the current motes' responsibility region. If so, activate the mote responsible for particles of this value and transmit the particle to that mote.
   - For each active mote that is no longer responsible for any particles, deactivate.

For each active mote:

3. Update:
   - Update the weights of all particles $\tilde{w}_t^{(i)} = \frac{\mathcal{L}(y_t|\tilde{x}_t)p(\tilde{\mathbf{x}}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{\pi(\tilde{\mathbf{x}}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)},\mathbf{y}_{0:t})}$.
4. Estimation:
   - Perform a weight summation of the particle values with their weights in order to form the estimate of the object's position.
5. Local Resampling:
   - For each active node, perform local resampling using its own particles and weights.
6. Global Resampling:
   - If $t = nT$ (where T is the period where resampling occurs), a global resampling procedure is invoked where aggregation of local normalization costs is performed.
7. Set $t \leftarrow t + 1$. Go to Step 2.

Fig. 5.1 High-level algorithm description of the distributed particle filter that maintains subset of particles at several nodes.

$$u_t \sim p(u_t|u_{t-1}), \tag{5.1}$$

$$\theta_t = \theta_{t-1} + c(u_t) + \epsilon_t, \tag{5.2}$$

$$x_t = x_{t-1} + m[\cos\theta_t, \sin\theta_t], \tag{5.3}$$

where $u_t \in \{0, 1, 2\}$ represents a discrete motion state of the object (continuing straight, making a 0.3 radian left turn or making a 0.3 radian right turn, respectively). $c(u_t)$ represents the angle of turn in radians. The angle of the motion is represented by θ_t, which has a zero-mean Gaussian innovation noise ϵ_t of variance 0.001. The object's position is x_t and has a constant speed $m = 0.5$. The update probability matrix for the discrete state is

$$p(u_t|u_{t-1}) = \begin{pmatrix} 0.75 & 0.65 & 0.65 \\ 0.125 & 0.3 & 0.05 \\ 0.125 & 0.05 & 0.3 \end{pmatrix}.$$

Let $\{r_t, t = 1, 2, ...\}$ denote a discrete-time Markov chain with a finite set of states and known transition probabilities. The object dynamics can be modelled as:

$$x_{t+1} = A(r_{t+1})x_t + B(r_{t+1})\eta_{t+1} + F(r_{t+1})u_{t+1} \tag{5.4}$$

where u_t denotes an exogenous input and η_t is an independent white Gaussian noise sequence. The term r_t model the direction the object is moving in (in one of three directions: straight, left-turn, right-turn) and x_t represents the position.

Each sensor has a detection radius of 8 metres and a probability of detection $p_d = 0.7$. We do not model any errors in communication. We use a variable communication radius for propagation of particles, which is substantially larger than the region of motion determined by the dynamic model. This variation is related to the mote density to ensure that the radius is large enough to reach neighbouring motes.

In the simulations, a mote is activated whenever it is required to make a measurement. The activation will persist for 5 time intervals from the time it received its last activation messages. This determines the set of active motes

at any given time instant. A subset of these motes will maintain particles.

5.3.2 Hardware Implementation

Each mote has a microprocessor and adequate memory in order to maintain a particle filter. Each of these nodes is equipped with a binary proximity sensor. These sensors are capable of detecting the presence of an object in a specific region. Furthermore, the sensors have a false-alarm rate, which describes the probability that a false reading can be reported when the object is not within range.

To prolong the lifespan of the network, optical line-of-sight communication will be use. This type of communication uses much less energy than radio-frequency wireless transmission and thus is very attractive. Power requirements of these motes can further be reduced by using an external querying transceiver, consisting of steerable lasers and a directionally sensitive optical receiver array, by reflecting light in a controllable fashion for long-range communication.

Our network thus consists of sensor nodes equipped with a binary proximity sensor, a controllable reflective device for communication with a global querying transceiver, and a low-power laser and direction-sensitive optical receivers for communication. Figure 5.2 shows the main properties of the network architecture.

The communication to neighbouring nodes works by using the guidable low-power laser, which can be pointed in one of eight directions. Each mote transmits at a different wavelength to ensure there is no interference of data between neighbouring nodes. Each mote also has an optical receiver to determine the direction of the incident communication. Again, this communication can be detected to within one of eight directions, each covering a forty-five degree angle. The communication transmitted by each node is restricted to a specific region of the sensor field, as shown in Figure 5.3. A threshold of minimum and maximum power level will be used at the receiver mote in order to validate any transmission directed towards it.

Overall, communication between nodes is very low since only a few weights are transmitted at any given time step. There is communication with a central transceiver but this is solely achieved through the reflection of light. The power

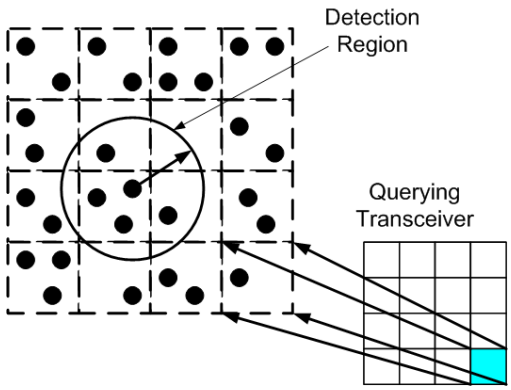


Fig. 5.2 Architecture of the sensor network. The binary sensor of each mote has a detection region, and an associated probability of detection and false-alarm. Communication between motes is accomplished through the use of a small set of lasers and directionally-sensitive optical receivers. The central querying transceiver communicates with the network by projecting light onto the network; very low bit-rate signals are embedded by modulation. Communication back to the transceiver is performed by controlled reflection at the motes.

of the light is set to be in proportion to its current total weight in order for the central transceiver to determine the total weight of the entire system for the particle filter.

