CHAPTER 1

INTRODUCTION

1.1 Motivation

In the last decades there has been a considerable growth in the semiconductor technology. With the advances in this field, the size of measuring devices are now smaller and smaller and their prices are quite low when compared to the past. Consequently these devices can now be deployed in various places where their small form factor is a major advantage. This gave rise to a big interest to the sensor networks and technology. Sensor networks have found application areas both in military and civil environments. In multi-sensor networks, multiple measurement devices, namely sensors, are employed in an area of interest to collect data and share this data with a data fusion center directly or by forwarding through their neighbors. Thus, we need to effectively control the sensors and interpret the information they send us. Effective utilization of sensors brings some other concepts into our consideration: Operational costs and sensor lifetime. Operational costs include cost of bandwith, power and computation while sensor lifetime is an issue directly related to the power consumption of the device. A powerful method for the effective utilization of sensors is to schedule them in a smart way. Sensor scheduling is performed to save the resources and improve the overall system performance.

1.2 Problem Statement

In this thesis we focus on a target tracking application based on observations received from multiple sensors. Our main objective is to estimate the position and velocity of our target. While doing this we want to utilize our sensors in an effective manner. The kinematics are defined with respect to a 2-dimensional Cartesian coordinate system. To estimate the position and velocity of our target we use Particle Filtering algorithm. For the sensor scheduling we propose a very simple algorithm by first grouping the sensors in clusters under the leadership of a master, and then comparing our position estimate with the position of each master one by one. The slaves associated with the closest master to the position estimate will be used to form the observation vector for the next time epoch.

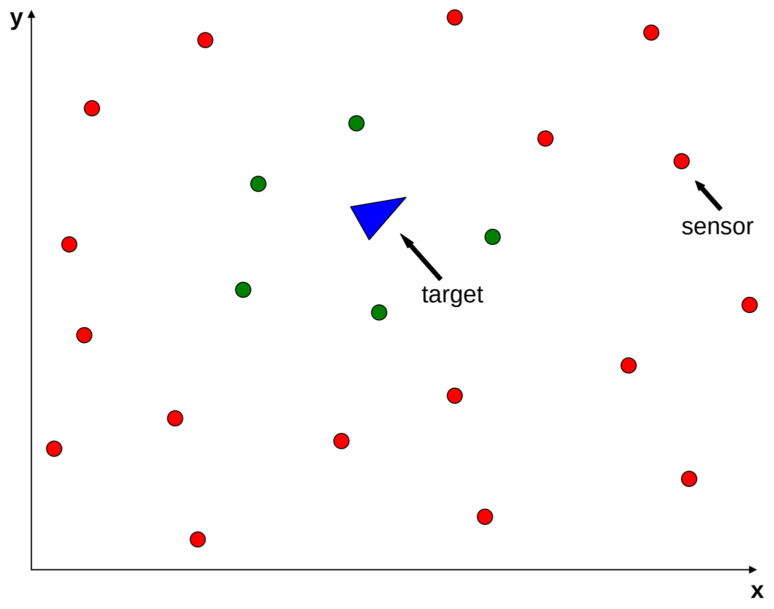

Figure 1: Basic diagram of sensor distribution. Green points represent active sensors while red colored ones indicate sleeping sensors.

1.3 Organisation

The thesis is organised as follows. In section 2 we will discuss Bayesian estimation techniques and the Particle filtering algorithm in a detailed manner. Our proposed algorithm for target tracking with sensor scheduling will be presented in Chapter 3. Finally in Chapter 4 we will provide our concluding remarks and future research plans.

CHAPTER 2

BAYESIAN ESTIMATION TECHNIQUES

2.1 State-Space Model

Many real world problems require estimation of the state of a system that is changing in time, using the sequence of noisy measurements from the system. The main goal of the Bayesian approach is to estimate the current state of the system based on the observations until that time. Generally, the observations are received sequentially in time. Thus, the current state estimate is also sequentially updated. In the state-space model, the parameters to be estimated form a state vector $x = \{x_{k;\, k \in \mathbb{N}}\}$ with dimension e_x. Observation vector $y = \{y_{k;\, k \in \mathbb{N}}\}$ with dimension e_y is the noisy measurement of this state vector. Both the current state of the system and the estimate of this state evolves dynamically in time. In the Bayesian framework, all the information about $X = \{x_0, x_1, ..., x_k\}$ can be obtained from the joint *a posteriori* distribution,

$p(x_0, x_1, ..., x_k \mid y_1, y_2, ..., y_k)$. So our final goal is to recursively estimate this joint *a posteriori* distribution or in some cases its marginals. We denote by $x_{0:k} \triangleq (x_0, x_1, ..., x_k)$ and $y_{1:k} \triangleq (y_1, y_2, ..., y_k)$ the state sequence, and the observations until time step k, respectively. Thus the joint *a posteriori* distribution can be written as $p(x_{0:k} \mid y_{1:k})$.

