Chapter 6

Distributive Tracking in Wireless Sensor Networks

This chapter develops the distributive tracking algorithms for tracking a single target in wireless sensor networks. Based on the collaborative information processing framework and the hierarchical sensor network architecture proposed in Chapter 3, this chapter extends PF, EKPF and PF-PDAF algorithms developed in Chapters 4 and 5 for distributively tracking a single target over a number of sensor clusters in the sensor field. To conserve communication bandwidth in propagating the estimation results amongst the sensor clusters, this chapter adopts the Gaussian mixture model (GMM) to approximate the probability density function of the target state. Moreover, this chapter develops a composite objective function to assist sensing nodes selection in the distributive tracking algorithms for the purpose of conserving energy consumption while still maintaining the desirable tracking accuracy in the wireless sensor networks.

6.1 Introduction

In PF, EKPF and PF-PDAF tracking algorithms developed in Chapters 4 and 5, the probability density function of the target state is updated based on the measurements obtained by a set of active sensing nodes within one sensor cluster. In principle, this sensor cluster can be expanded to include all sensing nodes deployed in the sensor field with the cluster leader collecting the measurements from all these sensing nodes to update the target state estimate. However, such a centralized scheme would require intensive communication resources and energy consumption since a large number of sensing nodes need to transmit their measurements to the cluster leader. For sensing nodes far from the cluster leader, multi-hop routing is needed which corresponds the energy requirements. Therefore, it is imperative to lower the communication overheads and energy consumption while still maintaining the desirable tracking accuracy through the development of distributive tracking schemes.

This chapter aims to develop distributive algorithms for tracking a single target in wireless sensor networks within the collaborative information processing framework proposed in Chapter 3. Across the whole sensor field, a number of sensor clusters are dynamically formed according to the geometry of the sensing nodes and the predicted position of the target. Each sensor cluster occupies a smaller region within the sensor field; this complies with the assumptions we have made in Chapter 3 that the sensing nodes are only equipped relatively short range radio transmitters. At each time step, only one sensor cluster is active; and a portion of sensing nodes within this sensor cluster are activated to sense and provide their measurements to the cluster leader, which then runs the tracking algorithm (i.e., either PF or EKPF or PF-PDAF) to update the probability density function of the target state (hereafter named as *"belief"*). After updating the target state estimate, the cluster leader computes the distance between itself and the predicted position of the target at the next time step. If this distance is larger than a pre-defined threshold, the cluster leader initiates a new cluster leader election process by broadcasting a message to its neighbouring leader nodes. The details of cluster leader election and the following sensor cluster formation can be found in Chapter 3. When the election process completes, the current cluster leader forwards its belief to the new cluster leader. The new cluster leader selects sensing nodes within the newly formed sensor cluster, activating these sensing nodes, collecting their measurements and updating the "belief". The above process will be continued until the target moves out of the sensor field. By this way, the estimation of the target state is updated sequentially in a distributive manner. Figure 6.1 depicts the scenario of distributive tracking a single target in a wireless sensor network.

To facilitate distributive target tracking, this chapter adopts a Gaussian mixture model (GMM) to approximate the "belief". Instead of transmitting a high volume of particles data , the current cluster leader only transmits a handful of GMM parameters to the next cluster leader. This will greatly conserve communication bandwidth and energy consumption in a wireless sensor network. Moreover, this chapter develops a composite objective function incorporating both the information utility measure and the energy consumption measure to facilitate the sensing node selection in the distributive tracking algorithms. This composite objective function will enable the distributive tracking algorithms to achieve the desirable tracking accuracy while still maintaining lowered energy consumption.

The organization of this chapter is as follows. Section 6.2 presents the Gaussian mixture model (GMM) which is adopted to approximate the "belief" of target state. The Expectation-Maximization (EM) algorithm which is used to compute the parameters of GMM is also detailed. Section 6.3 presents the sensing node selection scheme that can be integrated into the distributive tracking algorithms to further address the trade-off between tracking accuracy and energy consumption. Section 6.4 implements the distributive PF, EKPF and PF-PDAF tracking algorithms. Section 6.5 conducts the simulation and Section 6.6 summarizes this chapter.

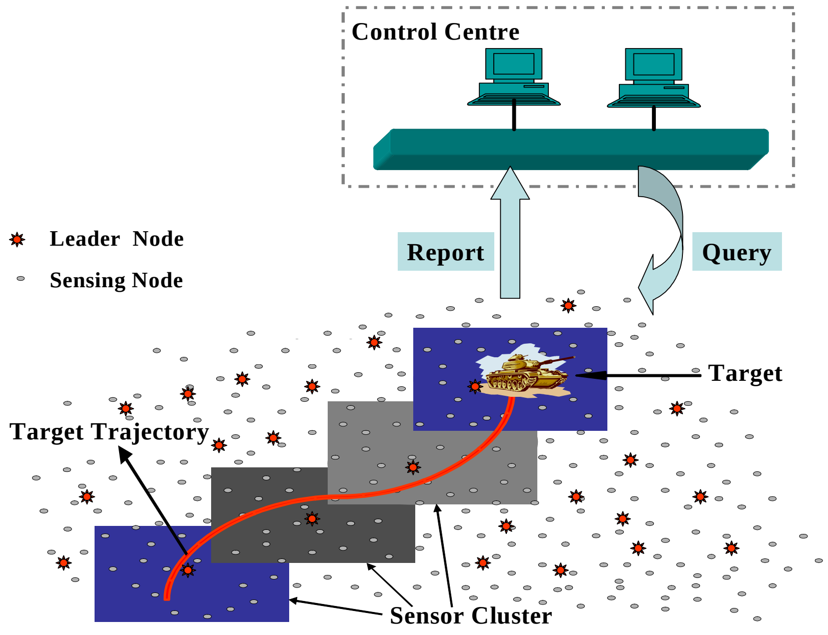


Figure 6.1 Distributive target tracking in a wireless sensor network

6.2 Gaussian Mixture Model (GMM) for the Propagation of Target State Estimate

6.2.1 GMM for the Approximation of Probability Density Function of the Target State

In the distributive target tracking as depicted in Figure 6.1, the current cluster leader needs to transmit its "belief", i.e. the probability density function of the target state $p\left(\mathbf{x}_k \middle| \mathbf{z}_{0:k}\right)$ to the next cluster leader when the target leaves the current sensor cluster. For PF, EKPF and PF-PDAF tracking algorithms developed in the previous chapters, at the k-th time step $p\left(\mathbf{x}_k \middle| \mathbf{z}_{0:k}\right)$ is approximated by a set of weighted particles as follows:

$$p\left(\mathbf{x}_k \middle| \mathbf{z}_{0:k}\right) \approx \hat{p}\left(\mathbf{x}_k \middle| \mathbf{z}_{0:k}\right) = \frac{1}{N}\sum_{i=1}^{N} \delta\left(\mathbf{x}_k - \mathbf{x}_k^{*i}\right) \tag{6.1}$$

where $\hat{p}\left(\mathbf{x}_k \middle| \mathbf{z}_{0:k}\right)$ is the particles' approximation of the probability density function of the target state. Note that in PF, EKPF and PF-PDAF algorithms, the particles $\left\{\mathbf{x}_k^{*i}\right\}_{i=1}^{N}$ have equal weights of $\frac{1}{N}$ after the resampling step. Therefore, to propagate the "belief" from the current cluster leader to the next cluster leader, a high volume of data (i.e. particles) needs to be transmitted. In previous chapters, we use 1000 particles in PF and PF-PDAF and 200 particles in EKPF for an acceptable level of tracking performance. This would require tremendous amount of power and bandwidth for communication and might not be feasible in the resource constrained wireless sensor networks. To address this problem, it is proposed that the Gaussian Mixture Model (GMM) be used to approximate the probability density function of the target state, $p\left(\mathbf{x}_k \mid \mathbf{z}_{0:k}\right)$.

The GMM is a mixture of several Gaussian distributions and its probability density function is defined as a weighted sum of Gaussians. By using the GMM, $p\left(\mathbf{x}_k \mid \mathbf{z}_{0:k}\right)$ can be approximated as follows:

$$p\left(\mathbf{x}_k \mid \mathbf{z}_{0:k}\right) \approx \sum_{j=1}^{M} p\left(\mathbf{x}_k \mid j\right) P(j) \tag{6.2}$$

In Equation 6.2, M is the number of mixture components. $p\left(\mathbf{x}_k \mid j\right)$, $j = 1,\ldots,M$ are the probability density functions of the mixture components which are Gaussian distributions with mean vector $\boldsymbol{\mu}_j$, $j = 1,\ldots,M$ and covariance matrix $\boldsymbol{\Sigma}_j$, $j = 1,\ldots,M$, i.e. $p\left(\mathbf{x}_k \mid j\right) \sim N\left(\mathbf{x}_k ; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)$. $P(j)$, $j = 1,\ldots,M$ are the weights of the mixture components (called the mixing coefficients) which satisfy $0 < P(j) < 1$ and $\sum_{j=1}^{M} P(j) = 1$. In the remaining of this chapter, the mixing coefficient $P(j)$, the mean $\boldsymbol{\mu}_j$ and the covariance $\boldsymbol{\Sigma}_j$ are referred to as the GMM parameters. Note that we discard the time step index k in the GMM parameters for the sake of clarity and this would not cause any ambiguity.

Given the probability density function $p\left(\mathbf{x}_k \mid \mathbf{z}_{0:k}\right)$ which is represented by N particles, now our purpose is to find a M -component GMM to approximate it. This problem can be considered as an unsupervised learning in which the particles are generated by the individual components of the Gaussian mixture distribution and we don't have the knowledge of which particle was generated by which component of such mixture distribution [19], [156]. Therefore, our task is to map N particles into M classes and calculate the GMM parameters $\left\{P(j), \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right\}$ $j = 1,\ldots,M$ for each of the M mixture components.

Normally, the maximum likelihood (ML) method can be adopted to estimate the GMM parameters. In ML method, the negative log-likelihood for the right-hand side of Equation 6.2 can be written as

$$L(\theta) = \sum_{i=1}^{N} \ln \left\{ \sum_{j=1}^{M} p\left(\mathbf{x}^{*i} \middle| j, \boldsymbol{\theta}_j\right) \right\} \tag{6.3}$$

where $\left\{\mathbf{x}^{*i}\right\}_{i=1}^{N}$ are particles (the time index k is also dropped for the sake of clarity) and $\boldsymbol{\theta}_j = \left\{P(j), \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right\}_{j=1}^{M}$ are GMM parameters to be calculated. By differentiating Equation 6.3 with respect to $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$, we could obtain the ML solution for estimating the GMM parameters as follows [156]:

$$\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{N} P\left(j \middle| \mathbf{x}^{*i}\right) \mathbf{x}^{*i}}{\sum_{i=1}^{N} P\left(j \middle| \mathbf{x}^{*i}\right)} \tag{6.4}$$

$$\boldsymbol{\Sigma}_j = \frac{\sum_{i=1}^{N} P\left(j \middle| \mathbf{x}^{*i}\right)\left(\mathbf{x}^{*i} - \boldsymbol{\mu}_j\right)\left(\mathbf{x}^{*i} - \boldsymbol{\mu}_j\right)^T}{\sum_{i=1}^{N} P\left(j \middle| \mathbf{x}^{*i}\right)} \tag{6.5}$$

$$P(j) = \frac{1}{N} \sum_{i=1}^{N} P\left(j \middle| \mathbf{x}^{*i}\right) \tag{6.6}$$

However, the quantity $\sum_{i=1}^{N} P\left(j \middle| \mathbf{x}^{*i}\right)$ in Equations 6.4, 6.5 and 6.6 is difficult to compute analytically. Hence, the GMM parameters $\left[P(j), \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right]$, $j = 1, \ldots, M$ can not be obtained explicitly by the above ML method. Therefore, we need to resort to the expectation maximization (EM) algorithm [134] to compute the GMM parameters.

6.2.2 Expectation Maximization (EM) Algorithm for Parameter Estimation

The EM algorithm is an iterative method for estimating the unknown parameters from an incomplete dataset [158]. Here, the term "incomplete" refers to the "hidden" information in the dataset. In the above GMM approximation problem, the "hidden" information refers to the knowledge of mapping between the GMM components and the particles, i.e. which component $j,\ j \in (1, \ldots M)$ of GMM generated which particle $i,\ i \in (1, \ldots N)$. This section briefly reviews the generic EM method for unknown parameters estimation and next section will derive the EM algorithm for estimating the GMM parameters for constructing the GMM model to approximate the probability density function of the target state $p(\mathbf{x}_k \mid \mathbf{z}_{0:k})$.

Let's define a complete dataset $\{x_i, y_i\}_{i=1}^{N}$ which consists of two subsets of univariate variables: a subset of N univariate variables $\{x_i\}_{i=1}^{N}$ whose values are known; and a subset of N hidden univariate variables $\{y_i\}_{i=1}^{N}$ whose values aren't known. It is assumed that both $\{x_i\}_{i=1}^{N}$ and $\{y_i\}_{i=1}^{N}$ are independently and identically distributed (i.i.d). The unknown parameters to be decided by EM algorithm for the above dataset is denoted as θ. The negative log-likelihood of the incomplete dataset $\{x_i\}_{i=1}^{N}$ and the complete dataset $\{x_i, y_i\}_{i=1}^{N}$ over this unknown parameter θ are as follows:

$$L(\theta) = \sum_{i=1}^{N} \ln p(x_i \mid \theta) \tag{6.7}$$

$$L^{\text{comp}}(\theta) = \sum_{i=1}^{N} \ln p(x_i, y_i \mid \theta) \tag{6.8}$$

where $p(x_i, y_i \mid \theta)$ is the joint probability density function of both known and hidden variables given the parameter θ. In EM algorithms, we aim to maximize $L(\theta)$. However, we may not be able to explicitly maximize it. Instead, we try to maximize $L^{\text{comp}}(\theta)$. Unfortunately, due to the unknown variables $\{y_i\}_{i=1}^{N}$, $L^{\text{comp}}(\theta)$ cannot be maximized directly either. To tackle this difficulty, we take the expectation of Equation 6.8 with respect to some joint distribution over all y_i, denoting as $q(\mathbf{y})$ where $\mathbf{y} = \{y_i\}_{i}^{N}$ as follows:

$$\begin{aligned}\varepsilon^{\text{comp}}(\theta) &= E_{q(\mathbf{y})}\left[\sum_{i=1}^{N}\ln p(x_i, y_i \mid \theta)\right] \\ &= E_{q(\mathbf{y})}\left[\sum_{i=1}^{N}\ln\{p(y_i \mid x_i, \theta)\, p(x_i \mid \theta)\}\right] \\ &= \sum_{i=1}^{N}\int_{-\infty}^{\infty} q(\mathbf{y}) \ln p(y_i \mid x_i, \theta)\, d\mathbf{y} + \sum_{i=1}^{N}\int_{-\infty}^{\infty} q(\mathbf{y}) \ln p(x_i \mid \theta)\, d\mathbf{y} \\ &= \sum_{i=1}^{N}\int_{-\infty}^{\infty} q(\mathbf{y}) \ln p(y_i \mid x_i, \theta)\, d\mathbf{y} + \sum_{i=1}^{N}\ln p(x_i \mid \theta) \\ &= \sum_{i=1}^{N}\int_{-\infty}^{\infty} q(\mathbf{y}) \ln p(y_i \mid x_i, \theta)\, d\mathbf{y} + L(\theta)\end{aligned} \tag{6.9}$$

Recall the i.i.d assumption of the “hidden” variables $\{y_i\}_{i=1}^{N}$ made earlier in this section, we have $q(\mathbf{y}) = \prod_{m=1}^{N} q_m(y_m)$ and thus Equation 6.9 becomes

$$\begin{aligned}\varepsilon^{\text{comp}}(\theta) &= \sum_{i=1}^{N}\int_{-\infty}^{\infty}\left[\prod_{m=1}^{N} q_m(y_m)\right]\ln p(y_i \mid x_i, \theta)\, d\mathbf{y} + L(\theta) \\ &= \sum_{i=1}^{N}\int_{-\infty}^{\infty} q_i(y_i) \ln p(y_i \mid x_i, \theta)\, dy_i + L(\theta)\end{aligned} \tag{6.10}$$

In Equation 6.10, all terms in the product for which $m \neq i$ can be integrated independently and being densities evaluate to unity, thus we made the simplification in the equation.