5.3.3 Example Algorithm

With the network architecture and hardware setup as described, we are able to formulate an example algorithm based on our proposal. That is, we are able to utilize a local likelihood function has to produce similar results to that of a global function.

The observations consist of binary variables y_t collected at a set N_t of active sensor motes. The observations are modelled as conditionally independent given the object position. For any active mote k,

$$p(y_t^{(k)} = 1) = \begin{cases} p_d & \text{if } x_t \in D^{(k)}, \\ p_f & \text{if } x_t \notin D^{(k)}, \end{cases} \tag{5.5}$$

where $D^{(k)}$ is the detection region of the sensor of mote k.

When the motes measurements are modelled as independent, the likelihood

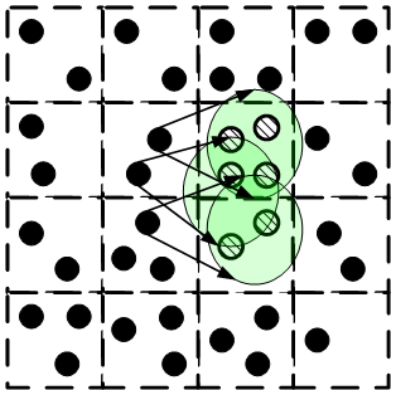


Fig. 5.3 Particle propagation in the sensor network. Motes activate new motes in the predicted direction of object travel. These become the new particles in the filter.

function is equal to the product of the observation probabilities. For a position x_t, the likelihood function is

$$\mathcal{L}(x_t|y_t) = p_d^{n_t}(1-p_d)^{s_t}(1-p_f)^{g_t}p_f^{f_t}. \tag{5.6}$$

The number of active sensors whose detection region includes the position x_t is n_t+s_t, where n_t and s_t are the number of these which record a positive and negative detection, respectively. The number of active sensors whose detection region exclude x_t is $f_t + g_t$, where f_t and g_t are the number of positive and negative detections, respectively. If there are N_t active sensors at a given time t, $N_t = n_t + s_t + f_t + g_t$.

In this thesis, we are interested in the case where the detection capabilities of the sensors have been maximized at the expense of a high false alarm rate. That is, if the object is not within range of the sensor, there is an equal probability of reporting either result, i.e. $p_f \approx 0.5$. The model implies that the sensor can calculate the likelihood of the object being at its location (to within a proportionality constant) based only on neighbourhood measurements. The exact number of active motes is not needed but only a (conservative) upper bound n_{max} on the maximum number of motes in the detection region. Therefore the likelihood evaluation is:

$$\hat{\mathcal{L}}(x_t = v^{(k)}|y_t) = p_d^{n_t^{(k)}}(1-p_d)^{s_t^{(k)}}0.5^{n_{max}-n_t^{(k)}-s_t^{(k)}} \tag{5.7}$$

where $v^{(k)}; k = 1, ..., N$ is the position of mote k, $n_t^{(k)}$ and $s_t^{(k)}$ are the number of sensors in the detection neighbourhood of sensor k that reports positive and negative response, respectively.

Procedure

The particles are maintained in a small set of active motes. The important innovation in the tracking algorithm is that the particles that are maintained at an active sensor k have the position $x_t = v^{(k)}$, where $v^{(k)}$ is the position of the sensor. This restricts the accuracy of the particle filter, but with a reasonable density of motes, there should be no substantial tracking penalty. This innovation allows for the maintenance of the particle to be kept simple. Nodes will know exactly which particles they are responsible for. This also allows for a simple process in propagating the particles. Particle weights are really the only value that has to be transmitted. Propagation of particle is thus simple and can only be passed on to neighbouring nodes. This means that in our scheme, the communication cost will consist mainly of particle weight transmission to neighbouring nodes without the need for a broadcast of measurements values to the entire network. Of course, this process has to be efficient or else the transmission of particles may be costly.

At the initialization stage with time t_0, all motes are activated. Each mote is allotted eight particles, one for each of the distinct directions it is capable of signalling. Each particle will be given the x_0 value as the (uncalibrated) sensor position. The direction r_0 of all particles will be set to straight. Finally the weights of all particles will be set to one.

In the propagation step, particles activate the neighbouring motes in their direction of motion. This is performed by sending a message containing the discrete state and particle weight. For example, a mote can send the message $\{\text{straight}, 0.2\}$ as the state and weight of coming from the west side. Note these messages are sent to a region (as shown in Figure 5.3) so it is possible that multiple particles are generated (with one copy of the message sent to each node within its region). If the motes are uniformly distributed in the region of communication, this replication does not pose a problem except for an additional variance on the estimate.

When selecting the parameters of the communication region, such as the

size, the possible region of movement is determined in large part by the dynamic model. For example, if there is the potential for substantial variation in the direction or distance moved, the region will be larger. If an exact match between the probability of communication to a mote and the probability of movement to that mote according to the dynamic model can be determined, then the importance sampling distribution is the prior [21]. Since in our setup, the match is only an approximation because the communication cannot mimic any complicated distribution, the difference is tolerable since there won't be any large errors relative to the other aspects of the monitoring system. The communication region thus needs to be larger than the region given by the prior to ensure that at least one mote receives a message from any given mote.

For the update step, the active motes need to broadcast a message to activate all nodes within the detection region. These motes then need to perform a proximity measurement and broadcast the results. Finally, each mote needs to calculate its likelihood according to Equation (5.6). The weights then need to be updated by multiplying the likelihood with prior (the importance sampling distribution).