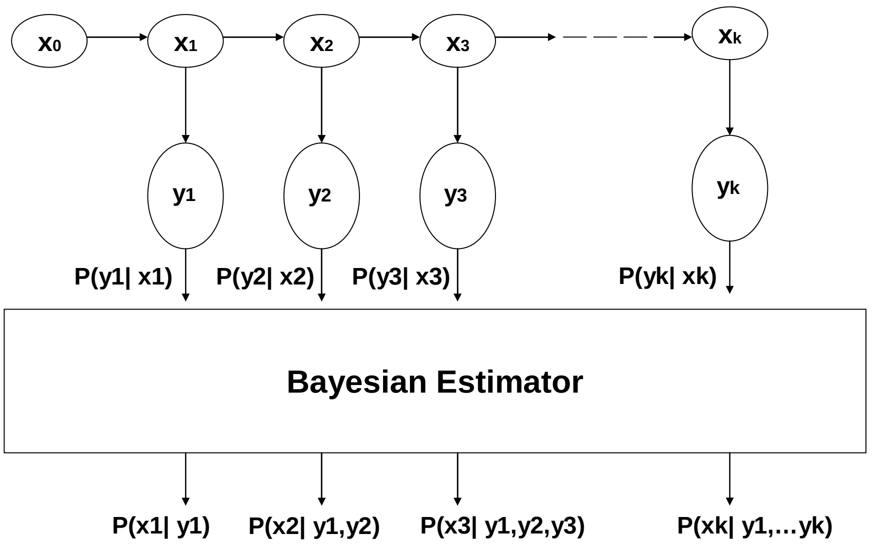

Figure 2: General bayesian approach to the state estimation problem

2.2 System Dynamics

In order to make inference about the current state of the system we need two models: System model and the observation model. These two models together describe the dynamics of our system. System model defines the transition from one state to the next in time and is described in [1] as

$$x_k = f_k(x_{k-1}, v_{k-1}) \qquad (1)$$

where $f_k : \Re^{e_x} \times \Re^{e_v} \rightarrow \Re^{e_x}$ is a non-linear function of the state x_{k-1}, v_{k-1} is an independent and identically distributed noise sequence, e_x and e_v are the dimensions of the state and noise vectors, respectively. Measurement model defines the observation vector as the noisy measurements of states and is described in [1] as

$$y_k = h_k(x_k, n_k) \qquad (2)$$

where $h_k : \Re^{e_x} \times \Re^{e_n} \rightarrow \Re^{e_y}$ is a non-linear function of the state x_k, n_k is an independent and identically distributed noise sequence, e_y and e_n are the dimensions of the observation and noise vectors, respectively.

We want to obtain the filtered estimates of x_k based on the set of all available measurements $y_{1:k} = (y_1, y_2, ..., y_k)$ upto time k. Thus we need to construct the pdf $p(x_k \mid y_{1:k})$. We are assuming that we have the knowledge of initial state x_0, therefore, $p(x_0 \mid y_0) = p(x_0)$. The distribution $p(x_k \mid y_{1:k})$ can be obtained in two stages : prediction and update.

If we assume that the posterior distribution $p(x_{k-1}|y_{1:k-1})$ at time step k-1 is available, prediction stage can be formulated by use of Chapman-Kolmogorov equation as follows

$$p(x_k|y_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|y_{1:k-1})dx_{k-1} \tag{3}$$

Stochastic model of the state transition, $p(x_k|x_{k-1})$, is defined by the system equation (1) .

At time step k, a new measurement y_k becomes available and this new information is used to update our prediction by use of Bayes' rule.

$$p(x_k|y_{1:k}) = \frac{p(y_k|x_k)p(x_k|y_{1:k-1})}{p(y_k|y_{1:k-1})} \tag{4}$$

where

$$p(y_k|y_{1:k-1}) = \int p(y_k|x_k)p(x_k|y_{1:k-1})dx_k \tag{5}$$

is the normalizing constant, and depends on the likelihood function defined by the measurement model (2).

By the equation (3) we are creating a prior density , namely the prediction, and by the equation (4) updating it with the likelihood function and normalizing with a constant. Recursive estimation of posterior density with this method is analytically possible only for some special cases of f_k, h_k, n_k and v_{k-1}. Two restrictions for the immediate analytic solution are the linearity of the models f_k and h_k, and the gaussainity of the process and observation noises, v_{k-1} and n_k.

2.3 Kalman Filter

Assuming that the restriction of linearity holds for both the system and observation models we can write them as

$$x_k = F_k x_{k-1} + v_{k-1}$$

$$v_k \sim \mathcal{N}(0, Q_k) \qquad (6)$$
$$n_k \sim \mathcal{N}(0, R_k)$$

$$y_k = H_k x_k + n_k \qquad (7)$$

Furthermore, if the process and observation noises are gaussian, Kalman Filter is the optimal recursive solution for the Bayesian estimation problem.

Considering the case when v_{k-1} and n_k are zero mean and statistically independent, Kalman Filter algorithm is described by the following relationships:

$$p(x_{k-1} | y_{1:k-1}) = \mathcal{N}(x_{k-1}; m_{k-1|k-1}, P_{k-1|k-1}) \qquad (8)$$

$$p(x_k | y_{1:k-1}) = \mathcal{N}(x_k; m_{k|k-1}, P_{k|k-1}) \qquad (9)$$

$$p(x_k | y_{1:k}) = \mathcal{N}(x_k; m_{k|k}, P_{k|k}) \qquad (10)$$

where

$$m_{k|k-1} = F_k m_{k-1|k-1} \qquad (11)$$

$$P_{k|k-1} = Q_{k-1} + F_k P_{k-1|k-1} F_k^T \tag{12}$$

$$m_{k|k} = m_{k|k-1} + K_k(y_k - H_k m_{k|k-1}) \tag{13}$$

$$P_{k|k} = P_{k|k-1} - K_k H_k P_{k|k-1} \tag{14}$$

where $\mathcal{N}(x; m, P)$ is Gaussian density with argument x, mean m and covariance P. We also assume $v_k \sim \mathcal{N}(0, Q_k)$ and $n_k \sim \mathcal{N}(0, R_k)$.

$$S_k = H_k P_{k|k-1} H_k^T + R_k \tag{15}$$

is the covariance of the innovation term and

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \tag{16}$$

is the Kalman gain.