It can be seen that all terms in Equation 6.10 are functions of the unknown parameters θ. Now we wish to adapt those parameters for the purpose of increasing the likelihood $L(\theta)$. Denoting the values of the parameters θ at the τ -th step as θ^{τ} and let the joint distribution over all hidden variables y_i take the following form

$$q_i(y_i) = p\left(y_i \middle| x_i, \theta^{\tau}\right) \tag{6.11}$$

Then we can rewrite Equation 6.10 for the $(\tau+1)$-th step as

$$\varepsilon^{\text{comp}}\left(\theta^{\tau+1}\right) = \sum_{i=1}^{N}\int_{-\infty}^{\infty} p\left(y_i \middle| x_i, \theta^{\tau}\right) \ln p\left(y_i \middle| x_i, \theta^{\tau+1}\right) dy_i + L\left(\theta^{\tau+1}\right) \tag{6.12}$$

In Equation 6.12, if we adjust $\theta^{\tau+1}$ to maximize $\varepsilon^{\text{comp}}\left(\theta^{\tau+1}\right)$ on the left-hand side, since the first term on the right-hand side is maximized when $\theta^{\tau+1} = \theta^{\tau}$, it can only decrease as

$\theta^{\tau+1}$ changes. To maintain equality of Equation 6.12, the second term $L\left(\theta^{\tau+1}\right)$ must increase. Therefore, maximizing $\varepsilon^{\text{comp}}\left(\theta^{\tau+1}\right)$, which may be able to do explicitly, will increase the true likelihood $L\left(\theta^{\tau+1}\right)$. As mentioned earlier in this section, $L\left(\theta^{\tau+1}\right)$ is the likelihood we are interested in. The general EM algorithm is based on this reasoning and it consists of following two steps [134], [156]:

(1) The *Expectation Step* to computing the expectation of the complete data log-likelihood $\varepsilon^{\text{comp}}\left(\theta^{\tau}\right)$, where the expectation is computed with respect to the distributions $p\left(y_i \mid x_i, \theta^{\tau}\right)$ using the current values of the parameters θ^{τ} .

(2) The *Maximization Step* to determining the new values of parameters $\theta^{\tau+1}$ by maximizing $\varepsilon^{\text{comp}}\left(\theta^{\tau}\right)$which is computed in the E-step.

The above two steps will be repeated until the likelihood reaches to the maximum.

6.2.3 EM Algorithm for Estimating the GMM Parameters

For the GMM parameters estimation problem defined earlier in this section, the dataset is regarded as incomplete because we do not know which GMM component $j,\ j \in (1, \dots M)$ generated a given data point $i,\ i \in (1, \dots N)$. Thus, in GMM the hidden variables $\{y_i\}_{i=1}^{N}$ denotes the generating component and each y_i takes integer values range over $1, \dots M$. For the log-likelihood of the complete dataset (Equation 6.8), we take its expectation with respect to the distribution $q(\mathbf{y}) = \prod_{i=1}^{N} P\left(y_i \middle| x_i, \theta^{\tau}\right)$. Since y_i is a discrete variable, the expectation over all $\{y_i\}_{i=1}^{N}$ is simply a combination of N sums:

$$
\begin{aligned}
\varepsilon^{\text{comp}}\left(\theta^{\tau+1}\right) &= E_{q(\mathbf{y})}\left[\sum_{i=1}^{N} \ln\left\{P\left(y_i \middle| x_i, \theta^{\tau+1}\right) p\left(x_i \middle| \theta^{\tau+1}\right)\right\}\right] \\
&= \sum_{i=1}^{N}\left[\sum_{y_1=1}^{M} \cdots \sum_{y_N=1}^{M} \prod_{m=1}^{N} P\left(y_m \middle| x_m, \theta^{\tau}\right) \ln p\left(x_i \middle| y_i, \theta^{\tau+1}\right) P\left(y_i \middle| \theta^{\tau+1}\right)\right] \\
&= \sum_{i=1}^{N}\left\{\sum_{y_1=1}^{M} \cdots \sum_{y_{i-1}=1}^{M} \sum_{y_{i+1}=1}^{M} \cdots \sum_{y_N=1}^{M} \prod_{m \neq i}^{N} P\left(y_m \middle| x_m, \theta^{\tau}\right)\right. \\
&\qquad \left. \times \sum_{y_i=1}^{M} P\left(y_i \middle| x_i, \theta^{\tau}\right) \ln p\left(x_i \middle| y_i, \theta^{\tau+1}\right) P\left(y_i \middle| \theta^{\tau+1}\right)\right\} \\
&= \sum_{i=1}^{N} \sum_{y_i=1}^{M} P\left(y_i \middle| x_i, \theta^{\tau}\right) \ln p\left(x_i \middle| y_i, \theta^{\tau+1}\right) P\left(y_i \middle| \theta^{\tau+1}\right)
\end{aligned}
\tag{6.13}
$$

In the derivation of Equation 6.13, we make use of the property $\sum_{y_m=1}^{M} P\left(y_m \middle| x_m, \theta^{\tau}\right) = 1$.

The above computation of $\varepsilon^{\mathrm{comp}}\left(\theta^{\tau+1}\right)$ is the Expectation-step of the EM algorithm; in the Maximization- step, we maximize $\varepsilon^{\mathrm{comp}}\left(\theta^{\tau+1}\right)$ with respect to the parameters $\theta^{\tau+1}$. So if we differentiate Equation 6.13 and set the derivatives to zero, we can obtain the iterative expression of $\theta^{\tau+1}$. The above derivation of EM algorithms is for the univariate dataset. However, the results can be extended to the multivariate dataset [156].

Starting with an initial trial values of the GMM parameters $\boldsymbol{\theta}_j^0 = \left\{ \boldsymbol{\mu}_j^0, \boldsymbol{\Sigma}_j^0, P^0(j) \right\}_{j=0}^{M}$, the EM algorithm for GMM parameter estimation is performed as follows (note we replace x_i with $\mathbf{x}^{*i}$ to denote the particles) [156]:

1. *Expectation Step:* computing the probability density function $P\left(j \middle| \mathbf{x}^{*i}, \boldsymbol{\theta}^{\tau}\right)$ for each particle i, $i \in (1, \ldots N)$ and each mixture component j, $j \in (1, \ldots M)$ using the current values of the parameters $\left\{ \boldsymbol{\mu}_j^{\tau}, \boldsymbol{\Sigma}_j^{\tau}, P^{\tau}(j) \right\}$

$$P\left(j \middle| \mathbf{x}^{*i}, \boldsymbol{\theta}^{\tau}\right) = \frac{P\left(\mathbf{x}^{*i} \middle| \boldsymbol{\mu}_j^{\tau}, \boldsymbol{\Sigma}_j^{\tau}\right) P^{\tau}(j)}{\sum_{j=1}^{M} P\left(\mathbf{x}^{*i} \middle| \boldsymbol{\mu}_j^{\tau}, \boldsymbol{\Sigma}_j^{\tau}\right) P^{\tau}(j)} \tag{6.14}$$

2. *Maximization Step:* computing the re-estimate values for the parameters $\left\{ \boldsymbol{\mu}^{\tau+1}, \boldsymbol{\Sigma}^{\tau+1}, P^{\tau+1}(j) \right\}$

$$\boldsymbol{\mu}_j^{\tau+1} = \frac{\sum_{i=1}^{N} P\left(j \middle| \mathbf{x}^{*i}, \boldsymbol{\theta}^{\tau}\right) \mathbf{x}^{*i}}{\sum_{i=1}^{N} P\left(j \middle| \mathbf{x}^{*i}, \boldsymbol{\theta}^{\tau}\right)} \tag{6.15}$$

$$\boldsymbol{\Sigma}_j^{\tau+1} = \frac{\sum_{i=1}^{N} P\left(j \middle| \mathbf{x}^{*i}, \boldsymbol{\theta}^{\tau}\right) \left(\mathbf{x}^{*i} - \boldsymbol{\mu}_j^{\tau+1}\right) \left(\mathbf{x}^{*i} - \boldsymbol{\mu}_j^{\tau+1}\right)^T}{\sum_{i=1}^{N} P\left(j \middle| \mathbf{x}^{*i}, \boldsymbol{\theta}^{\tau}\right)} \tag{6.16}$$

$$P^{\tau+1}(j) = \frac{1}{N}\sum_{i=1}^{N} P\left(j \middle| \mathbf{x}^{*i}, \boldsymbol{\theta}^{\tau}\right) \tag{6.17}$$

The above iteration is repeated and the likelihood function increased until a local maximum is achieved. However, there is no guarantee that the EM algorithm converges to a global maximum. For example, if the likelihood functions have multiple maxima, the EM algorithm may converge to a local maximum. Further discussion of convergence of EM algorithm can be found in [158] and won't be detailed here. In the simulation conducted in this chapter, we adopt five components GMM and set the stopping condition for the EM iterations as the relative change in log-likelihood falling below 0.01 or the number of iterations reaching 20 for an acceptable level of performance.

With the GMM approximation, the target state estimate can be propagated through the transmission of a low volume dataset of GMM parameters $\left[P(j), \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right],\ j = 1,...,M$ rather than a high volume dataset of particles. Taking PF-PDAF algorithm as an example, in the original PF-PDAF 1000 particles need to be transmitted; however, with the use of five components GMM, in the distributive PF-PDAF only 15 GMM parameters need to be transmitted.

After receiving the parameters of GMM, the new cluster leader re-draws particles from the GMM. This is performed straightforward: we first select a component j at random according to the mixing coefficients $P(j)$, and then sample from the relevant individual Gaussian density $p\left(\mathbf{x}_k \middle| j\right) \sim N\left(\mathbf{x}_k; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)$ to generate particles.

6.3 Sensing Nodes Selection Scheme

The introduction of distributive tracking algorithm brings with it a need to select sensing nodes for a sensor cluster. With a partition of the sensor field, each cluster leader only activates a subset of all sensing nodes. In the interest of further resource economisation, the cluster leader only activates three sensing nodes that lie closest to the predicted location of the target instead of all sensing nodes within the cluster and collects their measurements for target state estimate. Such sensing node selection scheme is intuitively appealing since the SNR is stronger for sensing nodes close to the target which typically implies that information obtained from these sensing nodes are more accurate. However, this strategy does not take into account the energy consumption measure and thus may not help the distributive tracking algorithms to achieve energy efficiency in a wireless sensor network.

Moreover, there exists many combinations of three sensing nodes within a sensor cluster; and the combination of the three closest sensing nodes may not achieve the best balance between tracking accuracy and energy consumption.

This section describes a composite objective function that combines an information utility measure with an energy consumption measure. By minimizing this composite objective function, the cluster leader dynamically selects and activates a combination of three sensing nodes at the beginning of each time step, such that this combination can provide higher information utility and consumes less energy. In turn, the distributive tracking algorithms can attain desirable tracking accuracy while still conserve the energy in wireless sensor networks.

6.3.1 Problem Formulation of Sensing Nodes Selection

In the distributive tracking algorithms, there are two inter-related components: the tracking component and the sensing nodes selection component. Both components are implemented on the cluster leader. At the beginning of the k-th time step, the selection component selects a subset of sensing nodes within the cluster based on the predicted target position [6] and the positions of the sensing nodes in the cluster and the remaining energy in the sensing nodes. The tracking component then invokes only those selected sensing nodes, collects their measurements and updates the estimate of target state for the k-th time step. Figure 6.2 depicts the above process.

It needs to be emphasized that due to the distributive nature of the wireless sensor networks, the sensing nodes selection has to be done by the cluster leader without the explicit knowledge of the measurements residing at each sensing node at the beginning pf each time step. Hence, the sensing nodes selection decision has to be made solely on the sensing nodes' characteristics (i.e. the positions of the sensing nodes and the remaining energy) and the predicted contribution made by the sensing nodes (i.e. the sensing nodes' contribution to the target state estimation). The assumption has already been made in Chapter 4 that the cluster leaders know the positions of the sensing nodes. The cluster leader gets the knowledge of the remaining energy of sensing nodes during the sensor cluster formation (Refer to Chapter 3). The sensing nodes selection scheme developed in

[6] The predicted target's position is computed through the system model by using the target state estimate at the $(k-1)$-th time step, i.e. $\mathbf{x}^{prediction} = \mathbf{A}_k \mathbf{x}_{k-1}$. The target state estimate $\mathbf{x}_{k-1}$ is provided by the tracking component at the end of the $(k-1)$-th time step.

this chapter makes use of the PCRLB to predict the contribution that the sensing nodes make to the target state estimate.

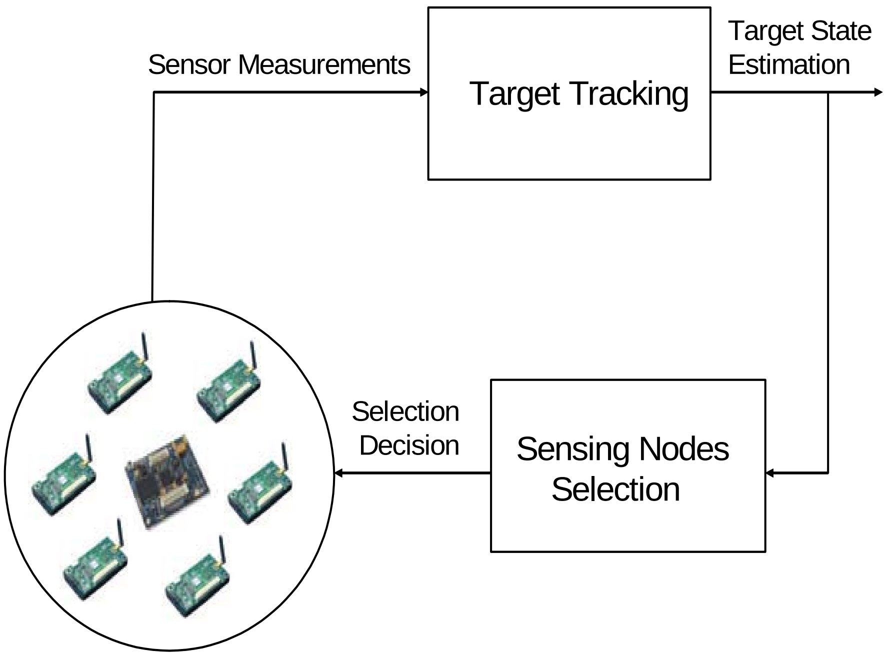


Figure 6.2 Sensing node selection and tracking

In energy-constrained wireless sensor networks, energy is a resource to be conserved to prolong the networks' operational lifetime. Moreover, it is typically the case that the energy cost of communications is orders of magnitudes greater than that of local computation [8]. Therefore, we only consider the energy consumption in communication in designing the sensing nodes selection scheme. However, other sources of energy consumptions such as activating the sensing nodes, taking the measurements, and executing the tracking algorithm may also be incorporated without changing the structure of the sensing nodes selection scheme developed in this chapter.