The estimation step can then be performed by having the global transceiver send a request to all active motes to respond with the total weights of their particles. This is performed by each mote remodulating the reflected power in proportion to its weight. The transceiver can then retrieve this information with its receiving array and perform an averaging of the received power to estimate the object's location.

Finally, resampling can be performed using the total weight of the system received by the global transceiver. Each mote can normalize its weight and then perform the systematic resampling [62] procedure on its own particles.

A brief high level summary of the algorithm is shown in Figure 5.4.

5.4 Tracking Accuracy

The experiments were conducted using 4000, 6000, 8000, 16000, 32000 and 64000 sensors, each performing 90 trials in each case. These values represent densities of approximately 0.25, 0.375, 0.5, 1, 2 and 4 motes per square metre. We compare the tracking performance of the distributed particle filter with that of a centralized one. The centralized particle filter propagates particles

1. Initialization, $t = 0$
   - For each sensor $k = 1, ..., K$
     - Activate and allot $i = 1, ..., 8$ particles for each signalling direction.
     - For each particle, set the position to $x_0^{(k,i)} = v^{(k)}$, the direction to $r_0^{(k,i)} = \{\text{straight}\}$, and weights to $w_0^{(k,i)} = 1$.
   - Set $t \leftarrow 1$.
2. Propagation:
   - For all particles i of all nodes $k = 1, ..., K$, sample $\tilde{\mathbf{x}}_t^{(k,i)} \sim p(\mathbf{x}_t | \mathbf{x}_{0:t-1}^{(k,i)})$.
   - Compute the direction of motion $r_t^{(k,i)}$ based on $\tilde{\mathbf{x}}_t^{(k,i)}$.
3. Update:
   - Update the weights of all particles $\tilde{w}_t^{(k,i)} = \frac{\mathcal{L}(y_t|\tilde{x}_t) p(\tilde{\mathbf{x}}_t^{(k,i)}|\mathbf{x}_{t-1}^{(k,i)})}{\pi(\tilde{\mathbf{x}}_t^{(k,i)}|\mathbf{x}_{0:t-1}^{(k,i)}, \mathbf{y}_{0:t})}$.
4. Send Messages:
   - For all particles i of all nodes $k = 1, ..., K$, send the message $\{r_t^{(k,i)}, w_t^{(k,i)}\}$ to the direction of motion $\{r_t^{(k,i)}\}$.
5. Estimation:
   - Perform averaging on the received power from a receiver array of all active motes to compute the estimate of the object's location.
6. Resampling:
   - Global transceiver sends total weight of system to all active motes to normalize the weights and perform resampling procedure on their own particle set.
7. Set $t \leftarrow t + 1$.

Fig. 5.4 High-level algorithm description of the example particle filter algorithm that maintains subsets of particles at multiple nodes.

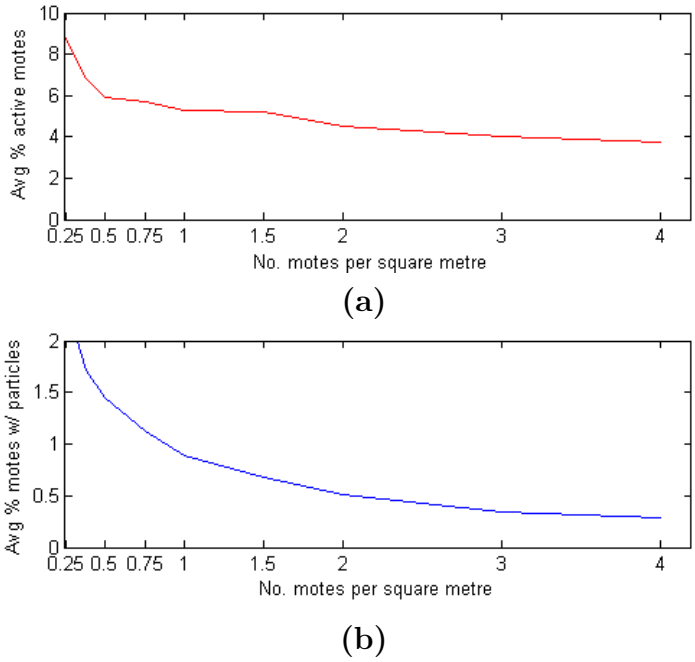


Fig. 5.5 The average percentage of active motes per time interval as a function of the sensor density in the network. (a) the average percentage of motes that make a measurement. (b) the average percentage of motes maintaining particles.

according to the dynamic model and used 2000 particles. The distributed filter uses approximately 2000 particles, but this number is subject to the variability discussed above, and there is a substantial duplication of particles because of quantization effects.

Experiments were conducted to determine the best communication radius. The optimum range determined was 4.5 metres for a mote density of 0.25, 3 metres for motes densities 0.375, 0.5 and 1, 2 and 4 metres for a mote density of 2.