Kalman Filter [2] is optimal in the sense that it minimizes the estimated error covariance under the following conditions:

- Evolution of the state is according to a known linear equation.
- Observation model is a linear function of the state with an additive zero mean WGN with known covariance.
- Initial state is assumed to be a random variable with known mean and covariance.
- Process and observation noise sequences are mutually uncorrelated.

2.4 Grid Based Methods

If the state is discrete and finite, grid-based methods can provide a good solution as an optimal way to update the filtering density $p(x_k \mid y_{1:k})$. Suppose the discrete state x consists of a finite number of distinct discrete states $\{1, 2, ..., N_x\}$. For the state x_{k-1}, let $\omega^i_{k-1|k-1}$ denote the conditional probability of each x^i_{k-1} given the measurements upto time k-1. That is, $\Pr(x_{k-1} = x^i_{k-1} \mid y_{1:k-1}) = \omega^i_{k-1|k-1}$. Then the posterior pdf at k-1 can be written as

$$p(x_{k-1} \mid y_{1:k-1}) = \sum_{i=1}^{Nx} \omega^i_{k-1|k-1} \delta(x_{k-1} - x^i_{k-1}) \tag{17}$$

Then, prediction and filtering equations are derived by substituting (17) into (3) and (4), respectively.

$$p(x_k \mid y_{1:k-1}) = \sum_{i=1}^{Nx} \omega^i_{k|k-1} \delta(x_k - x^i_k) \tag{18}$$

$$p(x_k \mid y_{1:k}) = \sum_{i=1}^{Nx} \omega^i_{k|k} \delta(x_k - x^i_k) \tag{19}$$

where

$$\omega^{i}_{k|k-1} \triangleq \sum_{j=1}^{Nx} \omega^{j}_{k-1|k-1} p(x^{i}_{k} | x^{j}_{k-1}) \tag{20}$$

$$\omega^{i}_{k|k} \triangleq \frac{\omega^{i}_{k|k-1} p(y_{k} | x^{i}_{k})}{\sum_{j=1}^{Nx} \omega^{j}_{k|k-1} p(y_{k} | x^{j}_{k})} \tag{21}$$

If the state space is continuous the approximate-grid based method can be similarly derived by discretizing the state space into Nx discrete cell states.

2.5 Extended Kalman Filter

Kalman Filter provides an exact solution for linear Gaussian prediction and filtering problem. But in practice it is limited by the non-linearity and the non-gaussianity of the physical world. If the functions $f_k(.)$ and $h_k(.)$ in equations (1) and (2) are non-linear then we need to use another method called Extended Kalman Filter (EKF). Method depends on the local linearization of the functions $f_k(.)$ and $h_k(.)$. Our previous assumption of gaussian and uncorrelated process and observation noise sequences still holds. EKF is based on the following approximations :

$$p(x_{k-1} \mid y_{1:k-1}) \approx \mathcal{N}(x_{k-1}; m_{k-1\mid k-1}, P_{k-1\mid k-1}) \tag{22}$$

$$p(x_k \mid y_{1:k-1}) \approx \mathcal{N}(x_k; m_{k\mid k-1}, P_{k\mid k-1}) \tag{23}$$

$$p(x_k \mid y_{1:k}) \approx \mathcal{N}(x_k; m_{k\mid k}, P_{k\mid k}) \tag{24}$$

where

$$m_{k\mid k-1} = f_k(m_{k-1\mid k-1}) \tag{25}$$

$$P_{k\mid k-1} = Q_{k-1} + \hat{F}_k P_{k-1\mid k-1} \hat{F}_k^T \tag{26}$$

$$m_{k\mid k} = m_{k\mid k-1} + K_k(y_k - h_k(m_{k\mid k-1})) \tag{27}$$

$$P_{k|k} = P_{k|k-1} - K_k \hat{H}_k P_{k|k-1} \qquad (28)$$

where $\hat{F}_k$ and $\hat{H}_k$ are local linearizations of $f_k(.)$ and $h_k(.)$.

$$\hat{F}_k = \left.\frac{df_k(x)}{dx}\right|_{x=mk-1|k-1} \qquad (29)$$

$$\hat{H}_k = \left.\frac{dh_k(x)}{dx}\right|_{x=mk|k-1} \qquad (30)$$

$$S_k = \hat{H}_k P_{k|k-1} \hat{H}_k^T + R_k \qquad (31)$$

$$K_k = P_{k|k-1} \hat{H}k^T S_k^{-1} \qquad (32)$$

2.6 Particle Filter

With the background knowledge of Stochastic filtering and Bayesian estimation we now focus our attention on the Particle Filtering for sequential state estimation. Particle Filtering is a kind of recursive Bayesian filter based on Monte Carlo simulation. Main idea is to represent the required posterior density function by a set of random samples with associated weights and to compute estimates based on these samples and weights. As the number of particles approaches infinity Monte Carlo characterization converges to the true posterior pdf. Higher the probability at a specific point, denser the particles concentrated around that point. The particles evolve along the time according to the state equation. By randomly sampling the state space we get a number of particles representing the evolving pdf. However, since the posterior density model is unknown we choose another distribution for the simplicity and call it as importance density.