To assess the energy consumption in communication, the following radio and signal processing model is employed [8]:

$$E_{tx} = e_{sp} + d^{\alpha} . e_{out} \tag{6.18}$$

where e_{sp} is the energy cost of the signal processing, e_{out} is the radio transmitter output energy and a simple geometric path loss model is assumed, so that the propagation loss is proportional to

$$\frac{1}{d^{\alpha}} \qquad 2 \le \alpha \le 4 \tag{6.19}$$

where α is the path loss exponent and d is the transmission distance in meter.

It is assumed that any sensing node within the sensor cluster can transmit its measurements to cluster leader through the one-hop communication (refer to Chapter 3). Based on the Equations 6.18 and 6.19, the energy consumption for the n-th sensing node to transmit its measurement to the cluster leader is modelled as being proportional to the square distance between itself and the cluster leader:

$$c_k^n \propto \| \mathbf{r}_k^n - \mathbf{r}_k^{leader} \|^2 \tag{6.20}$$

where c_k^n denotes the energy consumption of the n-th sensing node when it transmits the measurement to the cluster leader at the k-th time step, $\mathbf{r}_k^n$ and $\mathbf{r}_k^{leader}$ denote the position of the n-th sensing node and the cluster leader, respectively.

To take into account the information utility of a combination of sensing nodes, the posterior Cramer-Rao lower bound (PCRLB) is adopted since it can provide a theoretical lower bound on the tracking accuracy that tracking algorithms could attain. However, it needs to emphasis that the PCRLB is computed without using the measurements obtained by the sensing nodes at the k-th time step since these measurements are not available when the cluster leader makes the sensing node selection decision (Refer to Chapter 4). For the sake of integrity, the equations of PCRLB calculation for the k-th time step are re-written as follows:

$$\begin{aligned} PCRLB &= \left(\mathbf{J}_k\right)^{-1} \\ &= \left[\mathbf{D}_{k-1}^{22} - \mathbf{D}_{k-1}^{21}\left(\mathbf{J}_{k-1} + \mathbf{D}_{k-1}^{11}\right)^{-1}\mathbf{D}_{k-1}^{12} \right]^{-1} \end{aligned} \tag{6.21}$$

where

$$\mathbf{D}_{k-1}^{11} = \mathbf{A}_{k-1}^{T}\mathbf{Q}_{k-1}^{-1}\mathbf{A}_{k-1} \tag{6.22}$$

$$\mathbf{D}_{k-1}^{12} = -\mathbf{A}_{k-1}^{T}\mathbf{Q}_{k-1}^{-1} \tag{6.23}$$

$$\mathbf{D}_{k-1}^{21} = -\mathbf{Q}_{k-1}^{-1}\mathbf{A}_{k-1}^{T} \tag{6.24}$$

$$\mathbf{D}_{k-1}^{22} = \mathbf{Q}_{k-1}^{-1} + \mathbf{J}_{\mathbf{z}}(k) \tag{6.25}$$

$$\mathbf{J}_z(k) = \sum_{n}^{N_0} q_2^n \left(\mathbf{H}_k^n\right)^T \left(\sigma_k^n\right)^{-2} \mathbf{H}_k^n \tag{6.26}$$

where N_0 denotes a combination of N_0 sensing nodes, q_2^n is the information reduction factor (IRF) of the n-th sensing node, $\mathbf{H}_k^n$ is the Jacobian for the n-th sensing node, σ_k^n is the standard deviation of the measurement noise at the n-th sensing node, $\mathbf{A}_{k-1}$ is the system transition matrix, and $\mathbf{Q}_{k-1}$ is the covariance of the process noise. The calculation of the above parameters can be found in Chapters 4 and 5.

In sensing nodes selection, the ultimate goal is to select a subset of N_0 sensing nodes out of the total N_s sensing nodes in the current sensor cluster. This can be treated as a constrained optimization problem. Let $\mathbf{a}_k$ denote the combination of N_0 sensing nodes (named as sensing option) at the k-th time step. The communication cost associated with sensing option $\mathbf{a}_k$ is denoted as $\Theta(\mathbf{a}_k)$ and the information utility associated with sensing option $\mathbf{a}_k$ is denoted as $PCRLB(\mathbf{a}_k)$. Two types of constrained optimization problems for sensing nodes selection can be formulated:

(1) *Minimizing the communication cost*

The objective is to activate the lowest communication cost combination of N_0 sensing nodes that maintains a desirable accuracy in the target state estimate. It involves obtaining the lowest communication cost sensing option $\mathbf{a}_k$ at the k-th time step such that $PCRLB(\mathbf{a}_k)$ does not exceed a desired threshold $PCRLB(Th)$

$$\mathbf{a}_k^{opt} = \arg\min \Theta(\mathbf{a}_k)$$

$$\text{such that } PCRLB(\mathbf{a}_k) - PCRLB(Th) \le 0 \tag{6.27}$$

(2) *Minimizing the tracking error*

The objective is to activate the combination of N_0 sensing nodes that minimizes the tracking error subject to the constraints on the communication cost. It involves obtaining the sensing option $\mathbf{a}_k$ at the k-th time step such that $PCRLB(\mathbf{a}_k)$ is minimized subject to a constraint on the communication cost $\Theta(th)$

$$\mathbf{a}_k^{opt} = \arg\min PCRLB\left(\mathbf{a}_k\right)$$

$$\text{such that } \Theta\left(\mathbf{a}_k\right) - \Theta(th) \leq 0 \tag{6.28}$$

6.3.2 Brief Review of Sensing Nodes Selection for Target Tracking Applications

Sensing nodes selection is normally referred to as sensor scheduling in the literature. Recently, it has received considerable attentions in target tracking applications [148]-[151], [157]. Several sensor scheduling frameworks and optimization techniques have been developed and a brief review of these works is supplied as follows.

Sensor scheduling algorithms based on the predicted error covariance matrix were developed in [119], [157]. In [119], a PCRLB based framework was used to schedule the deployment of sensors to control the tracking accuracy. In [157], a scheduling algorithm was developed for linear Gaussian models in which the minimum number of sensors was selected to drive the error covariance matrix to a desired matrix. However, these two algorithms did not consider the energy consumption of sensors.

In [150] and [151], the sensor scheduling problem was treated as a constrained optimization problem in which a performance metric (i.e. the predicted tracking performance or the energy consumption) is optimized under certain constraints (i.e. the energy consumption or the predicted tracking performance). The optimization problems were then posed as the binary (0-1) convex programming problems which can be further simplified to 0-1 mixed integer programming problems under certain assumptions. The (0-1) convex programming problems and 0-1 mixed integer programming problems were solved by using outer approximation and linear programming relaxation based branch-and-bound algorithms. However, the sensor scheduling algorithms developed in [150] and [151] require complex algorithmic implementation and introduce a heavy computation burden.

In [149], a lightweight sensor scheduling scheme was proposed. Instead of posing the sensor scheduling as the complex constrained optimization problem, sensor scheduling is achieved by using an objective function which incorporates both the information utility measure and the energy consumption measure. However, since the authors in [149] adopted the single leader-based tracking scheme, only one sensor node is considered in their sensor scheduling algorithm. In this chapter, we develop a similarly lightweight, composite objective function based sensor scheduling (sensing nodes selection) scheme, but extended to scheduling more than one sensing nodes.

6.3.3 Sensing Nodes Selection by Adopting a Composite Objective Function

The proposed composite objective function comprises of an information utility measure and an energy consumption measure. The information utility measure is based on PCRLB and the energy consumption measure is decided by the communication cost between the sensing node and the cluster leader.. Instead of posing the sensing node selection problem as the constrained optimization problem in Equations 6.26 and 6.27, we solve the sensing node selection problem by adopting a weighted composite objective function in which the information utility measure and the energy consumption measure carry the different weights which are decided by the requirements of the tracking applications. The composite objective function takes the following form:

$$\Psi\left(\mathbf{a}_k\right) = \alpha\, PCRLB\left(\mathbf{a}_k\right) + (1-\alpha)\Theta\left(\mathbf{a}_k\right) \tag{6.29}$$

where $PCRLB\left(\mathbf{a}_k\right)$ and $\Theta\left(\mathbf{a}_k\right)$ are the information utility measure and the energy consumption measure (communication cost) of the sensing option $\mathbf{a}_k$, respectively. The weight $\alpha \in [0,1]$ is used to balance the information utility measure $PCRLB\left(\mathbf{a}_k\right)$ and the energy consumption measure $\Theta\left(\mathbf{a}_k\right)$ in the composite objective function. The magnitude of α depends on the requirements of tracking applications. The adoption of Equation 6.29 for sensing nodes selection avoids the complicated computation in explicitly solving the constrained optimization problem of Equations 6.27 and 6.28.

For the problem of selecting N_0 sensing nodes from total N_s sensing nodes within a sensor cluster, there will be $\binom{N_s}{N_0} = \frac{N_s!}{N_0!\,(N_s - N_0)}$ possible combinations. Now the objective is to find a optimal combination of N_0 sensing nodes that minimizes $\Psi\left(\mathbf{a}_k\right)$, i.e.

$$\mathbf{a}_k^{opt} = \arg\min \Psi\left(\mathbf{a}_k^j\right), \quad j \in \binom{N_s}{N_0} \tag{6.30}$$

$\Psi\left(\mathbf{a}_k^j\right)$ is the composite function for the j-th combination of N_0 sensing nodes out of total N_s sensing nodes.

The process of choosing the best combination of sensing nodes is straightforward: the composite objective functions associated with different sensing nodes combinations are computed according to Equation 6.29; and the sensing nodes combination which leads to the smallest value of the objective function is selected by the cluster leader. The sensing nodes in this combination will then be activated to take sensing action and transmit their measurements to the cluster leader.

It is noted that combination number will become large with the increasing number of sensing nodes in the sensor cluster. To combat this exponential increase, the sensing node selection scheme proposed in this chapter is performed in two steps: in the first step, a small portion of the sensing nodes, say, N_m sensing nodes amongst all N_s sensing nodes within the sensor cluster are selected solely based on their locations to the predicted target position; in the second step, the composite objective function (Equation 6.28) is calculated for all the possible combination of N_0 sensing nodes out of N_m sensing nodes. However, we do not expect a very large number of sensing nodes within one sensor cluster because sensor cluster is formed in a relative smaller region with relative smaller number of sensor nodes (refer to Chapter 3). In the simulation conducted in this chapter, at each time step, five sensing nodes that are closest to the predicted target location are firstly selected, i.e., $N_m = 5$. Then the composite objective function for each combination of $N_0 = 3$ sensing nodes out of $N_m = 5$ sensing nodes is calculated. There will be total 10 combinations. Finally, the three sensing nodes in the combination that gives the smallest value of the objective function Ψ_j, $j \in \binom{N_o}{N_m}$ is activated to take sensing action and transmit their measurements to the cluster leader.

In some practical large target tracking applications in wireless sensor networks, it may be required to collect more measurements from a large number of sensing nodes and consequently, the combinations number becomes large. Thus, the above exhaustive computation of all combinations of these sensing nodes may be prohibitive and some search techniques would be needed [151]. However, the implementation of these search techniques is beyond the scope of this thesis.

6.4 Distributive PF, EKPF, and PF-PDAF Tracking Algorithms

By making use of the GMM model and the sensing nodes selection scheme, the original PF, EKPF and PF-PDAF tracking algorithms developed in Chapters 4 and 5 can be readily extended to their distributed forms, i.e. the distributive PF, EKPF and PF-PDAF tracking algorithms. The pseudo-code of the distributive PF, EKPF and PF-PDAF is listed in Algorithm 6.1.

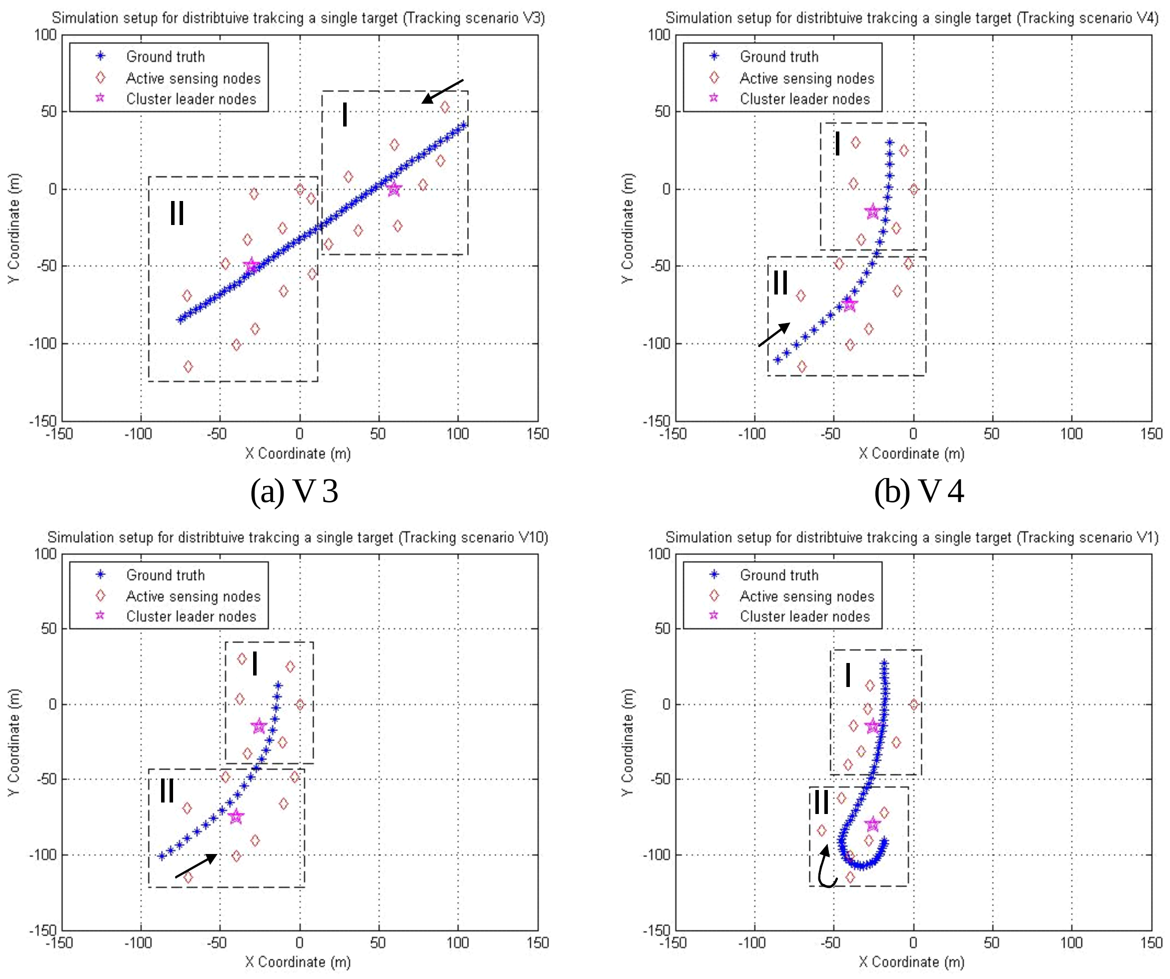
Algorithm 6.1 Distributed PF/EKPF/PF-PDAF Tracking Algorithm

```
1. The cluster leader of sensor cluster $a$ does the following:
   1.1 For $i = 1, 2, ..., N$ draws particle $\mathbf{x}_k^i$ from the initial probability density function
       of the target state $p(\mathbf{x}_0)$.
   1.2 Selects sensing nodes and collects their measurements.
   1.3 Calculates the probability density function $p(\mathbf{x}_k | \mathbf{Z}_{0:k})$ by using
       one of PF/EKPF/PF-PDAF algorithms developed in previous chapters.
   1.4 Computes the GMM approximation of $p(\mathbf{x}_k | \mathbf{Z}_{0:k})$ by Equations 6.14 ~6.17.
   1.5 Forwards GMM parameters to the cluster leader of sensor cluster $b$
       if the target moves out of sensor cluster $a$.

2. The cluster leader of sensor cluster $b$ does the following:
   2.1 For $i = 1, 2, ..., N$, draws particles from GMM to approximate
       the prior probability density function $p(\mathbf{x}_k | \mathbf{Z}_{0:k-1})$ at the previous time step.
   2.2 Selects sensing nodes and collects their measurements.
   2.3 Calculates the probability density function $p(\mathbf{x}_k | \mathbf{Z}_{0:k})$ by using
       one of PF/EKPF/PF-PDAF Algorithms developed in previous chapters.
   2.4 Computes the GMM approximation of $p(\mathbf{x}_k | \mathbf{Z}_{0:k})$ by Equations 6.14 ~6.17.
   2.5 Forwards GMM parameters to the cluster leader of sensor cluster $c$
       if the target moves out of sensor cluster $b$.

3. The cluster leader of sensor cluster $c$ repeats the above steps 2.1 ~ 2.6.
```

6.5 Simulations

This section conducts extensive simulations to evaluate the performance of three distributive tracking algorithms namely distributive PF, distributive EKPF and distributive PF-PDAF. It also assesses several sensing node selection schemes including the composite function based sensing node selection scheme for the distributive PF-PDAF.