Figure 5.5 shows the average percentage of active motes in the system as a function of the sensor density. Over the range of 0.25 to 4 (which is 4000 to 64000 sensors), the percentage is a slow decay from approximately 6 to 3 percent, which corresponds to approximately 240 to 1920 motes. The average percentage of motes maintaining particles at any instant ranges from 1.5 to 0.5 percent, corresponding to approximately 60 to 320 motes. It is clear from this that the distributed particle filter approach achieves a dramatic reduction in

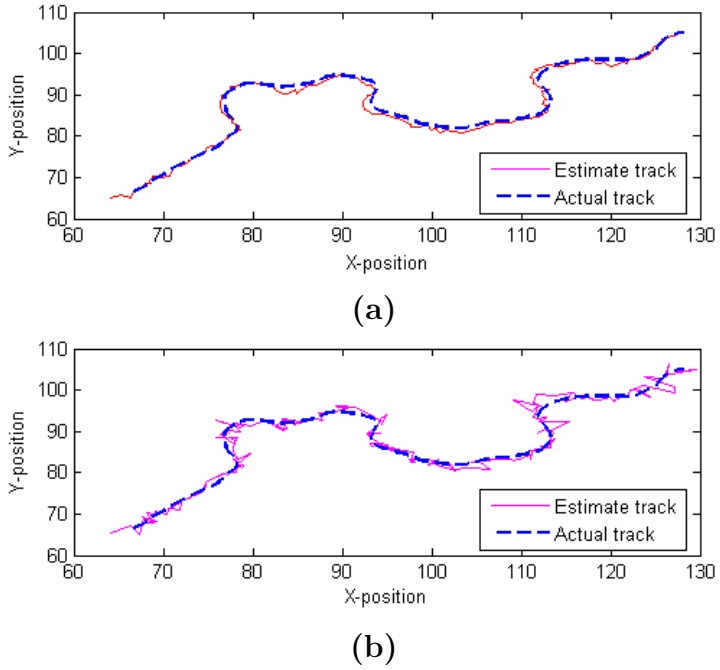


Fig. 5.6 Examples of tracking performance for a motes density of 1 (16,000 sensors). (a) Centralized particle filter; (b) Distributed particle filter.

the number of active motes (and hence the energy expenditure of the system). It also provides an effective distribution of the computational requirements.

Figure 5.6 shows examples of tracking behaviour for the centralized and distributed algorithms in the case of 16,000 motes. The distributed filter is much less smooth in its estimates, primarily due to the sparsity of the motes. However, in terms of the average mean squared error, the performance of the two algorithms for this density of motes is comparable, as indicated by Figure 5.7. This relative performance of the distributed algorithm deteriorates as the number of sensors is decreased. Particle positions must correspond to associated mote positions, and if there are too few motes, the resultant particle distribution is a poor approximation to the filtering distribution. With a density greater than that of 1.7 (or 28,000 motes), it appears that the distributed case performs better than the centralized algorithm. This could possibly be due to a few factors. First, the number of unique particle in the distributed case is not always consistent, as shown in Figure 5.8. Due to the replication factor of the particles in the propagation stage, the number of unique parti-

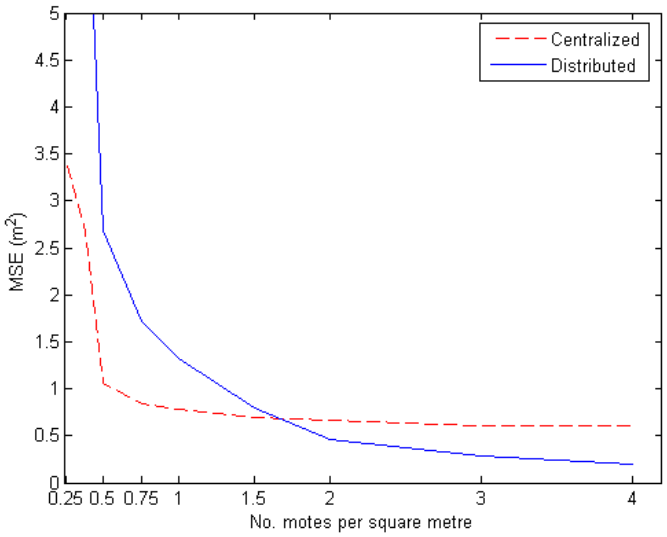


Fig. 5.7 A comparison of the average mean-squared error in position estimation as a function of the number of network motes.

cles hovers slightly above 2000 particles, which is the exact number used in the centralized case. Note that using more particles generally produces more accurate results. In addition, another factor that could account for the discrepancy in MSE values is that when the mote density is greater than 1.5, the experimental conditions were altered. The communication radius was lowered since the average distance to the next neighbour was smaller.

5.5 Communication & Computational Costs

Computing the communication and computational costs in this scenario is complex. There is no good way to make a direct comparison of this method with any other algorithm since a low-power laser is used for communicating with neighbouring nodes in this proposed scenario, as oppose to radio-frequency communications. Nevertheless, the communication between nodes is very low since a few weights are transmitted. Communication with the central transceiver is also required. However, this is done through the reflection of light which is not energy demanding.

Pertaining to computational costs, maintaining the particles at multiple nodes does not cost any more than maintaining them at one node. This is

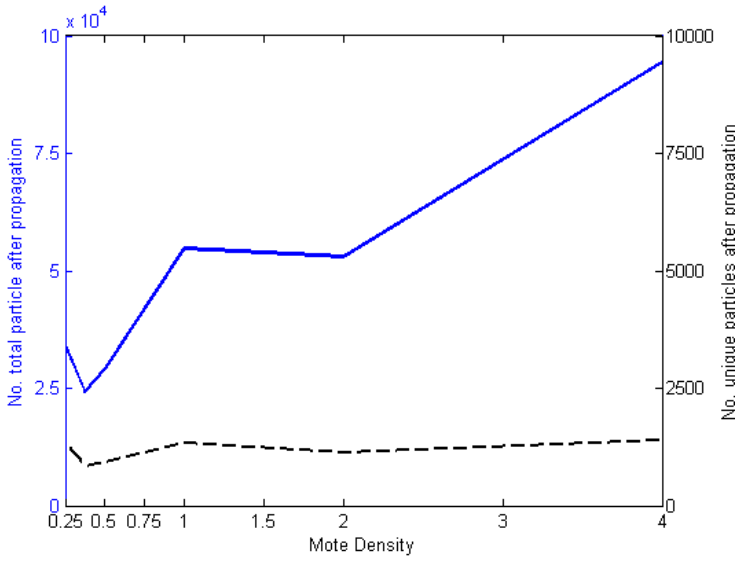


Fig. 5.8 Average number of particles in the sensor network after message passing.

because the computation costs are directly proportional to the number of particles. Thus, the computational costs are just distributed over various nodes in this scenario. The computational costs of this algorithm are certainly less than the PDPF, since duplicate computations do not have to be performed to maintain multiple identical particle filters.