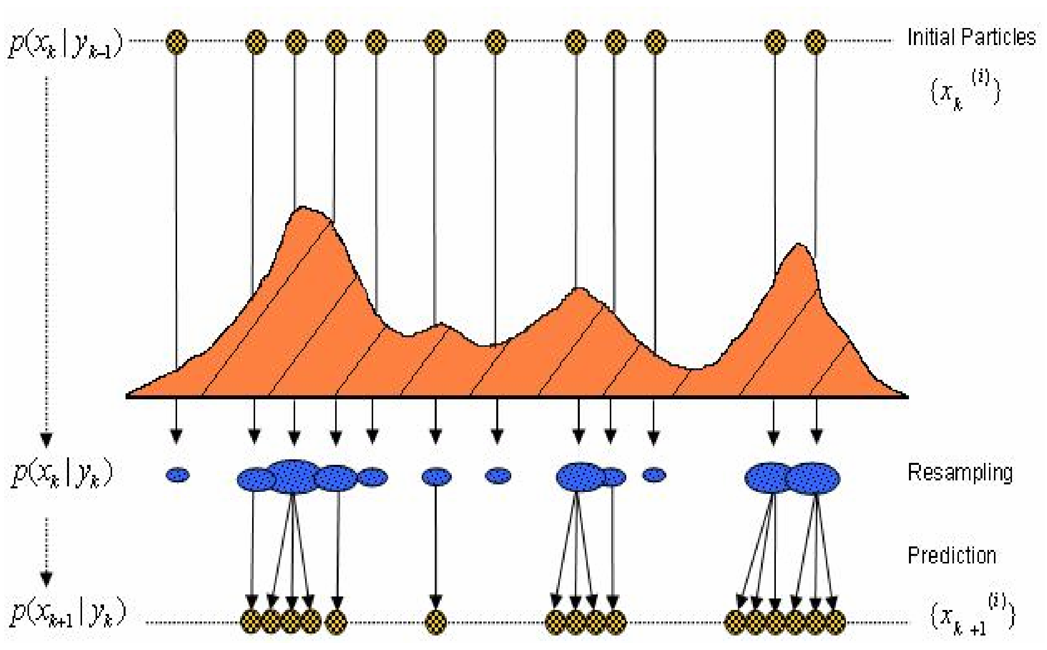

Figure 3 : Particle Filter illustration with importance sampling and resampling

If $\{x^i{}_{0:k}, \omega^i{}_k\}_{i=1}^{Ns}$ denotes a random measure characterizing the posterior pdf $p(x_{0:k} \mid y_{1:k})$ where $\{x^i{}_{0:k}, i = 1, ...N_s\}$ is the set of support points with associated weights $\{\omega^i{}_k, i = 1, ...N_s\}$ and $x_{0:k} = \{x_j, j = 0, ...k\}$ is the set of all states upto time k. The weights are normalized such that $\sum_i \omega^i{}_k = 1$. Then, the posterior density at k can be approximated as

$$p(x_{0:k} \mid y_{1:k}) \approx \sum_{i=1}^{Ns} \omega^i{}_k \delta(x_{0:k} - x^i{}_{0:k}) \qquad (33)$$

Validity of this approximation is guaranteed by the strong law of large numbers (SLLN), which states that the average of many independent random variables with common mean and finite variance converges to their common mean [8].

Thus, we have a discrete weighted approximation to the true posterior, $p(x_{0:k} \mid y_{1:k})$. Weights are chosen by use of importance density. If the samples $x^i{}_{0:k}$ were drawn from an importance density $q(x_{0:k} \mid y_{1:k})$ then the weights in (33) are defined to be

$$\omega^i{}_k \propto \frac{p(x^i{}_{0:k} \mid y_{1:k})}{q(x^i{}_{0:k} \mid y_{1:k})} \qquad (34)$$

Where $\propto$ tells us that there is a proportionality rather than the exact equality. The constant coefficient in order to turn (34) into a equation can be found by the fact that $\sum_i \omega^i{}_k = 1$.

Main idea behind this equation can be summarized as follows: If the weight of a sample at a specific suppport point is smaller than it should be, this means at that point, the importance density $q(.)$ that we proposed have a smaller value than the real density $p(.)$. Considering the fact that initially we have assigned equal weights to all samples this results in an increase in the weight.