The simulation setup is chosen to be the same as Chapters 4 and 5 as much as possible. Figure 6.3 depicts the simulation setup (Figure 6.3 is a repeat of Figure 5.1 in Chapter 5). The ground vehicle traverses through a two-dimensional (2D) sensor field and four different tracking scenarios are used in the simulation. These tracking scenarios are synthesized with different target trajectories, target dynamics, clutter rates, detection rates and sensing nodes. The target trajectories are digested from a real on-site experiment [23]. Differ from Chapters 4 and 5, the sensor field in each tracking scenario in Figure 6.3 is partitioned into two regions and in each region, a sensor cluster is formed. This sensor field partition is for the evaluation of the distributive tracking algorithms and sensing node selection schemes developed in this chapter. Since two clusters of senor nodes can reveal the key performance of the “belief” propagation using GMM and sensing node selection schemes without introducing heavy computations in simulations, we only choose two sensor clusters and not more sensor clusters in the simulations. However, the distributive tracking algorithms and the sensing nodes selection schemes developed in this chapter can be readily applied to track the target over an arbitrary number of sensor clusters in the wireless sensor networks. Moreover, the above two sensor clusters are formed statically; however, the dynamic formation of sensor clusters can be achieved within the collaborative information processing framework proposed in Chapter 3.



(a) V3 (b) V4

(c) V10 (d) V1

Figure 6.3 Simulation setup for assessing the distributive tracking algorithms (V3, V4, V10 and V1 denote four different tracking scenarios with different target trajectories, target dynamics and active sensing nodes. I and II denote two different sensor clusters)

This section is organized as follows. Subsection 6.5.1 presents the simulation results of the distributive PF and distributive EKPF algorithms. Subsection 6.5.2 presents the simulation results of the distributive PF-PDAF algorithm. Subsection 6.5.3 compares four sensing node selection schemes for the distributive PF-PDAF algorithm, and also presents the simulation results of the distributive PF-PDAF algorithm that adopts the composite objective function for the sensing node selection.

6.5.1 The Simulation Results of Distributive PF and Distributive EKPF Algorithms

This section evaluates the tracking accuracy of distributive PF and distributive EKPF algorithms. In the simulations, at each time step, *the cluster leader selects three sensing nodes that are closest to the predicted target position* and collects their measurement for the update of the “belief”. When the target moves out of the first sensor cluster, the cluster leader passes its “belief” in GMM format (in the simulation, we use five components GMM to approximate the “belief”) to the cluster leader in the second sensor cluster (refer to Chapter 3 for the details of “belief” propagation).

In the simulations, 200 independent Monte Carlo runs are conducted for both distributive PF and distributive EKPF algorithms. In each Monte Carlo run, the target trajectory remains unchanged. However, the measurements obtained at the sensing nodes are synthesized according to the measurement model as defined below:

$$\xi_{t,k}^{n} = \frac{S}{(x_k - x_n)^2 + (y_k - y_n)^2} + \varepsilon_k^n \tag{6.31}$$

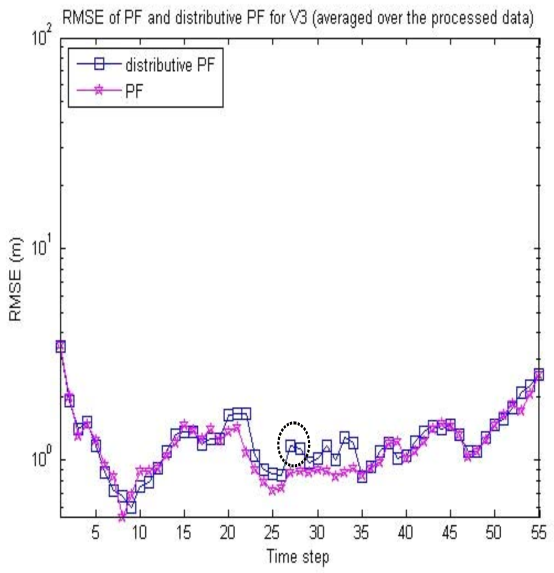
where (x_k, y_k) and (x_n, y_n) are the positions of the target and the n-th sensing node in x- and y-coordinate, respectively. S is the source energy and set to 5000 in the simulations. The background noise is assumed to be Gaussian and set to $\varepsilon_k^n \sim N(0,1),\ n = 1, 2, \ldots, N_s$ for all sensing nodes. Hence, the SNR is 37 dB at the target position. In the simulations, the particles number adopted in the distributive PF is 1000 and in the distributive EKPF is 200. The particles are initialized according to Gaussian with mean vector $\mathbf{x}_{0|0}$ and covariance matrix $\mathbf{P}_{0|0}$ as follows:

$$\mathbf{x}_{0|0} = \mathbf{x}_{truth} + \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \qquad \mathbf{P}_{0|0} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.32}$$

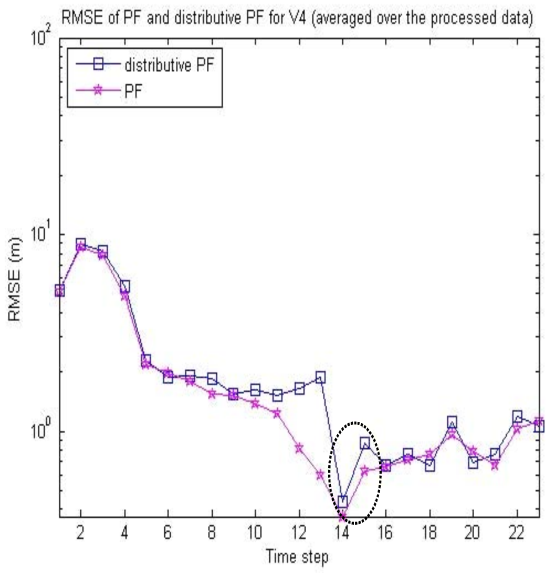
where $\mathbf{x}_{truth}$ is the ground truth at time step $k = 0$ (i.e. the initial position of the target).

In the simulations, the root mean square error (RMSE) is used to assess the performance of the tracking algorithms. As in Chapters 4 and 5, two types of RMSEs are used: the RMSE^n which refers to the RMSE value of the n-th Monte Carlo run that is averaged over all time steps of the tracking task; and the RMSE_k which is the RMSE value of the k-th time step that is averaged over all Monte Carlo runs (200 runs for both PF and EKPF). The definition of the above two RMSEs can be found in Chapters 4 and 5 and will not be repeated here.

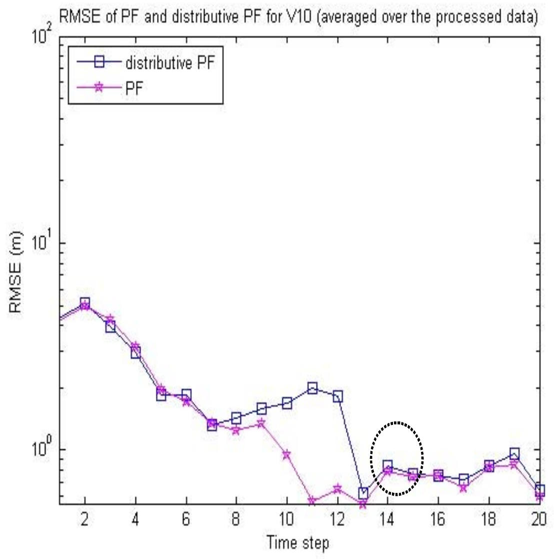
Figure 6.4 presents the simulation results of the distributive PF algorithm and the original PF algorithm developed in Chapter 4 for each of the four tracking scenarios as depicted in Figure 6.3. Figure 6.5 presents the simulation results of the distributive EKPF algorithm and the original EKPF algorithm developed in Chapter 4 for each of the four tracking scenarios. The RMSE_k values in Figures 6.4 and 6.5 are computed by excluding the bottom 50 runs with the lowest RMSE^n values and the top 50 runs with the largest RMSE^n values (It is referred to as the processed data in the figures). The purpose of this data exclusion is to remove the outliner and reduce the bias imposed by the very low or the very large RMSE^n values in some runs of the simulation.



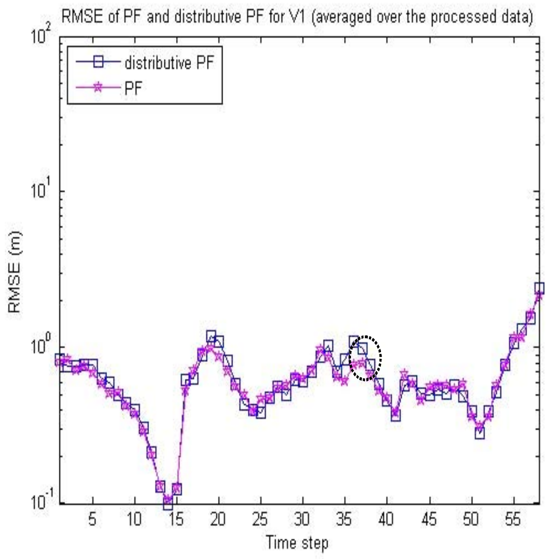
(a) V3



(b) V4



(c) V 10



(d) V1

Figure 6.4 The performance of distributive PF and original PF algorithms of four tracking scenarios (averaged over the processed data)

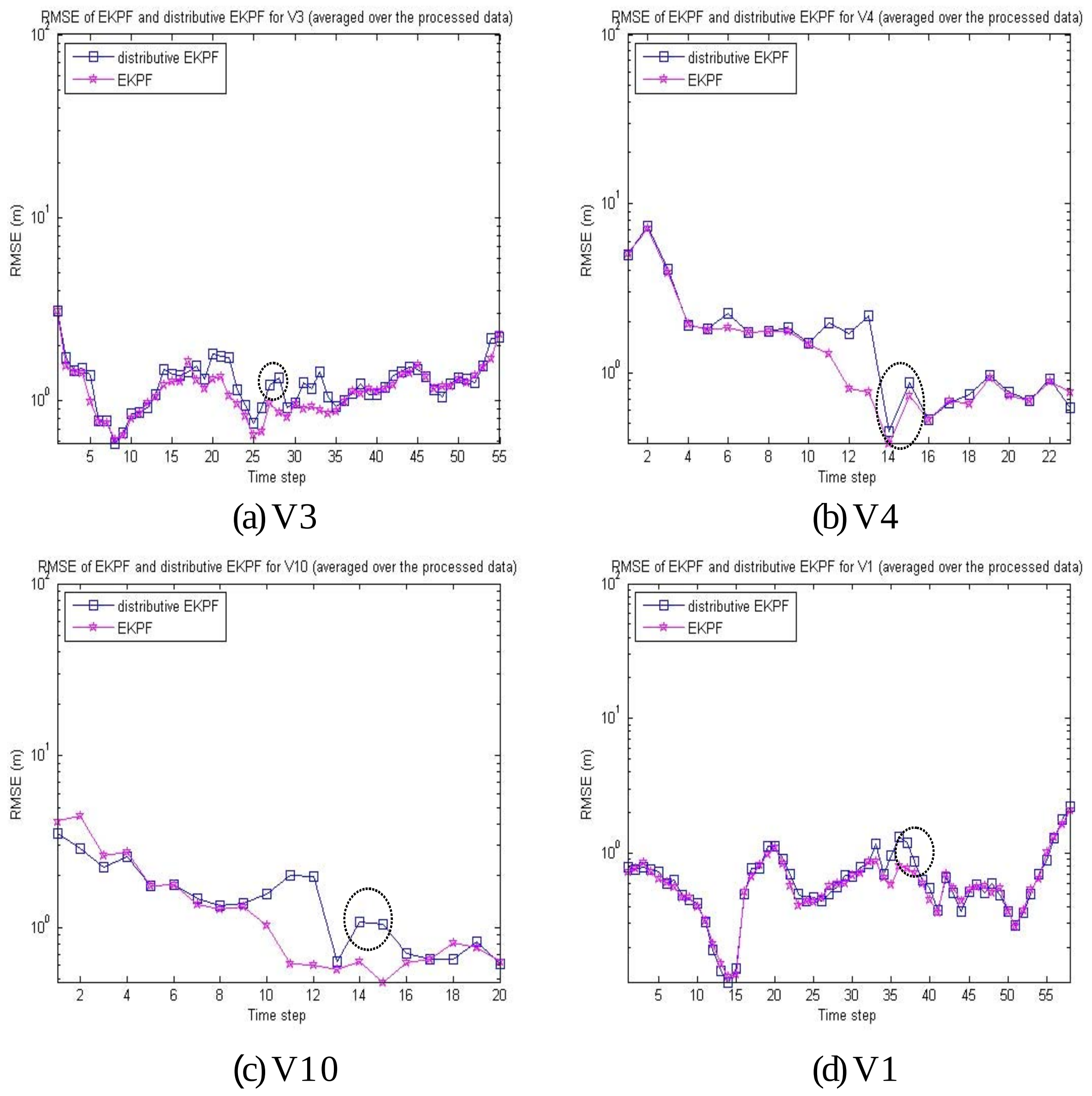


Figure 6.5 The performance of distributive EKPF and original EKPF algorithms of four tracking scenarios (averaged over the processed data)

In Figures 6.4 and 6.5, the dotted circles denote the time steps of "belief" handover between two cluster leaders. It consists of two RMSE_k values: the first RMSE_k value is for the last time step of the first sensor cluster; and the second RMSE_k value is for the first time step of the second sensor cluster. The estimate result obtained at the last time step of the first sensor cluster is approximated by the GMM and then propagated to the second sensor cluster. At the first time step of the second sensor cluster, the cluster leader firstly generates new particles by sampling the above GMM, and then updates its "belief" by using the measurements collected from the three selected sensing nodes in the second cluster. However, the above “belief” handover process might degrade the performance of distributive PF and distributive EKPF algorithms. In V3 tracking scenario (Figures 6.4 (a) and 6.5 (a)), the handover happens at the time steps 27 and 28, it can be seen that the magnitudes of RMSE values of both distributive PF and distributive EKPF algorithms at these two time steps are larger than those of original PF and original EKPF algorithms. In V4 tracking scenario (Figures 6.4 (b) and 6.5 (b)), the handover happens at the time steps 14 and 15, it can also be found the increase in RMSE in both distributive PF and distributive EKPF algorithms. The performance degradation during the “belief” handover time steps in the distributive PF and distributive EKPF algorithms is due to the error introduced by the GMM approximation. However, as evidenced in both figures, after the handover period, the cluster leader in the second sensor cluster collects measurements from the selected sensing nodes to update its “belief” at each time step, and eventually, the distributive PF and distributive EKPF could attain almost the same tracking accuracy as the original PF and original EKPF algorithms developed in Chapter 4.