In addition as indicated before, energy costs associated to computation far exceeds the costs related to communication. We have kept the communication costs low due to the hardware we used. This makes the local distributed particle filter a more desirable algorithm in the context of a sensor network.

5.6 Overall Assessment

Under the conditions presented, we are able to successfully create an algorithm based on distributing subsets of particles to local nodes to process. There is no need for a central fusion or a node to perform data fusion since the likelihood function was modified in order to avoid the collection of all data in the network. Therefore, substantial savings can be obtained in terms of communication costs since there is no data routing to be performed on measurement values or filtering information. Since the measurement values are only binary with

out-of-range measurements of current nodes having a 50% false alarm rate, this effectively means that knowing the exact measurement values of non-local sensors is not required. Therefore, each node is able to maintain its own particle filter independently.

For the algorithm to produce acceptable tracking results in our specific model, a large number of sensors was required in the network. In our scenario of a 128×128 plane, this means at least $26,000$ sensors. Fortunately, there is only an average of 125 active motes with particles. Therefore, the downside to this is that the cost of the hardware is more expensive than that of the PDPF due to the increase in the number of motes required.

Nevertheless, this algorithm is still more beneficial than a centralized particle filter if a suitable likelihood function can be used. This alleviates the need of collecting external measurement data from other nodes. While hardware costs may have increased, communication costs have significantly dropped and energy resources required to run this algorithm is low and efficient. This consequently means that network lifetime of this algorithm is very high. If the amount of nodes used in the network is as many as described, such as in the ten-thousands, then each node will be active for a very short time meaning that it will not lose energy for a very long time.

Chapter 6

Conclusion

In this thesis, we have analyzed the use of particle filters for tracking in sensor networks. As communication cost and energy consumption are two main concerns in an energy-limited sensor network, our interest lies in creating an algorithm which can reduce the resource costs.

In particular, we were interested in using the particle filter because of the powerful nature of the algorithm in an estimation problem. The particle filter is simple to perform and can be used for any real-world problem.

A distributed algorithm is desirable for a sensor network problem as this reduces the communication cost associated with aggregating data at a single location. Furthermore, a side-effect to sending data to a fusion centre is the unbalanced energy consumption used by nodes near the centre node to route the necessary information. Our aim is to formulate a distributed algorithm to achieve the goal of a centralized particle filter without the need of a command centre. In particular, we are interested in using a distributed version of the particle filter to achieve our tracking goals in a sensor network environment.

We first analyzed the particle filter to determine what exactly needs to be modified. In a distributed algorithm, the main idea is to avoid having one device perform all the computation in order to alleviate the heavy energy consumption at one node. There were four proposals suggested for the placement of a particle filter – filter maintained at a leader node, common filter maintained at multiple particle, unique filters maintained at multiple nodes, subsets of particles distributed over multiple nodes. Each of these suggestions varied in terms of transmission costs of data.

When utilizing a distributed algorithm, there is a need to update the nodes with particle filter information. The simplest way to do this is to transmit the particle states (along with their weights), as this is the raw data of a particle filter. A more cost efficient method can be to send the parametric representation of the distribution of the particle set. While this saves on transmission costs, a representation will not allow the updated node to reconstruct exactly the particle set and thus will have some error. Depending on the application, this may or may not be acceptable. Another way to update another node is by strictly distributing the measurement data and having the particle filter of the other node update itself with the sensor information. This only works if the particle filter residing on each node is identically initialized.

When considering the issues of the approaches to maintaining a particle filter and the issue of particle filter transfer, it is difficult to say which combination of these methods is the most optimal to use. This is because there are many other factors to consider, such as the issue of maintaining an accurate particle set, object estimation and transmission efficiency. In the maintaining an accurate particle set problem, there was a discussion on how the particle set will have an increased variance as time passes. This can be rectified either by a particle set replacement or resampling. The particle set replacement uses the same methodology as the transfer of particle information and the resampling step involves using some type of aggregation process, which can be performed either using a centralized device, or a structured process. Concerning object estimation, an aggregation procedure similar to resampling has to be considered. Finally, a brief discussion of transmission efficiency was made to note that despite the effort put into selecting the above elements for a particle filter algorithm, the method of transmission is always important to consider. The protocol must have a balance between efficiently and effectiveness (i.e. handling the case when errors arise). The method of transmitting data should be compressed or quantized, in order to reduce costs.

Following our look at the particle filter, we described two specific networking scenarios and proposed for each a distributed particle filtering algorithm. The first is an architecture whereby sensor measurements are performed at multiple nodes. In this particular scenario, we proposed an algorithm called the *parallel distributed particle filter* to track the position of an object. The algorithm works by maintaining a particle filter at two levels – a local and

global. The local particle filter acts on the local measurements to perform an estimation on a temporary basis. A global particle filter is then used to perform a more accurate estimation using all the measurements available in the network.