As stated in [1], if we choose the importance density such that

$$q(x_{0:k} | y_{1:k}) = q(x_k | x_{0:k-1}, y_{1:k}) q(x_{0:k-1} | y_{1:k-1}) \qquad (35)$$

we can obtain samples $x^i{}_{0:k} \sim q(x_{0:k} | y_{1:k})$ by augmenting each of the existing samples $x^i{}_{0:k-1} \sim q(x_{0:k-1} | y_{1:k-1})$ with the new state

$x^i{}_k \sim q(x_k | x_{0:k-1}, y_{1:k})$. That is, once we have decided the samples for the initial state we can update those samples in time by the state evolution equation (we are going to define these steps in the SIS algorithm). If we express

$p(x_{0:k} | y_{1:k})$ in terms of $p(x_{0:k-1} | y_{1:k-1})$, $p(y_k | x_k)$ and $p(x_k | x_{k-1})$

$$p(x_{0:k} | y_{1:k}) = \frac{p(y_k | x_{0:k}, y_{1:k-1}) p(x_{0:k} | y_{1:k-1})}{p(y_k | y_{1:k-1})}$$

$$= \frac{p(y_k | x_{0:k}, y_{1:k-1}) p(x_k | x_{0:k-1}, y_{1:k-1})}{p(y_k | y_{1:k-1})} \times p(x_{0:k-1} | y_{1:k-1})$$

$$= \frac{p(y_k \mid x_k)\, p(x_k \mid x_{k-1})}{p(y_k \mid y_{1:k-1})} \times p(x_{0:k-1} \mid y_{1:k-1}) \qquad (36)$$

$$\propto p(y_k \mid x_k)\, p(x_k \mid x_{k-1})\, p(x_{0:k-1} \mid y_{1:k-1}) \qquad (37)$$

The weight update equation is obtained by substituting (35) and (37) into (34) as

$$\omega^i{}_k \propto \frac{p(y_k \mid x^i{}_k)\, p(x^i{}_k \mid x^i{}_{k-1})\, p(x^i{}_{0:k-1} \mid y_{1:k-1})}{q(x^i{}_k \mid x^i{}_{0:k-1}, y_{1:k})\, q(x^i{}_{0:k-1} \mid y_{1:k})}$$

$$= \omega^i{}_{k-1} \frac{p(y_k \mid x^i{}_k)\, p(x^i{}_k \mid x^i{}_{k-1})}{q(x^i{}_k \mid x^i{}_{0:k-1}, y_{1:k})} \qquad (38)$$

If moreover $q(x_k \mid x_{0:k-1}, y_{1:k}) = q(x_k \mid x_{k-1}, y_k)$, which is generaly true because of the first order markovian characteristics of the system model plus the observation model itself , importance density becomes only dependent on previous state x_{k-1} and the latest observation y_k. The weight update equation is then

$$\omega^i{}_k \propto \omega^i{}_{k-1} \frac{p(y_k \mid x^i{}_k)\, p(x^i{}_k \mid x^i{}_{k-1})}{q(x^i{}_k \mid x^i{}_{k-1}, y_k)} \qquad (39)$$

And the posterior filtered density is approximated as

$$p(x_k \mid y_{1:k}) \approx \sum_{i=1}^{Ns} \omega^i{}_k \delta(x_k - x^i{}_k) \qquad (40)$$

As $N_s \to \infty$, the approximation in (40) approaches the true posterior.
SIS algorithm consists of recursive propagation of the weights and support points as each measurement is received sequentially.

```
Algorithm 1: Sequential Importance Sampling
$[\{x^i{}_k, \omega^i{}_k\}_1^{Ns}] = SIS[\{x^i{}_{k-1}, \omega^i{}_{k-1}\}_1^{Ns}, y_k]$

•FOR i=1:Ns
    - Draw $x^i{}_k \sim q(x_k \mid x_{k-1}, y_k)$
    - Calculate $\omega^i{}_k$ according to (39)
•END FOR
```

Figure 4: Sequential Importance Sampling Algorithm

Altough we now have a viable statistical approach for approximating a recursive Bayesian filter, the SIS algorithm has a significant practical shortcoming. After a few iterations all but one particle will have negligible weight. This situation is called degeneracy phenomenon and causes to devote large computational effort to update the particles whose contribution to the approximation to $p(x_k \mid y_{1:k})$ is almost zero. It has been shown [3] that the variance of the importance weights can only increase over time and thus it is impossible to avoid degeneracy.

Effects of degeneracy can be reduced by the method of resampling. Resampling step is aimed to eliminate the samples with small importance weights and duplicate the samples with large weights. Resampling usually occurs between two sampling steps. In resampling step particles and associated importance weights $\{x^i{}_k, \omega^i{}_k\}$ are replaced by the new samples with equal importance weights $(\omega^i = \frac{1}{N_s})$ where N_s is the total number of samples that we have drawn. Resampling schedule can be deterministic or dynamic. In deterministic framework resampling is taken at every time step after running the sampling algorithm. In a dynamic schedule, a sequence of thresholds are set up and the variance of the importance weights are monitored; resampling is taken only if the variance exceeds the threshold.

Although the resampling step solves the problem of degeneracy, it introduces some other practical problems. It limits the opportunity to parallelize since all the particles must be combined. Also since the particles with high weight are statistically multiplexed this results in a loss of diversity. This second problem is known as *sample impoverishment* [4] and is severe especially in the case of small process noise.

```
Algorithm 2: Resampling

$[\{x_k^{j*}, \omega_k^j, i^j\}_{j=1}^{Ns}] = RESAMPLE[\{x_k^i, \omega_k^i\}_{i=1}^{Ns}]$

•Initialize the CDF: $c_1=0$
•FOR $i=2:N_s$
    - Construct CDF: $c_i=c_{i-1}+\omega_k^i$
•END FOR
•Start at the bottom of the CDF: $i=1$
•Draw a starting point: $u_1 \sim \mathcal{U}[0, N_s^{-1}]$
•FOR $j=1:N_s$
    - Move along the CDF: $u_j=u_1+N_s^{-1}(j-1)$
    - WHILE $u_j > c_i$
    * $i=i+1$
    - END WHILE
    - Assign sample : $x_k^{j*} = x_k^i$
    - Assign weight : $\omega_k^j = N_s^{-1}$
    - Assign parent : $i^j = i$
•END FOR
```

Figure 5: Resampling Algorithm

One of the critical point in the sequential importance sampling algorithm is the choice of proposal density. SLLN guarantees the convergence to the true posterior as $N_s \to \infty$. However, to obtain satisfactory performance for finite N_s, more care is required when choosing $q(x_{0:k} | y_{1:k})$.