In tracking scenario V1 (Figures 6.4 (d) and 6.5 (d)), the “belief” handover happens at the time steps 37 and 38 and there is a significant decrease of the magnitude of RMSE from time step 37 to time step 38 in both distributive PF and distributive EKPF algorithm. This decrease is due to the fact that when the target moves toward the boundary of the first sensor cluster, the sensing nodes provide less information (due to their locations to the target) of target. When the target enters the second sensor cluster, although the GMM approximation introduces error in target state estimate, the sensing nodes in the second sensor cluster can provide more accurate information for the target state estimate since they are closer to the target.

From the above discussion, it can be concluded that the degradation in performance of distributive PF and distributive EKPF algorithms is not very significantly; both distributive PF and distributive EKPF algorithms can attain almost the same accurate level as the original PF and original EKPF algorithms. To propagate the estimation results from one cluster leader to another cluster leader, only 15 parameters (both distributive PF and distributive EKPF adopt 5 components GMM) need to be transmitted. In contrast, there will be 1000 particles (PF) or 200 particles (EKPF) need to be transmitted without adopting GMM. Apparently, the adoption of GMM greatly conserves the communication overheads in wireless sensor networks.

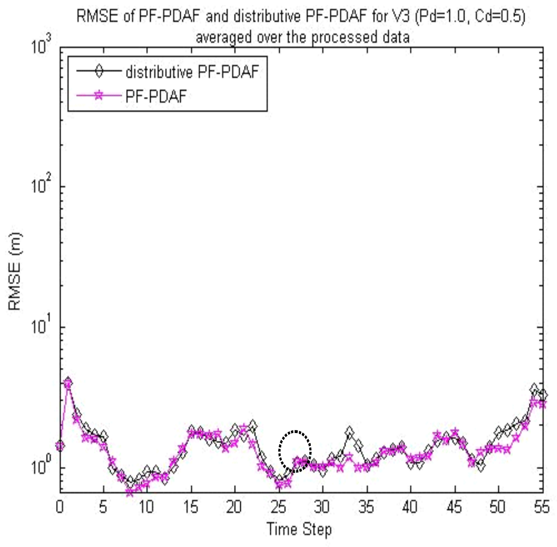
It is necessary to mention that in the above simulations, we did not take into account the energy consumption in sensing nodes selection and only simply select three closest sensing nodes for the distributive PF and distributive EKPF algorithms well as the original PF and original EKPF algorithms. This is because we want to compare the tracking performance of distributive PF and distributive EKPF algorithms with that of original PF and original EKPF algorithm at the same conditions. The simulation results of distributive tracking algorithms that consider both tracking accuracy and energy consumption will be presented in Section 6.5.3.

6.5.2 The Simulation Results of Distributive PF-PDAF Algorithm

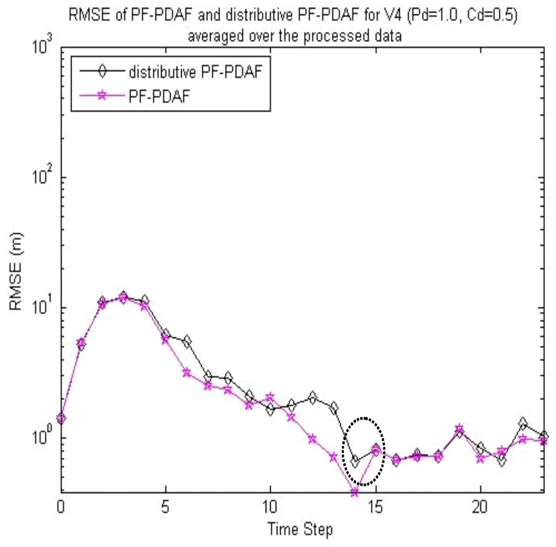
This section conducts simulations to evaluate the distributive PF-PDAF algorithm, and compare its performance with that of the original PF-PDAF algorithm developed in Chapter 5. As in Section 6.5.1, in both distributive PF-PDAF and original PF-PDAF algorithms, at each time step, *the cluster leader selects three sensing nodes that are closest to the predicted target position* and collects their measurement for the target state estimate. The settings of SNR and prior estimate of target state are also the same as those in Section 6.5.1. The number of particles in both distributive PF-PDAF and original PF-PDAF is 1000.

Figures 6.6~6.8 compare the tracking performance of distributive PF-PDAF algorithm and the original PF-PDAF algorithm under varying detection rates ($\mathrm{Pd} = 1, 0.9, 0.8$) while the clutter rate λ is fixed such that $\mathrm{Cd} = \lambda V = 0.5$ ($\mathrm{Cd} = \lambda V$ is the averaged number of clutter originated measurements in the observation space V of a sensing node) for each of the four tracking scenarios as depicted in Figure 6.3.. For each setting, 100 independent Monte Carlo runs are conducted. The RMSE_k values in these figures are computed by excluding the top 25 runs with the largest RMSE^n values and the bottom 25 runs with the lowest RMSE^n values (this is referred to as the processed data in the figures).

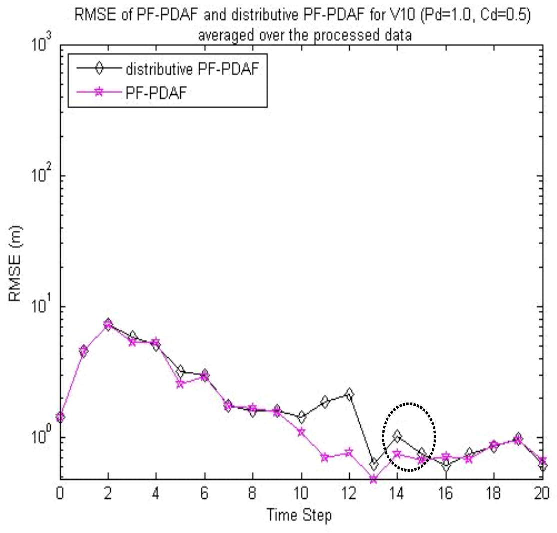
Figure 6.9 depicts the RMSE values of distributive PF-PDAF algorithm for tracking scenario V3 under different detection rates $\mathrm{Pd} = 1, 0.9, 0.8, 0,7$ with the clutter rate fixed at $\mathrm{Cd} = \lambda V = 0.5$. Figure 6.9 (a) shows the RMSE_k value of each time step that averaged over 100 independent runs. Figure 6.9 (b) also shows the RMSE_k value of each time step; however, instead of being averaged over all 100 runs, the RMSE_k value in Figure 6.9 (b) is computed by excluding the top 25 runs with the largest RMSE^n values and the bottom 25 runs with the lowest RMSE^n values. Figures 6.9 (c) and 6.5 (d) are the RMSE^n values of all 100 independent runs.



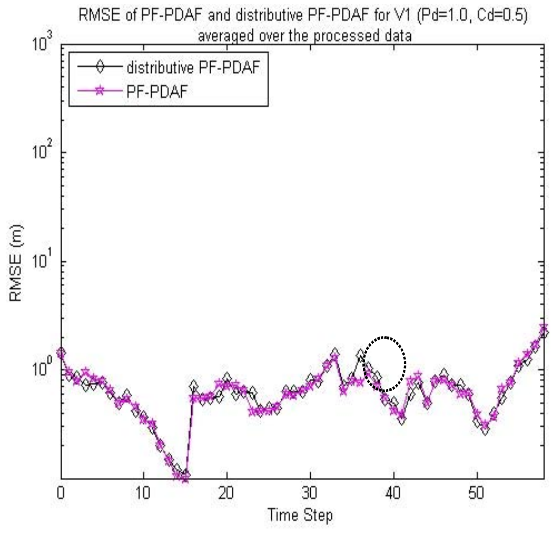
(a) V3



(b) V4

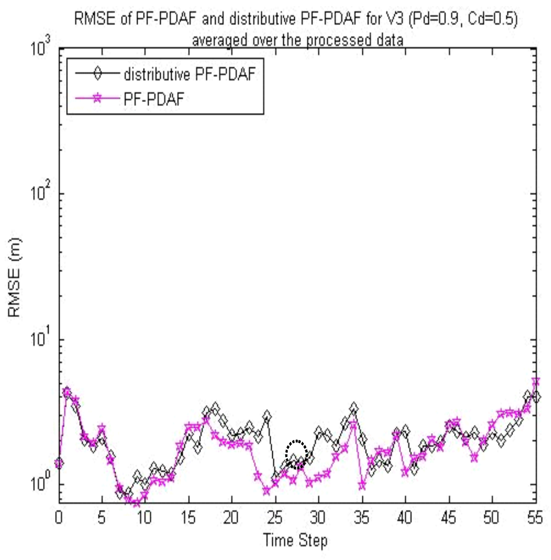


(c) V 10

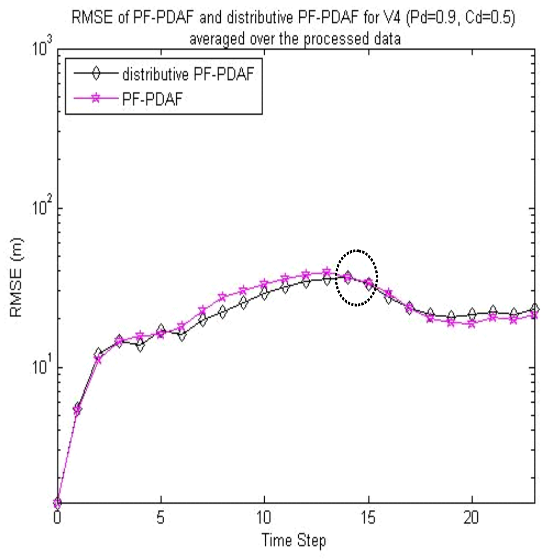


(d) V1

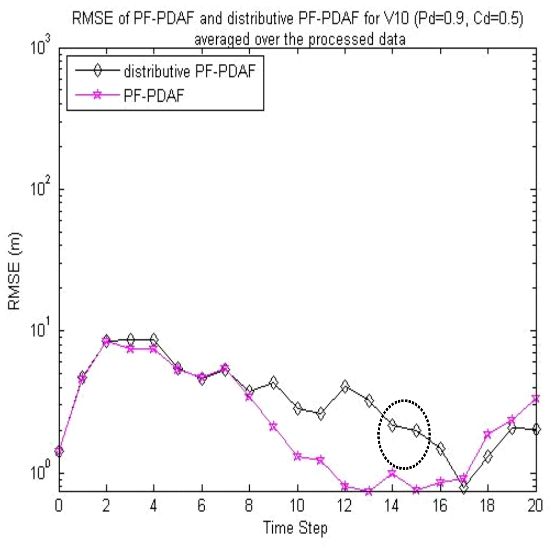
Figure 6.6 The performance of distributive PF-PDAF and original PF-PDAF algorithms of four tracking scenarios with the setting of $Pd = 1, Cd = 0.5$
(averaged over the processed data)



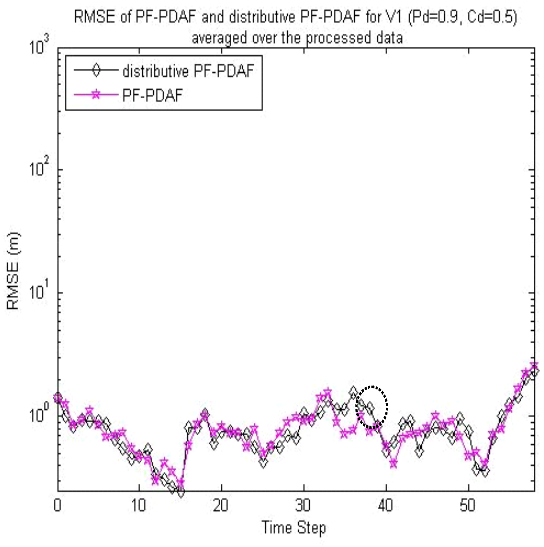
(a) V3



(b) V4

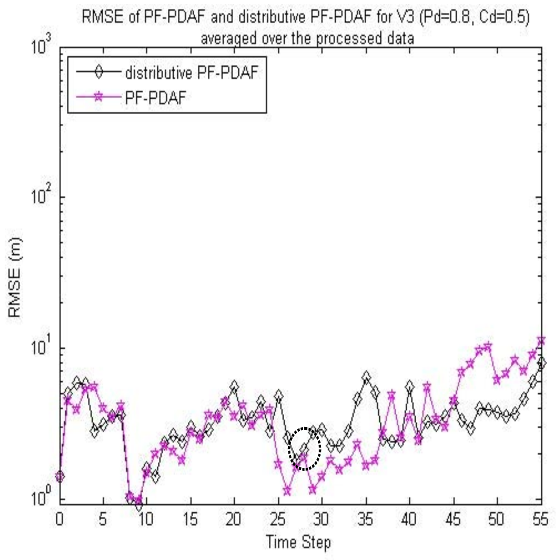


(c) V 10

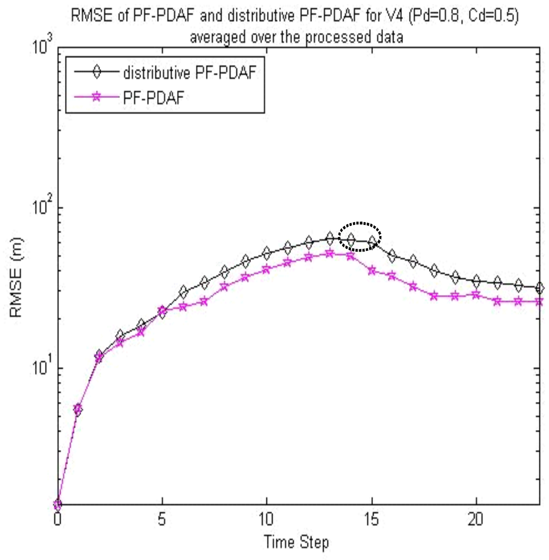


(d) V1

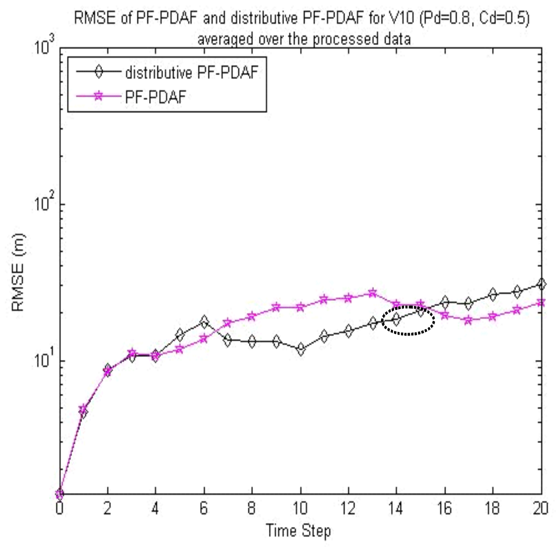
Figure 6.7 The performance of distributive PF-PDAF and original PF-PDAF algorithms of four tracking scenarios with the setting of $Pd = 0.9, Cd = 0.5$ (averaged over the processed data)