The novelties brought by this algorithm are two fold. The first is a quantization and encoding scheme to compress the measurement data for encoding. The scheme takes advantage of the local particle filters by using them, along with a Huffman tree to encode measurement data. This could compress the data to only a few bits, which is a substantial savings compared to more straightforward compression schemes which can take as much as 16 bits. The second innovation comes by saving on the communication cost by collecting several time steps worth of measurement values before sending the information out to the network. The communication savings comes in the form of eliminating some packet overhead bits sent into the network. A 60% reduction in transmitted bits could be achieved.

The parallel distributed particle filter works by having active node collecting data run the particle filter on its own measurement data. This will help the filtering algorithm compute a local estimate. Then, quantization and vectorization can be used on each active node to compress and share its data to the rest of the network. Each individual node will then receive all the data collected throughout the network and perform the particle filter on this data. Consequently, each node will be able to compute the exact same estimate and carry on its tracking tasks. The slight disadvantage of this algorithm is that it requires more computational steps, but it does use an efficient process to disseminate local sensor data. The procedure is particular useful when the rate of query of the object's location is the same as vector length for transmission. This is because the global estimate can consistently be used and its results are almost identical to that of the centralized process. When the rate of query is different from the vector length of transmission, the local estimate can still be used, albeit not as accurate as the global estimate. This algorithm makes very good use of the particle filter and data dissemination procedure to make this method very effective.

The second distributed particle filtering algorithm we proposed was the *localized distributed particle filter*. This method is based on the distribution of subsets of the particles to individual nodes. If the right networking scenario is

achieved, each node can maintain a particle filter independently. The Achilles heal is the likelihood function, which is normally computed using the set of measurement data in the network. However, under the right networking conditions, it was shown that the likelihood function which could be calculated using local measurements only. Because of this arrangement, data fusion was not necessary and the algorithm required less energy to perform as compared to a centralized algorithm.

A drawback was that the resampling and estimation steps required the use of a global transceiver. In our specific scenario, we used lasers to alleviate the energy costs but any other efficient aggregation process can be used. In addition, another disadvantage is large amount of hardware (motes) required to operate. Fortunately, the amount of active motes is small, so energy consumption is still low.

6.1 Limitations

While we have explored a variety of distributed algorithms using the particle filter, there is still much work to perform. We have shown in a simulation based model that the algorithm could be effective in the real-world. However, a hardware system built with these parameters can be explored to further show the theory brought forth in this paper is valid. It has to be noted that a real world implementation will produce other complications, ignored in this paper, such as communication interference and individual node failures.

Resampling is also another issue which can be reviewed. Ideally, network devices should be totally autonomous. Therefore, the development of a completely distributed resampling procedure, possibly done through in-network algorithm for weight aggregation and normalization is needed. This can go along nicely with the distributed scenario of the parallel distributed particle filter and the localized distributed particle filter.

A further avenue to explore is a case when the sensor network region is substantially bigger than the area we consider, possibly with a smaller mote density. This will complicated matters as data transmission is very limited and may only reach a neighbour one hop away.

References

[1] B. Warneke, M. Last, B. Liebowitz, and K. S. Pister, "Smart dust: Communicating with a cubic-millimeter," *IEEE Computer*, vol. 34, pp. 44–51, 2001.

[2] J. Kahn, R. Katz, and K. Pister, "Mobile networking for smart dust," in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking*, (Seattle, WA), August 1999.

[3] V. Hsu, R. Kahn, and K. Pister, "Wireless communications for smart dust," *Technical report, Electronics Research Laboratory Tech. Memo. M98/2*, February 1998.

[4] C.-Y. Chong and S. P. Kumar, "Sensor networks: Evolution, opportunities, and challenges," *Proceedings of the IEEE*, vol. 91, pp. 1247–1256, August 2003.

[5] G. Welch and G. Bishop, "An introduction to the Kalman filter," in *Proceedings of the International Conference on Computer Graphics and Interactive Techniques*, (Los Angeles, CA), 2001.

[6] Y. Ho and R. Lee, "A Bayesian approach to problems in stochastic estimation and control," *IEEE Trans. Atomat. Contr.*, vol. AC-9, pp. 333–339, 1964.

[7] A. Jazwinski, *Stochastic processes and filtering theory*. New York: Academic Press, 1970.

[8] H. Tanizaki, "Nonlinear filters: Estimation and applications.," *Lecture Notes in Economics and Mathematical Systems*, vol. 400, 1998.

[9] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "A novel approach to nonlinear and non-Gaussian Bayesian state estimation," *Radar and Signal Processing, IEE Proceedings F*, vol. 140, pp. 107–113, April 1993.

[10] R. J. Meinhold and N. D. Singpurwalla, "Robustification of Kalman filter models," *Journal of the American Statistical Association*, vol. 84, pp. 479–486, 1989.

[11] R. Bucy and K. Senne, "Digital synthesis of nonlinear filters," *Automatica*, vol. 7, pp. 287–298, 1971.

[12] F. Martinerie, "Data fusion and tracking using HMMs in a distributed sensor network," *IEEE Transactions on Aerospace and Electronics System*, vol. 33, pp. 11–28, January 1997.

[13] D. Alspach and H. Sorenson, "Nonlinear Bayesian estimation using Gaussian sum approximations," *IEEE Transactions on Automatic Control*, vol. 17, pp. 439–448, August 1972.

[14] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, pp. 174–188, February 2002.

[15] Wikipedia, "Monte Carlo method." `http://en.wikipedia.org/wiki/Monte_Carlo_method`, 2005.