Theorem :

To reduce the effects of degeneracy in the SIS algorithm a reasonable choice of $q(x_k | x_{k-1}, y_k)$ is the distribution that minimizes the conditional variance of the importance weights. This proposal distribution that minimizes $\mathrm{var}_q[\omega^i{}_k \mid x^i{}_{k-1}, y_k]$ is

$$q_{opt}(x_k \mid x^i{}_{k-1}, y_k) = p(x_k \mid x^i{}_{k-1}, y_k) \tag{41}$$

Proof [3]:

Beginning with (39), we have

$$\omega^i{}_k = \omega^i{}_{k-1} \frac{p(y_k \mid x^i{}_k)\, p(x^i{}_k \mid x^i{}_{k-1})}{q_{opt}(x^i{}_k \mid x^i{}_{k-1}, y_k)} \tag{42}$$

$$= \omega^i{}_{k-1} \frac{p(x^i{}_k, y_k \mid x^i{}_{k-1})}{p(x^i{}_k \mid x^i{}_{k-1}, y_k)} \tag{43}$$

$$= \omega^i{}_{k-1}\, p(y_k \mid x^i{}_{k-1}) \tag{44}$$

where we used the conditional independence of y_k given x_k to go from (42) to (43). Thus for the proposal distribution suggested in (41), the weight $\omega^i{}_k$ is conditionally independent of the actual draw of the current state $x^i{}_k$, or $\mathrm{var}_{q_{opt}}[\omega^i{}_k \mid x^i{}_{k-1}, y_k] = 0$, as stated. There are two problems with the optimal proposal distribution. First, it requires the ability to sample from $p(x_k \mid x^i{}_{k-1}, y_k)$, a distribution that may be nonstandard. Second, calculation of $\omega^i{}_k$ as specified in (44) requires the evaluation of the integral

$$p(y_k \mid x^i{}_{k-1}) = \int p(y_k \mid x_k)\, p(x_k \mid x^i{}_{k-1})\, dx_k \tag{45}$$

which may be analytically intractable. There are two cases where the use of q_{opt} is possible. The first is when x_k is from a discrete finite state space. In that case, the integral in (45) becomes a sum and sampling from $p(x_k \mid x^i_{k-1}, y_k)$ is possible. The second case occurs for dynamic models with additive Gaussian noise processes and linear measurement equations. With this model, equation (45) can be solved analytically because both terms in the integrand are Gaussian. Furthermore, because the measurement process is linear, $p(x_k \mid x^i_{k-1}, y_k)$ is also Gaussian and can be sampled.

There is one more possibility for the proposal distribution. Although it may be far from optimal, a popular and easy-to-implement choice is the so called prior distribution.

$$q(x_k \mid x^i_{k-1}, y_k) = p(x_k \mid x^i_{k-1}) \tag{46}$$

Sampling from the prior is often straightforward. For instance, in the case of additive noise model , $x_k = f_k(x_{k-1}) + v_{k-1}$, sampling from $p(x_k \mid x^i_{k-1})$ amounts to sampling from the noise distribution $p(v_{k-1})$. Furthermore, the weight update equation assumes even a simpler form,

$$\omega^i_k = \omega^i_{k-1} p(y_k \mid x^i_k) \tag{47}$$

Although the prior density as proposal distribution does not take the current measurement y_k into account and thus is suboptimal in that sense, we are going to use it in our simulations in this thesis.

Advanced and more efficient particle filtering techniques like the Auxiliary Particle Filtering [9], Marginalized Particle Filtering [10], Unscented Particle Filtering [11], [12] and the Rao-Blackwellized Particle Filtering [13] over the generic method described above exist but they are not going to be discussed in this thesis.

CHAPTER 3

PARTICLE FILTER BASED TARGET TRACKING

3.1 Description of Tracking Scenario

Because it provides the ability of a broad spatial coverage and multiplicity in sensing aspect, Sensor Networks are ideally suited for target tracking applications. Our problem of sensor scheduling for target tracking in sensor networks is illustrated in Figure 3. We don’t have a road constraint and therefore no prior knowledge of possible vehicle trajectories can be exploited.