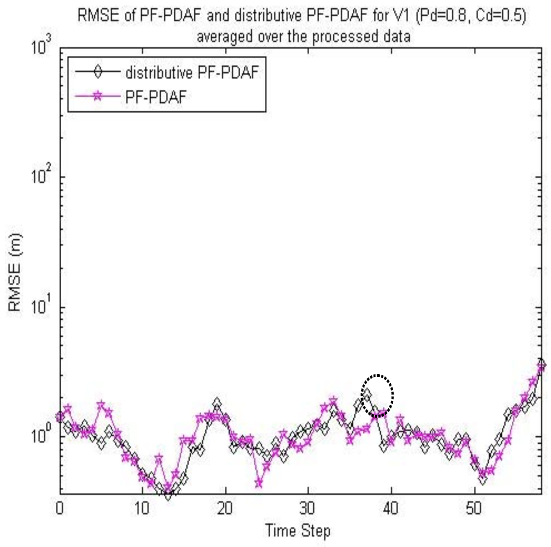
(a) V3



(b) V4

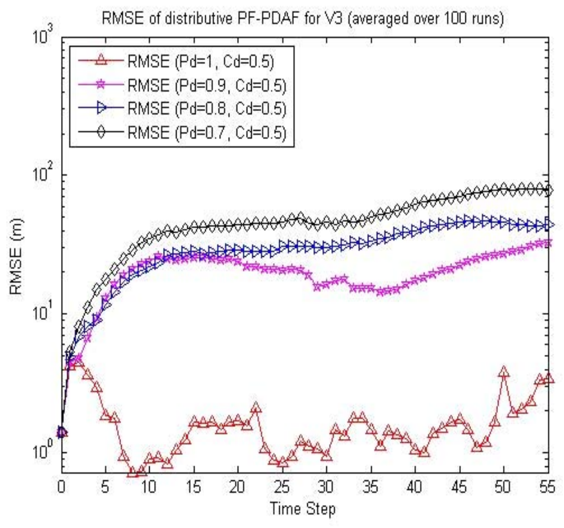


(c) V 10

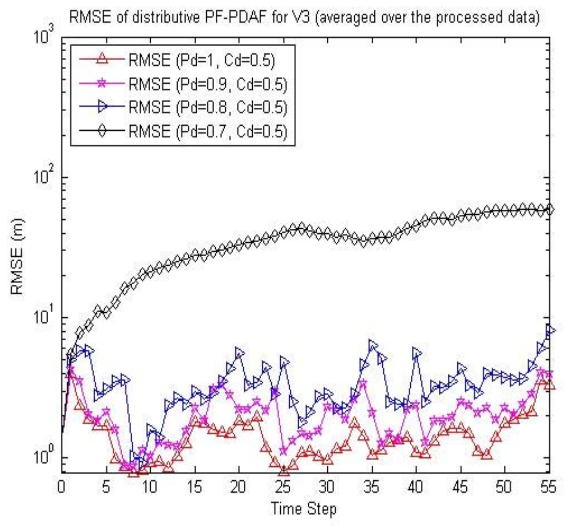


(d) V1

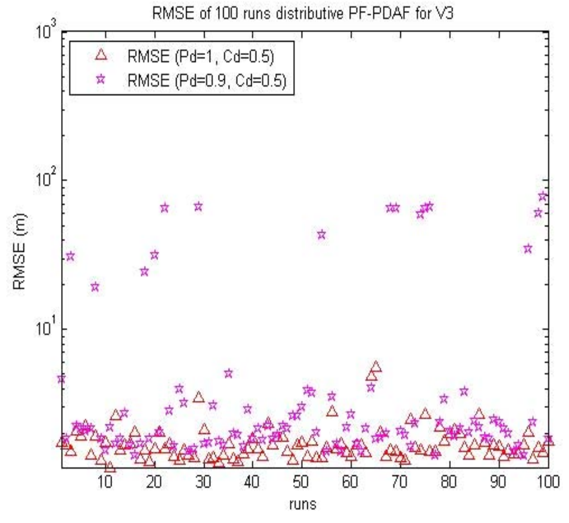
Figure 6.8 The performance of distributive PF-PDAF and original PF-PDAF algorithms of four tracking scenarios with the setting of $\mathrm{Pd} = 0.8, \mathrm{Cd} = 0.5$ (averaged over the processed data)



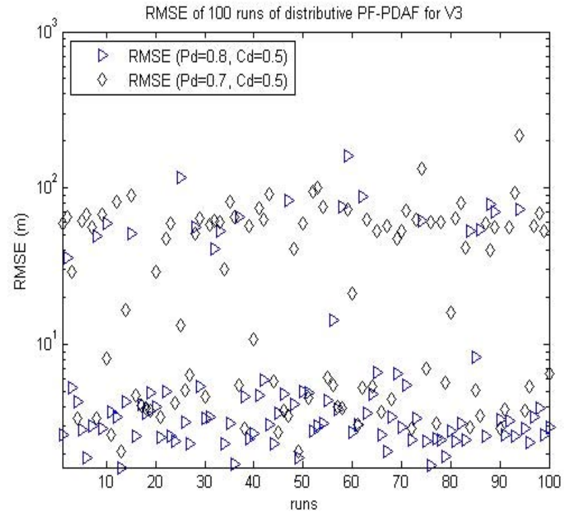
(a)



(b)



(c)

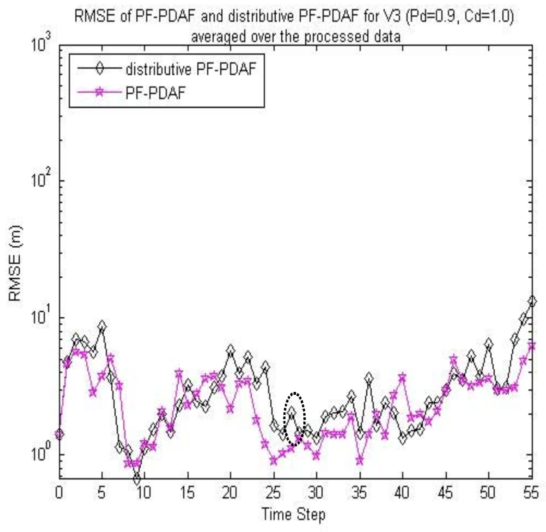


(d)

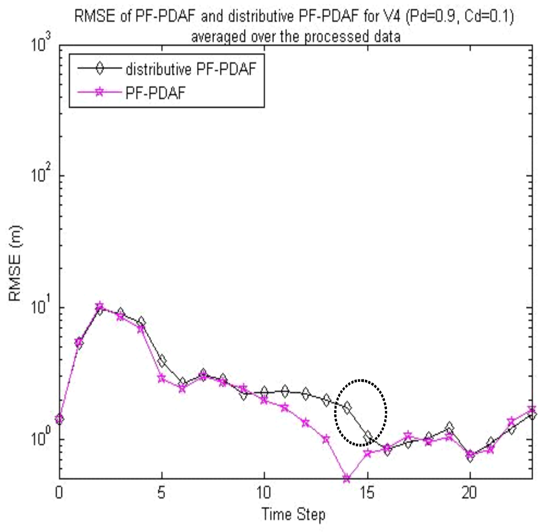
Figure 6.9 RMSE of distributive PF-PDAF algorithm with different detection rates for tracking scenario V3 (clutter rate fixed)
(a) RMSE_k value of each time step averaged over 100 runs
(b) RMSE_k value of each time step averaged over the processed data
(c) RMSE^n value of 100 runs ($\mathrm{Pd}=1, \mathrm{Cd}=0.5$ and $\mathrm{Pd}=0.9, \mathrm{Cd}=0.5$)
(d) RMSE^n value of 100 runs ($\mathrm{Pd}=0.8, \mathrm{Cd}=0.5$ and $\mathrm{Pd}=0.7, \mathrm{Cd}=0.5$)

Figures 6.10~6.12 compare the tracking performance of the distributive PF-PDAF algorithm and the original PF-PDAF algorithm with the varying clutter rates ($\mathrm{Cd}=0.1, 1.0, 1.5$) while the detection rate is fixed at $\mathrm{Pd}=0.9$ for each of the four tracking scenarios. The RMSE_k values in these figures are computed by the exclusion of the top 25 runs with the largest RMSE^n values and the bottom 25 runs with the lowest

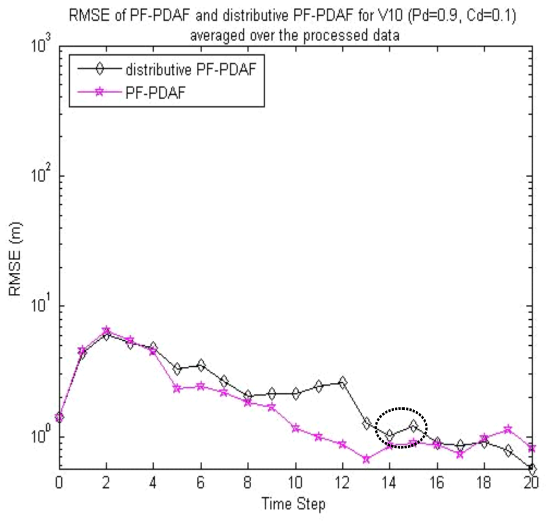
RMSE^n values (This is refereed to as the processed data in the figures). Figure 6.13 depicts the RMSE value of the distributive PF-PDAF algorithm in which the clutter rates are varying ($\text{Cd} = 0.1, 0.5, 1.0, 1.5$) while the detection rate is fixed at $\text{Pd} = 0.9$ for the tracking scenario V3. Figure 6.13 (a) shows the RMSE_k value that averaged over all 100 independent runs while Figure 6.13 (b) shows the RMSE_k value that computed by excluding the top 25 runs and the bottom 25 runs as before. Figures 6.13 (c) and 6.13 (d) show the RMSE^n value of all 100 runs for each setting.



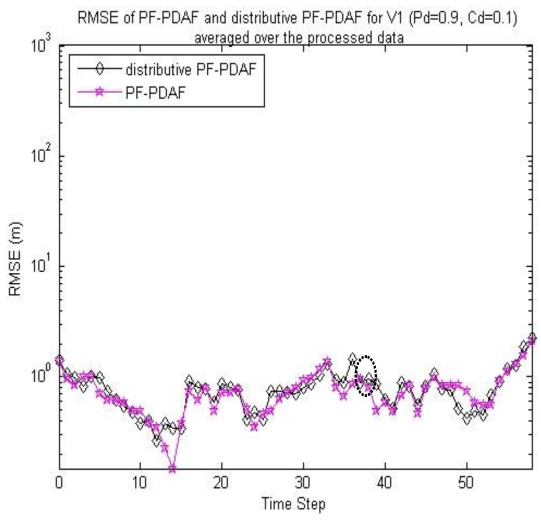
(a) V3



(b) V4

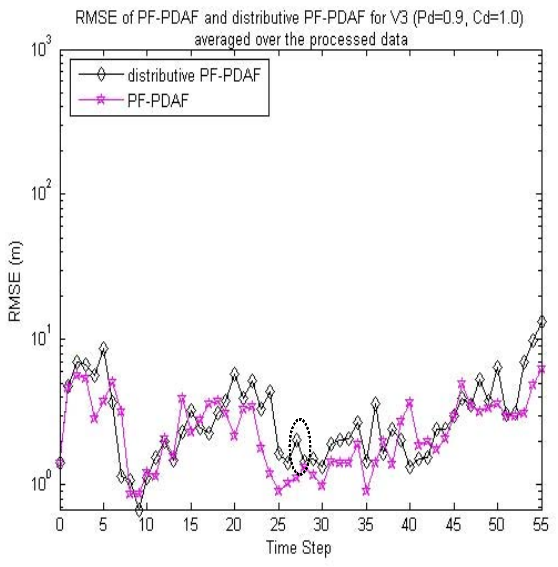


(c) V 10

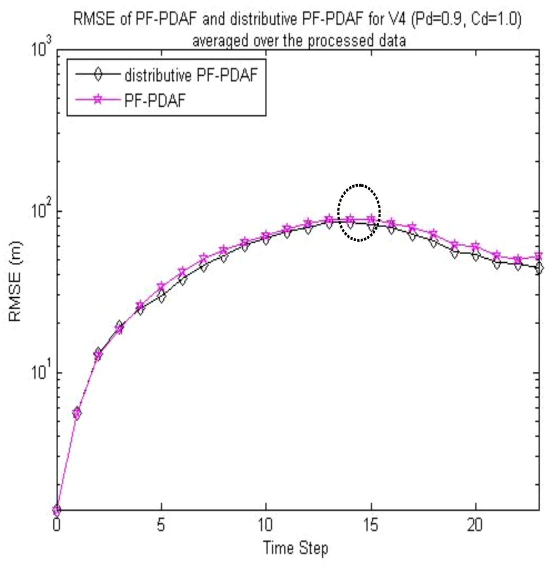


(d) V1

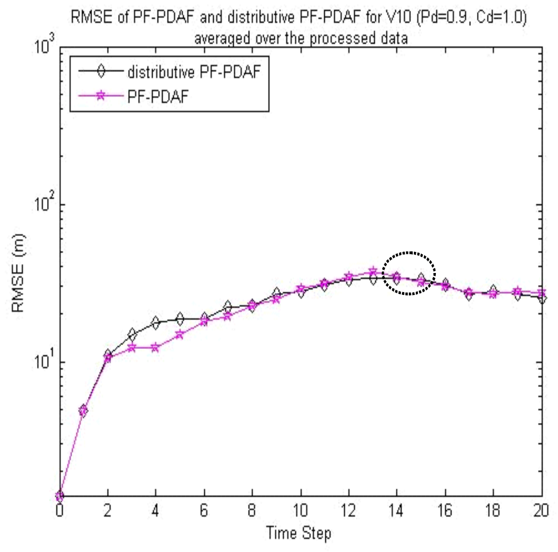
Figure 6.10 The performance of distributive PF-PDAF and original PF-PDAF algorithms of four tracking scenarios with the setting of $\text{Pd} = 0.9, \text{Cd} = 0.1$ (averaged over the processed data)