[16] M. Isard and A. Blake, "Condensation – conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, pp. 5–28, 1998.

[17] J. Deutscher, A. Blake, and I. Reid, "Articulated body motion capture by annealed particle filtering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, (Hilton Head Island, SC), pp. 126–133, 2000.

[18] I. M. Rekleitis, G. Dudek, and E. Milios, "Probabilistic cooperative localization and mapping in practice," in *Proceedings of the IEEE International Conference in Robotics and Automation*, vol. 2, (Taipei, Taiwan), pp. 1907–1912, September 2003.

[19] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P. J. Nordlund, "Particle filters for positioning, navigation, and tracking," *International Journal of Computer Vision*, vol. 50, pp. 425–437, February 2002.

[20] K. Kanazawa, D. Koller, and S. J. Russell, "Stochastic simulation algorithms for dynamic probabilistic networks," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pp. 346–351, 1995.

[21] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monto Carlo Methods*. New York: Springer-Verlag, 2001.

[22] W. Gilks, S. Richardson, and D. Spiegelhalter, *Markov Chain Monte Carlo in Practice*. London: Chapman and Hall, 1996.

[23] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*. New York: Springer-Verlag, 1999.

[24] A. Doucet, N. J. Gordon, and V. Krishnamurthy, "Particle filters for state estimation of jump Markov linear systems," *IEEE Trans. Signal Processing*, vol. 49, pp. 613–624, 1999.

[25] J. Liu and R. Chen, "Sequential Monte Carlo methods for dynamical system," *Journal of the American Statistical Association*, vol. 93, pp. 1032–1044, 1998.

[26] J. Carpenter, P. Clifford, and P. Fearnhead, "Building robust simulation-based filters for evolving data sets," *Technical report, University of Oxford, Dept. of Statistics*, 1999.

[27] J. S. Liu and R. Chen, "Blind deconvolution via sequential imputation," *Journal of the American Statistical Association*, vol. 90, pp. 567–576, 1995.

[28] M. Pitt and N. Shephard, "Filtering via simulation: auxiliary particle filters,"

[29] E. Wan and R. van der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proceedings of the IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pp. 153–158, October 2000.

[30] N. Gordon and A. Whitby, "A Bayesian approach to target tracking in the presence of glint," in *Proceedings of the International Society for Optical Engineering*, vol. 2561, (Orlando, FL), pp. 472–483, October 1995.

[31] C. Berzuini, N. G. Best, W. R. Gilks, and C. Larizza, "Dynamic conditional independence models and Markov chain Monte Carlo methods," *Journal of the American Statistical Association*, vol. 92, no. 440, pp. 1403–1412, 1997.

[32] W. Gilks and C. Berzuini, "Following a moving target–Monte Carlo inference for Bayesian dynamic models," *Journal of Royal Statistic Society B*, vol. 63, pp. 127–146, 2004.

[33] C. Andrieu, P. Djurić, and A. Doucet, "Model selection by MCMC computation," *Signal Processing*, vol. 81, pp. 19–37, 2001.

[34] J. Pike, "The Sound Surveillance System (SOSUS)." `http://www.fas.org/irp/program/collect/sosus.htm`, 1999.

[35] W. Chen, Y. Gong, and X. Liu, "Applications of sensor networks." `http://www.cs.umass.edu/~chenwf/application.ppt`.

[36] T. T. Hsieh, "Using sensor networks for highway and traffic applications," *IEEE Potentials*, vol. 23, pp. 13–16, April–May 2004.

[37] K. Martinez, J. K. Hart, and R. Ong, "Environmental sensor networks," *IEEE Computer*, vol. 37, pp. 50–56, August 2004.

[38] Crossbow, "Wireless sensor network." `http://www.xbow.com`, 2005.

[39] University of California, Berkeley, "TinyOS." `http://www.tinyos.net/`, 2005.

[40] M. J. Coates, "Distributed particle filtering for sensor networks," in *Proceedings of the International Symposium of Information Processing in Sensor Networks*, (Berkeley, CA), April 2004.

[41] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Information Theory*, vol. 28, pp. 129–137, March 1982.

[42] J. Max, "Quantization for minimum distortion," *IRE Trans. Information Theory*, vol. 6, pp. 7–12, March 1960.

[43] R. R. Brooks, P. Ramanathan, and A. M. Sayeed, "Distributed target classification and tracking in sensor networks," *Proceedings of the IEEE*, vol. 91, pp. 1163–1171, August 2003.

[44] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking*, (Seattle, WA), pp. 263–270, August 1999.

[45] J. Liu, J. Liu, J. Reich, P. Cheung, and F. Zhao, "Distributed group management for track initiation and maintenance in target localization applications," in *Proceedings of the International Workshop on Information Processing in Sensor Networks*, vol. 90, (Palo Alto, CA), pp. 567–576, April 2003.

[46] F. Zhao, J. Shin, and J. Reich, "Information-driven dynamic sensor collaboration," *IEEE Signal Processing Magazine*, vol. 19, pp. 61–72, March 2002.

[47] J. Liu, J. Reich, and F. Zhao, "Collaborative in-network processing for target tracking," *Journal on Applied Signal Processing*, pp. 378–391, 2003.

[48] X. Sheng and Y.-H. Hu, "Distributed particle filter with GMM approximation for multiple target localization and tracking in wireless sensor network," in *Proceedings of the International Symposium of Information Processing in Sensor Networks*, (Los Angeles, CA), April 2005.

[49] R. Matthew, G. Geoffrey, and T. Sebastian, "Decentralized sensor fusion with distributed particle filters," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, (Acapulco, Mexico), August 2003.