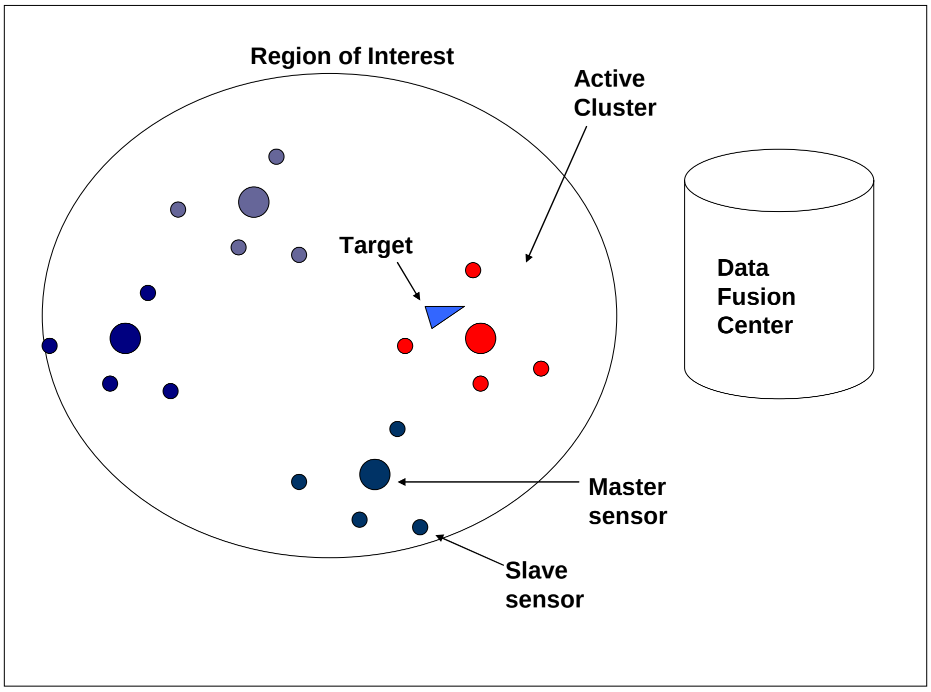

Figure 6: Tracking scneario. A vehicle moves through the sensor field. Active cluster is shown in red color.

We consider the task of tracking a moving vehicle through our two dimensional stationary sensor field under surveillance while conserving power by minimizing the number of active sensors. Before we run our tracking algorithm there is a set-up procedure which works as follows : First, we randomly distribute all the sensors into our region of interest. Then we again randomly decide which of the sensors will be masters. Master sensors are basically responsible for communicating with the data fusion center. Remaining sensors will be called slaves. Slave sensors report the position of target to their master periodically or if there is no target detected, they report this situation as well . After randomly distributing both type of sensors, we associate each slave with a master by runing our master-slave association algorithm. Basic criteria for this process is the Cartesian distance between the master and the slave sensors. Each slave is associated with the closest master. Another practical real world constraint that we take into account at this point is the service capacity of a master node. Maximum number of slaves that we can associate with each master is defined. If this limit is exceeded for any of the master during the set-up process than the remaining slaves are associated with another master. If we summarize our assumptions :

- We track a single target.
- We know the initial position of this target.
- Target state to be tracked consists of its two dimensional position and velocity.
- There are randomly distributed M stationary master sensors.
- There are randomly distributed S stationary slave sensors.
- Each slave sensor is associated with a master.
- Maximum number of slaves that can be associated with a master is C.
- At each time step there is only one active master communicating with the sink.
- All masters can communicate with each other.
- Slaves can communicate with their associated master sensor only.

```
Algorithm 3 : Master-Slave  Association

$S_{max}$ $\triangleq$ Number of Slave Sensors
$M_{max}$ $\triangleq$ Number of Master Sensors
$C_{max}$ $\triangleq$ Max Number of slaves that can be
            associated with a master

$\mathcal{A}$ $\triangleq$ Master sensors
$\mathcal{B}$ $\triangleq$ Slave sensors

Calculate Distance from all slaves to all masters:
•FOR i=1:$M_{max}$
    •FOR j=1:$S_{max}$
        - $\mathcal{D}(i : j) = ||\mathcal{A}_i - \mathcal{B}_j||$
    •END FOR
•END FOR

Assign each slave to the most close master:
•FOR j=1:$S_{max}$
    - $(d , i) = \min(\mathcal{D}(:, j))$
    •IF capacity of  ith master <= Cmax
        Assign slave j to master i
    •END IF
•END FOR
```

Figure 7: Master-Slave association algorithm

The driving force behind this scenario is the limiting conditions of the physical world. Above scenario for instance, can be exactly achieved by throwing the sensors away from a plane. Since the average communication time and the power consumption of the master sensors will be much more than the slaves, they can be equipped with longer life batteries and higher output transmit power RF communication ICs. That means we can differentiate master and slave sensors before the setup process and distribute both type of sensors in a completely random fashion. Number of master sensors and the number of slaves that each master can give service is a matter of optimization.

Another advantage of this scenario is the efficient use of bandwith when compared with the case each sensor sends the target position individually to the sink. By limiting the output transmission power of the slave sensors we can overcome the possible interference problem between the neighboring clusters as well. The reader may refer to [5] and [7] for a detailed description of different real world scenarios.

As mentioned before, our main objective is to accurately track the target while minimizing the number of active sensors. Only the active sensors provide observation about target position, otherwise they are configured to remain in sleep mode to reduce the power consumption. Thus, activation of sensors within a specified distance from the current target position estimate is quite important. Several different formulations of this problem are possible as target of interest moves through our randomly distributed sensors. Our approach at this point is simply to compare the current position estimate of the target with the position of each master sensor at every time step and to activate the associated slaves of the closest master for the next epoch. Here we are using the assumption that master sensors are always active and thus leadership can be immediately transferred from one master to another. Every master can activate its own slaves whenever needed. More sophisticated algorithms for this procedure can be implemented [7] such as the adaptive sensor activation regions. This problem will not be addressed in this thesis.