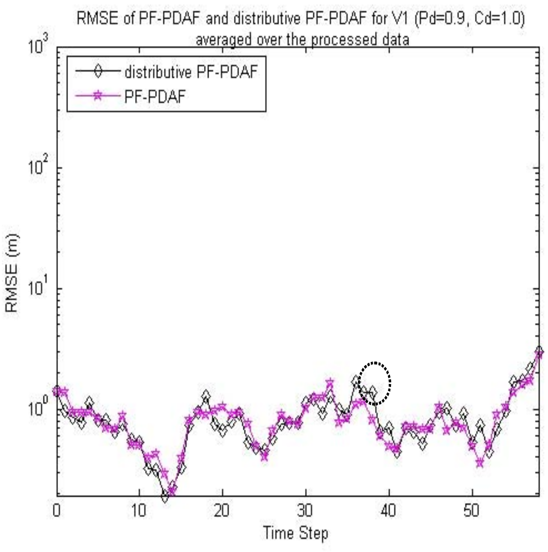
(a) V3



(b) V4

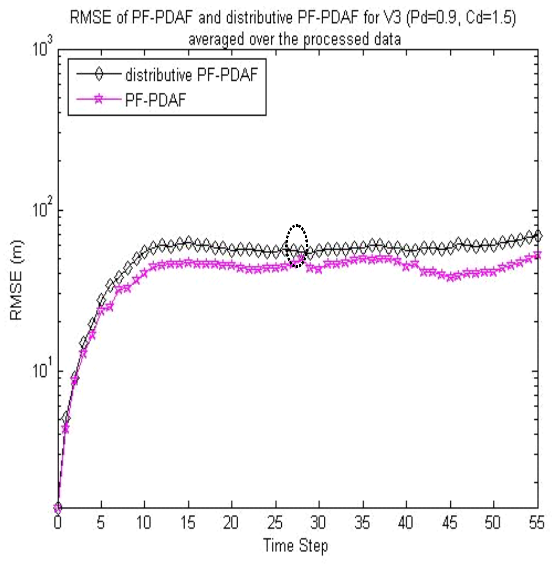


(c) V 10

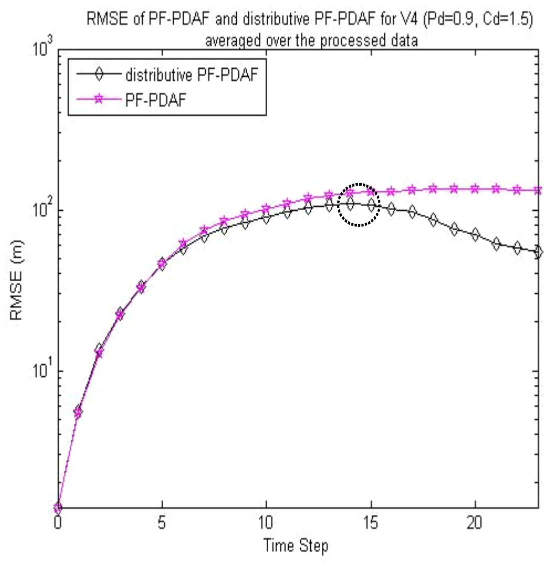


(d) V1

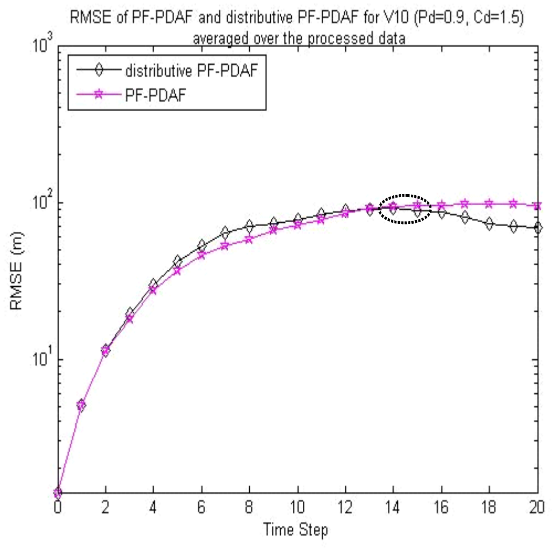
Figure 6.11 The performance of distributive PF-PDAF and original PF-PDAF algorithms of four tracking scenarios with the setting of $Pd = 0.9, Cd = 1.0$ (averaged over the processed data)



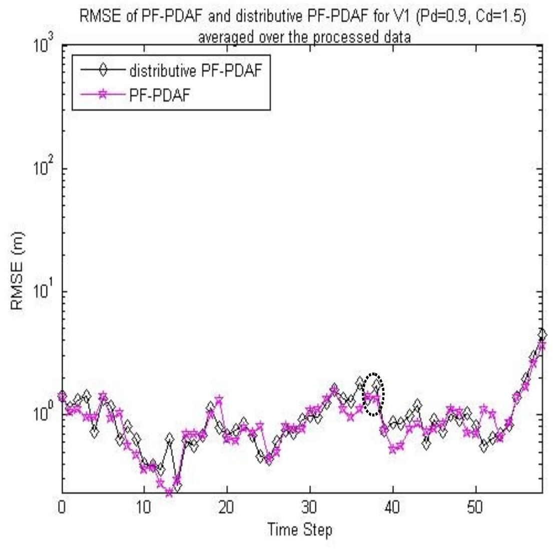
(a) V3



(b) V4



(c) V 10



(d) V1

Figure 6.12 The performance of distributive PF-PDAF and original PF-PDAF algorithms of four tracking scenarios with the setting of $Pd = 0.9, Cd = 1.5$ (averaged over the processed data)

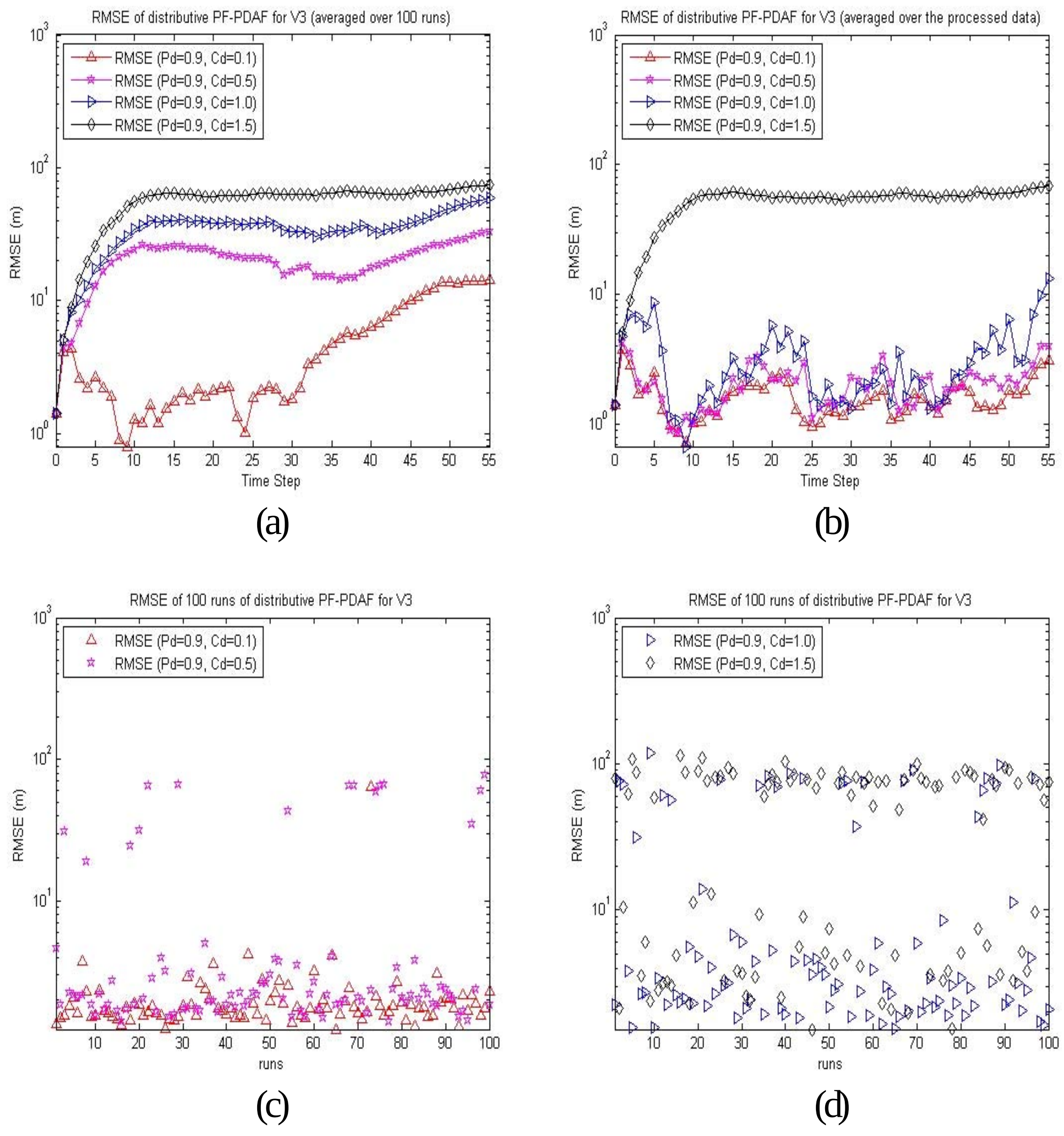


Figure 6.13 RMSE of distributive PF-PDAF algorithm with different clutter rates for tracking scenario V3 (detection rate fixed)

(a) RMSE_k value of each time step averaged over all 100 runs.

(b) RMSE_k value of each time step averaged over the processed data

(c) RMSE^n value of 100 runs ($\text{Pd} = 0.9, \text{Cd} = 0.1$ and $\text{Pd} = 0.9, \text{Cd} = 0.5$)

(d) RMSE^n value of 100 runs ($\text{Pd} = 0.9, \text{Cd} = 1.0$ and $\text{Pd} = 0.9, \text{Cd} = 1.5$)

From Figures 6.6~6.8 and Figures 6.10~6.12, it can be seen that during the “belief” handover period, the performance of the distributive PF-PDAF algorithm is degraded. However, after the handover period, the cluster leader in the second cluster collects measurements from sensing nodes to update its “belief” and eventually it attains almost the same tracking accuracy as the original PF-PDAF does. The overall performance of the distributive PF-PDAF under various detection rates and clutter rates is comparative with that of the original PF-PDAF algorithm developed in Chapter 5.

From Figures 6.9 and 6.13, it can be seen that, similar to the PF-PDAF, the increased measurement origin uncertainty deteriorates the tracking performance of the distributive PF-PDAF algorithm.

6.5.3 The Evaluation of Different Sensing Nodes Selection Schemes

In this section, firstly we evaluate four different sensing nodes selection schemes for the distributive PF-PDAF algorithm. These four sensing node selection schemes are described as follows:

Scheme 1. The cluster leader selects five sensing nodes that are closest to the predicted target position in the sensor cluster;

Scheme 2. The cluster leader selects three sensing nodes that are closest to the predicted target position in the sensor cluster;

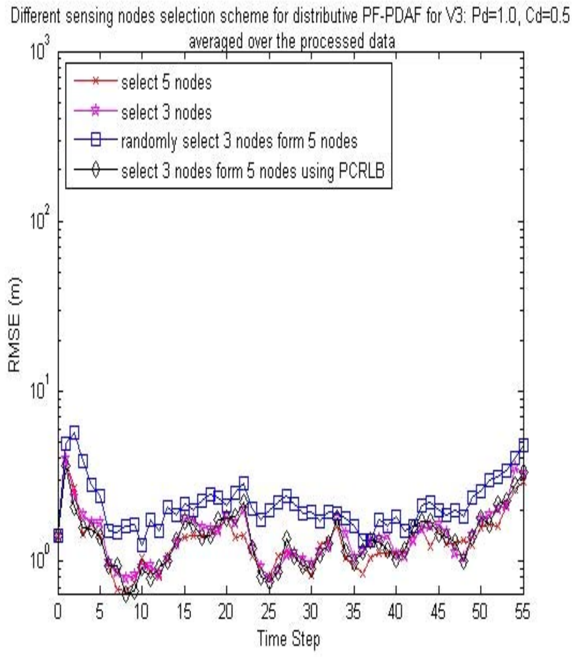
Scheme 3. This is a two-stage selection scheme; the cluster leader further randomly selects a combination of three sensing nodes from the five sensing nodes that have already been selected by *Scheme 1*;

Scheme 4. This is also a two-stage selection scheme; by calculating PCRLB, the cluster leader further selects a combination of three sensing nodes from the five sensing nodes that have already been selected by *Scheme 1*, this combination of three sensing nodes have the lowest PCRLB value amongst the all combinations of three sensing nodes.

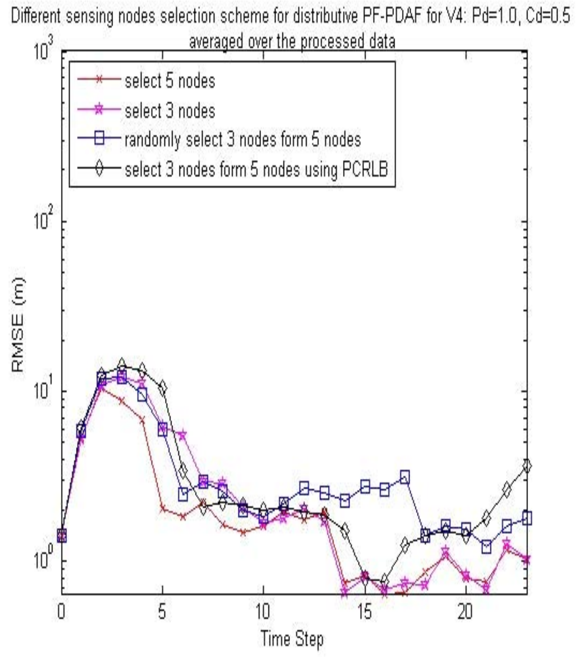
It can be seen that none of the above sensing nodes selection schemes incorporates the energy consumption measure. In contrast, the composite objective function based sensing nodes selection scheme developed in Section 6.3 takes into account both information utility and energy consumption. This scheme will be evaluated later in this section.

In the simulation, the settings of SNR and prior estimate of target state are the same as those in Section 6.5.2. However, the particles number used in the distributive PF-PDAF algorithm is 2000. The increase of particles number helps to mitigate the distraction on the evaluation of the sensing node selection schemes. For each tracking scenario as depicted in Figure 6.3, 100 independent Monte Carlo runs are conducted for each of the above four sensing nodes selection schemes.

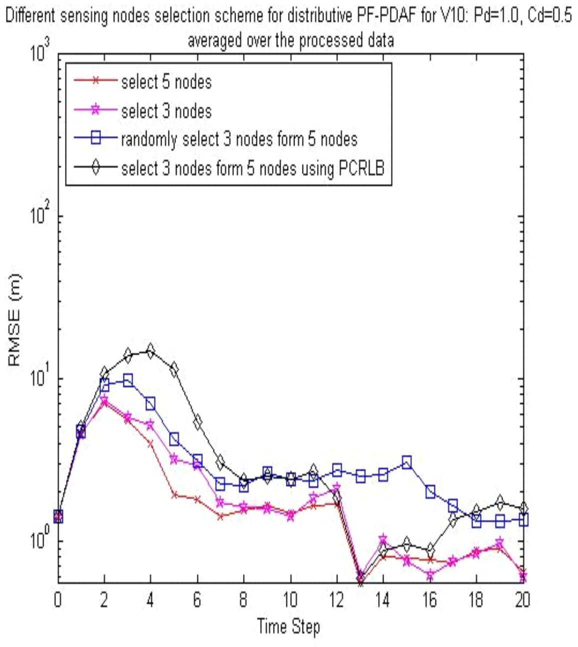
Figure 6.14 compares the tracking accuracy of the distributive PF-PDAF algorithm that adopts the above four different sensing nodes selection schemes. The RMSE_k values in the figure are calculated with data exclusion as before (i.e. the processed data).