[50] J. Shin, N. Lee, S. Thrun, and L. Guibas, "Lazy inference on object identities in wireless sensor networks," in *Proceedings of the International Symposium of Information Processing in Sensor Networks*, (Los Angeles, CA), April 2005.

[51] A. T. Ihler, J. W. F. III, and A. S. Wilsky, "Particle filtering under communications constraints," in *Proceedings of the IEEE Workshop on Statistical Signal Processing*, (Bordeaux, France), July 2005.

[52] M. Bolic, P. M. Djuric, and S. Hong, "Resampling algorithms and architectures for distributed particle filter," *IEEE Trans. Signal Processing*, vol. 53, pp. 2442–2450, July 2005.

[53] A. Papoulis and S. U. Pillai, *Probability, Random Variables and Stochastic Processes with Errata Sheet*. New York: McGraw-Hill Science/Engineering/Math, 2002.

[54] R. G. Gallager, P. A. Humblet, and P. M. Spira, "A distributed algorithm for minimum weight spanning trees," *ACM Trans. Program. Lang. and Systems*, vol. 5, pp. 66–77, January 1983.

[55] R. M. Gray, "Quantization methods." `http://www.stanford.edu/class/ee372/methods4.pdf`, 2003.

[56] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Transactions on Information Theory*, vol. 44, pp. 2325–2384, October 1998.

[57] A. Gersho and R. Gray, *Vector Quantization and Signal Compression*. Kluwer Academic Press, 1992.

[58] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, Massachusetts: MIT Press, 2 ed., 2001.

[59] P.-J. Arida and A. Chan-You, "Wireless sensor networks hardware setup and testing," *Technical report, McGill University*, 2004.

[60] C. Miller and P. Robitaille-Grenier, "Wireless sensor network setup for a distributed particle filter algorithm," *Technical report, McGill University*, 2005.

[61] B. C. Arnold and D. J. Strauss, "Pseudolikelihood estimation: Some examples," *Sankhya*, vol. 53, pp. 233–243, 1991.

[62] G. Kitagawa, "Monte Carlo filter and smoother for nonlinear non-Gaussian state space models," *J. Comp. Graph. Statist.*, vol. 5, pp. 1–25, 1996.

[63] G. Ing and M. J. Coates, "Parallel particle filters for tracking in wireless sensors," in *Proceedings of the IEEE Workshop on Signal Processing Advances in Wireless Communications*, (New York, NY), June 2005.

[64] M. J. Coates and G. Ing, "Sensor network particle filters: Motes as particles," in *Proceedings of the IEEE Workshop on Statistical Signal Processing*, (Bordeaux, France), July 2005.

[65] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications*, vol. 40, pp. 102–114, August 2002.

[66] R. R. Brooks, C. Griffin, and D. Friedlander, "Self-organized distributed sensor network entity tracking," *International Journal of High-Performance Computing Applications*, vol. 16, pp. 207–219, 2002.

[67] D. McErlean and S. Narayanan, "Distributed detection and tracking in sensor networks," in *Proceedings of the Asilomar Conference on Signals, Systems and Computers*, vol. 2, (Pacific Grove, CA), pp. 1174–1178, November 2002.

[68] D. Li, K. D. Wong, Y. H. Hu, and A. M. Sayeed, "Detection, classification and tracking of targets in distributed sensor networks," *IEEE Signal Processing Magazine*, vol. 19, pp. 17–29, March 2002.

[69] L. Guibas, "Sensing, tracking and reasoning with relations," *IEEE Signal Processing Magazine*, vol. 19, pp. 73–85, March 2002.

[70] L. M. Kaplan, Q. Le, and P. Molnar, "Maximum likelihood methods for bearing-only target localization," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, (Salt Lake City, UT), pp. 554–557, May 2001.

[71] K. Yao, R. E. Hudson, C. W. Reed, T. Tung, D. Chen, and J. C. Chen, "Estimation and tracking of an acoustic/seismic source using a beamforming array based on residual minimizing methods," *Proceedings of IRIA-IRIS*, pp. 153–163, January 2001.

[72] H. Yang and B. Sikdar, "A protocol for tracking mobile targets using sensor networks," in *Proceedings of the International Workshop on Sensor Network Protocols and Applications*, (Anchorage, AK), pp. 71–81, March 2003.

[73] J. Shin, L. J. Guibas, and F. Zhao, "A distributed algorithm for managing multi-target identities in wireless ad-hoc sensor networks," in *Proceedings of the International Conference on Information Processing in Sensor Networks*, (Palo Alto, CA), pp. 223–238, April 2003.

[74] C. Gui and P. Mohapatra, "Power conservation and quality of surveillance in target tracking sensor networks," in *Proceedings of the International Conference on Mobile Computing and Networking*, (Philadelphia,PA), pp. 129–143, September 2004.

[75] D. Schulz, W. Burgard, D. Fox, and A. B. Cremers, "Tracking multiple moving targets with a mobile robot using particle filters and statistical data association," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, (Sydney, Austrailia), pp. 1665–1670, 2001.

[76] J. Kurose and V. Lesser, "Sensor networks: intro, overview, example." `http://www-net.cs.umass.edu/cs791_sensornets/sensornets_class1_kurose.ppt`, 2002.

[77] V. Isler, J. Spletzer, S. Khanna, and C. Taylor, "Target tracking with distributed sensors: The focus of attention problem," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Las Vegas, NV), pp. 792–798, October 2003.

[78] Y. Bar-Shalom and T. Fortmann, *Tracking and Data Association*. New York: Academic Press, 1988.