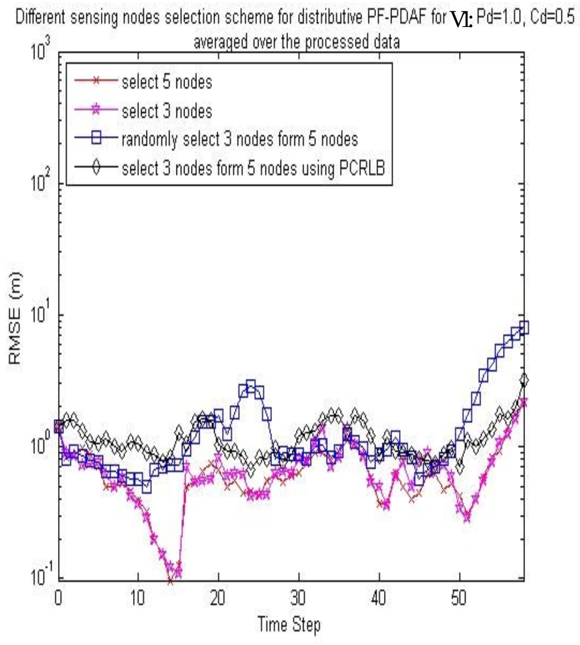
(a) V3



(b) V4



(c) V10



(d) V1

Figure 6.14 RMSE of distributive PF-PDAF algorithm adopting different sensing nodes selection scheme of four tracking scenarios (averaged over the processed data) (a) V3 (b) V4 (c) V10 (d) V1

From Figure 6.14, it can be seen that amongst the four selection schemes, Scheme 3 (randomly selecting a combination of three sensing nodes from the five closest sensing nodes) is outperformed by the other three schemes. The performance of Scheme 1, in which the cluster leader collects measurements from five closest sensing nodes, is slightly better than that of Schemes 2 and 4. However, Scheme 1 demands higher communication overhead since it needs to transmit five sets of measurements while Schemes 2 and 4 only need to transmit three sets of measurements. Recall from Chapter 5 that in single target tracking under measurement origin uncertainty, each set of measurements may contain several measurements which include the target originated measurement and the clutter originated measurements. Adopting Scheme 1 implies that a large number of measurements need to be transmitted in the sensor cluster and this will be at the cost of very high energy and bandwidth consumption in wireless sensor networks.

Comparing Scheme 2 which selects three closest sensing nodes and Scheme 4 which selects a combination of three sensing nodes that have the lowest PCRLB value from the five closest sensing nodes, it can be found that the performance of Scheme 2 is better than that of Scheme 4. This is due to the adoption of PCRLB as sensing nodes selection criteria in Scheme 4. The distributive PF-PDAF is suboptimal; however, the PCRLB is the bound on the performance of an optimal estimator. Hence there is a “gap” between the PCRLB and the MSE of distributive PF-PDAF algorithm (refer to Chapter 5 for the details) and this may cause some misleading in sensing nodes selection of Scheme 4. Moreover, the computational cost of Scheme 4 is slightly higher than that of Scheme 2. For one step updating of target state estimate, the distributive PF-PDAF algorithm employing Scheme 2 takes 0.1404 second while distributive PF-PDAF algorithm employing Scheme 4 takes 0.1717 second[7].

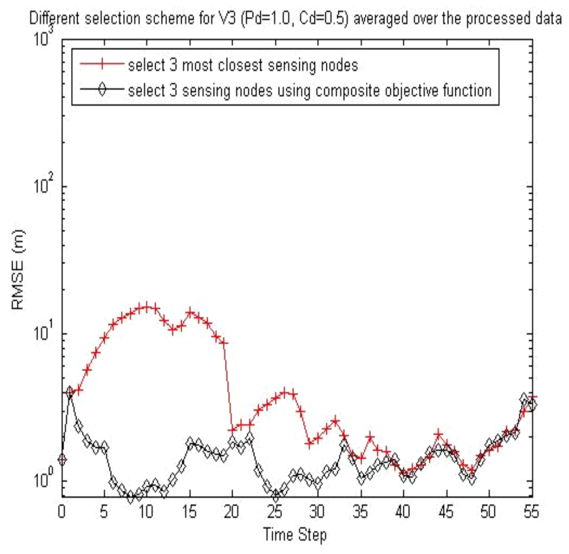
Nevertheless, when it needs to take into account both information utility and energy consumption measures in sensing nodes selection, the combination of three most closest sensing nodes in Scheme 2 may not be the optimal combination. This is because the three sensing nodes in Scheme 2 may not lie closely to the cluster leader (although they lie closely to the predicted target position) and they will consume considerable energy when transmitting their measurements to the cluster leader. Therefore, we employ the composite objective function proposed in Section 6.3 in sensing nodes selection scheme to trade-off

[7] This execution time refers to the run time of the non-optimized Matlab code running on a 2.80 GHz, Pentium 4 laptop.

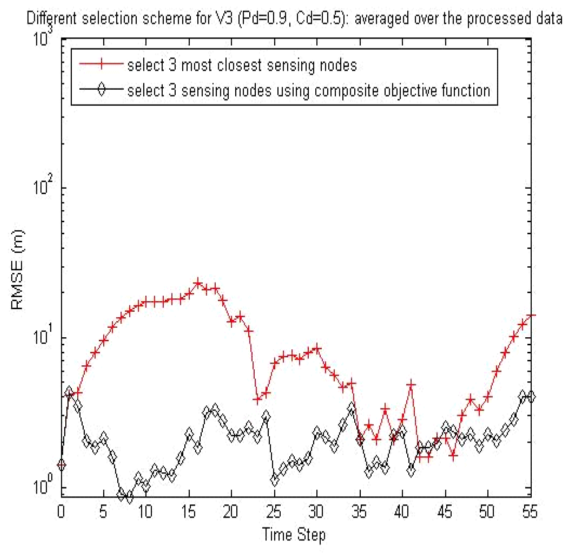
information utility and energy consumption of sensing nodes. As mentioned in Section 6.3, the information utility measure is computed based on the PCRLB and the energy consumption is computed based on the distance between the sensing node and the cluster leader (Equation 6.20).

Figure 6.15 compares the tracking performance of the distributive PF-PDAF algorithm that adopts the composite objective function based sensing node selection scheme and the distributive PF-PDAF algorithm that adopts the above sensing node selection Scheme 2 for tracking scenario V3. In Figure 6.15, the RMSE_k is computed by excluding the bottom 25 runs with the lowest RMSE^n values and the top 25 runs with the largest RMSE^n values. The setting of the simulation remains unchanged and the balance parameter α of the composite objective function (Equation 6.29) is set to 0.5 ($\alpha = 0.5$). From Figure 6.15, it can be seen that resulting distributive PF-PDAF algorithm can still track the target properly. Compared to the distributive PF-PDAF algorithm adopting Scheme 2, the magnitude of RMSE_k value of the distributive PF-PDAF adopting the composite objective function based selection scheme is higher. However, we can not arbitrarily conclude that the overall performance of distributive PF-PDAF algorithm is degraded, since the composite objective function based sensing node selection scheme represents a compromise between tracking accuracy and energy consumption in the wireless sensor network. At each time step, the sensing nodes are selected not only on the basis of their information utility, but also the energy consumption when they transmit their measurements to the cluster leader. The major benefit of adopting the composite objective function in sensing nodes selection is to attain the reasonable tracking accuracy while still maintaining the lower energy consumption. This helps prolong the lifetime of the whole wireless sensor network.

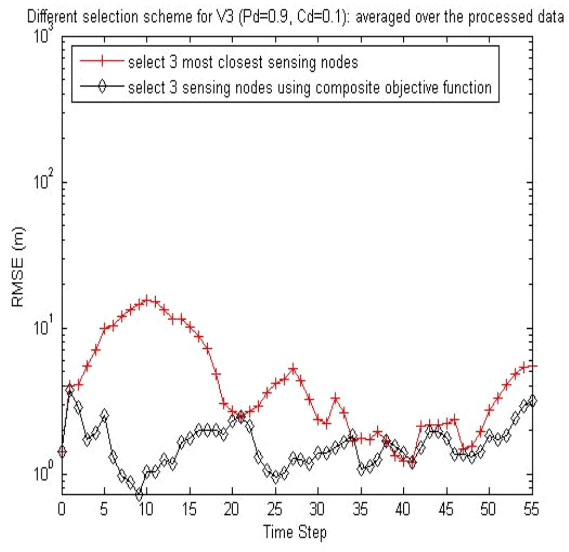
Figure 6.16 presents the simulation results of the distributive PF-PDAF algorithm that adopts the composite objective function with varying balance parameter α for each of the four tracking scenarios as depicted in Figure 6.3. The setting of the simulation remains unchanged and the balance parameter α is set as $PCRLB\left(\mathbf{a}_k\right) : \Theta\left(\mathbf{a}_k\right) = 1:9$ and $PCRLB\left(\mathbf{a}_k\right) : \Theta\left(\mathbf{a}_k\right) = 9:1$. It can be seen the more weight put on the information utility measure (represented by PCRLB), the more accuracy that the distributive PF-PDAF algorithm could attain.



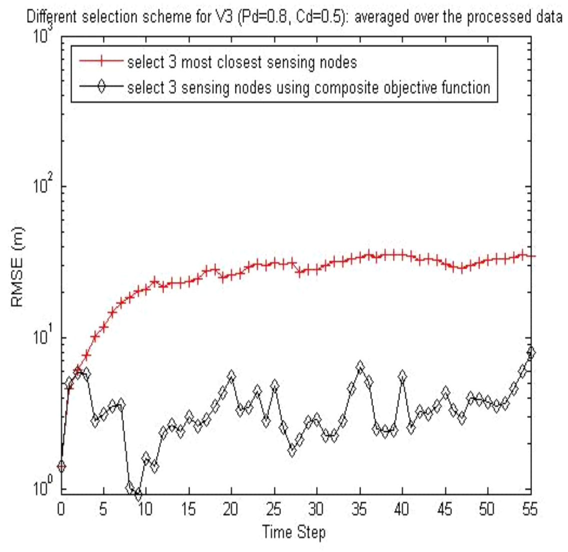
(a)



(b)



(c)

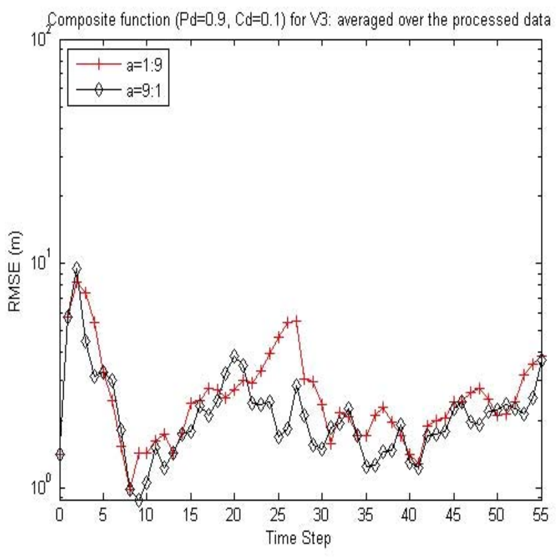


(d)

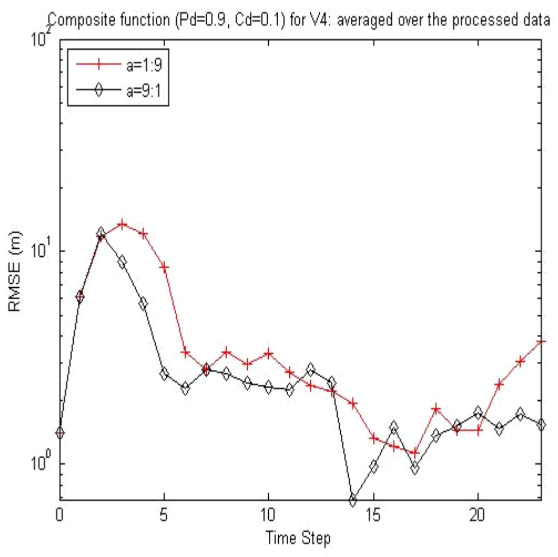
Figure 6.15 The performance of distributive PF-PDAF algorithm adopting Scheme 2 (selecting three closest sensing nodes) and adopting the composite objective function ($\alpha = 0.5$) for tracking scenario V3

(a) $\mathrm{Pd} = 1.0, \mathrm{Cd} = 0.5$ (b) $\mathrm{Pd} = 0.9, \mathrm{Cd} = 0.5$

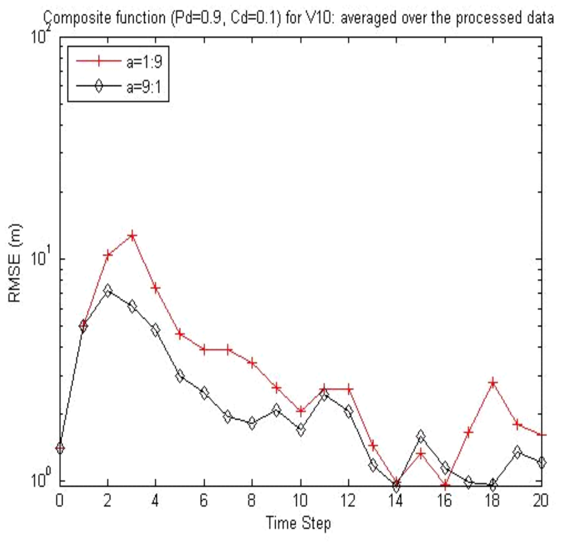
(c) $\mathrm{Pd} = 0.9, \mathrm{Cd} = 0.1$ (d) $\mathrm{Pd} = 0.8, \mathrm{Cd} = 0.5$



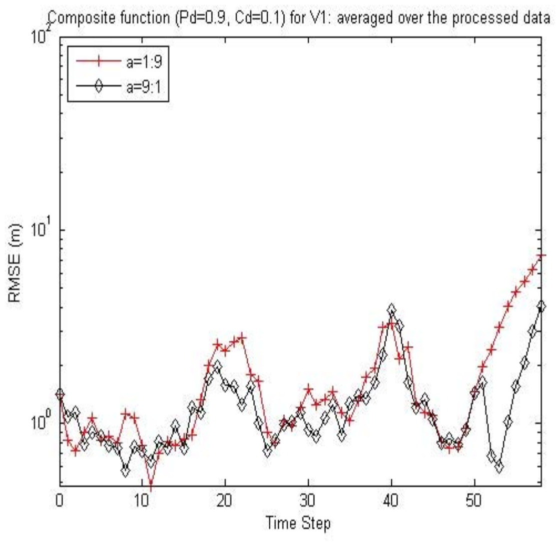
(a) V3



(b) V4



(c) V 10



(d) V1

Figure 6.16 The performance of distributive PF=PDAF algorithm adopting the composite objective function with varying balance parameter α ($Pd = 0.9, Cd = 0.1$, averaged over the processed data)

6.6 Summary

This chapter develops distributive tracking algorithms for tracking a single target in wireless sensor networks. Based on the collaborative information processing framework and the hierarchical sensor network architecture proposed in Chapter 3, the PF, EKPF and PF-PDAF tracking algorithms are extended for implementation in a distributive manner. The Gaussian mixture model (GMM) is adopted for the approximation of the probability density function of the target state. This helps to conserve communication bandwidth in the distributive PF, EKPF and PF-PDAF algorithms. Moreover, a composite cost function which balances the information utility measure and the energy consumption measure is developed for the sensing nodes selection in the distributive tracking algorithms. By adopting this composite cost function, we can conserves energy consumption while still maintaining the desirable tracking accuracy in the wireless sensor networks.