3.2 Models

We are now going to define system and obsevation models given in (1) and (2) in a detailed manner. For the 2-dimensional case state vector x_k contains four elements: positions in the x and y directions and velocities in the x and y directions.

$$x_k = \begin{bmatrix} x_{1k} \\ x_{2k} \\ \dot{x}_{1k} \\ \dot{x}_{2k} \end{bmatrix} \tag{48}$$

where x_{1k} and x_{2k} are the position in the x and y directions, respectively and $\dot{x}_{1k}$ and $\dot{x}_{2k}$ are the velocities in x and y directions, respectively. Kinematics for the target can be written as

$$x_{1k} = x_{1k-1} + \dot{x}_{1k-1}\Delta t + \frac{1}{2}\ddot{x}_{1k-1}\Delta t^2 + \frac{1}{3}\dddot{x}_{1k-1}\Delta t^3 \tag{49}$$

$$x_{2k} = x_{2k-1} + \dot{x}_{2k-1}\Delta t + \frac{1}{2}\ddot{x}_{2k-1}\Delta t^2 + \frac{1}{3}\dddot{x}_{2k-1}\Delta t^3 \tag{50}$$

Δt is the time difference between state transitions or simply the sampling frequency of us. The parameters $\ddot{x}_{1k}$ and $\ddot{x}_{2k}$ represents the acceleration in the x and y directions, respectively. Finally, $\dddot{x}_{1k}$ and $\dddot{x}_{2k}$ are to represent the variations in the acceleration in two directions again. We model the acceleration components using random noise. Using (49) and (50), the state equation can be written as

$$\begin{bmatrix} x_{1k} \\ x_{2k} \\ \dot{x}_{1k} \\ \dot{x}_{2k} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{1k-1} \\ x_{2k-1} \\ \dot{x}_{1k-1} \\ \dot{x}_{2k-1} \end{bmatrix} + \sqrt{q} \begin{bmatrix} \Delta t^3/3 & 0 & \Delta t^2/2 & 0 \\ 0 & \Delta t^3/3 & 0 & \Delta t^2/2 \\ \Delta t^2/2 & 0 & \Delta t & 0 \\ 0 & \Delta t^2/2 & 0 & \Delta t \end{bmatrix}^{1/2} \begin{bmatrix} v^1{}_{k-1} \\ v^2{}_{k-1} \\ v^3{}_{k-1} \\ v^4{}_{k-1} \end{bmatrix}$$

(51)

where q is used to control the intensity of the process noise. This equation can be expressed as

$$x_k = Fx_{k-1} + Q^{1/2} v_{k-1} \tag{52}$$

where Q is the state error covariance matrix. The matrix Q models the acceleration terms in the x and y directions. The vector v_k is a Gaussian random vector of zero mean, unit variance and independent components. The observation vector can simply be linearly related to the satate vector as

$$y_k = x_k + \sqrt{R} n_k \tag{53}$$

where R denotes the measurement error covariance matrix. The noise component n_k is an m x 1 vector whose elements are generated by a Gaussian random variable of zero mean and unit variance, where m is the number of slave sensors used to generate observations at time step k.

We want to emphasize that the reason behind our choice of this linear observation model instead of nonlinear range and bearing model is our application itself.

Based on this model we use the particle filtering method to estimate the state of the system. As mentioned earlier, we use the prior distribution as the importance density. At time step k, having the samples $x^i{}_k$ with associated weights $\omega^i{}_k$, the estimate of the state is given by

$$\hat{x}_k = \sum_{i=1}^{Ns} \omega^i{}_k x^i{}_k \tag{54}$$

If we assume each $\omega^i{}_k$ as the discrete probability masses at corresponding support points, then equation (54) can be thought of as an MMSE Estimator which is optimum only for Gaussian densities.

And the estimation error covariance matrix is given by

$$P_k = \sum_{i=1}^{Ns} \omega^i{}_k (x^i{}_k - \hat{x}_k)(x^i{}_k - \hat{x}_k)^T \tag{55}$$

We resample the particles at each time step instead of using dynamic resampling. After resampling, the weights are all initialized to $1/Ns$ to overcome the degeneracy.

3.3 Sensor Scheduling

In sensor scheduling tasks, we first need to define a cost function. This cost function should consider the combination of the following criteria depending on the physical situations and the specific problem at hand.

- Cost of bandwith
- Cost of sensor usage
- Cost of power
- Cost of accuracy of measurement

There may be several different costs for different applications. But the ones that we mentioned above are the most common ones. In this thesis when scheduling our sensors we only considered the cost of power assuming that power consumption is the most important constraint in sensor network applications.

Defining

$\mathcal{M} \triangleq$ Number of master sensors

$\mathcal{S} \triangleq$ Number of slave sensors

$\mathcal{A} \triangleq \{\mathcal{A}i;\ i = 1,....,\mathcal{M}\}$ set of master sensors in our region of interest

$\mathcal{B} \triangleq \{\mathcal{B}i;\ i = 1,....,\mathcal{S}\}$ set of slave sensors in our region of interest

We can define the cost function at time step k for a master sensor as the cartesian distance between the position estimate of target and the position of that master

$$C^i{}_k \triangleq \| \hat{x}_k - \mathcal{A}i \| \qquad (56)$$

and finally our scheduling decision is that we choose the master sensor for which the cost function is minimized.

$$\mathcal{A}_{opt} = \arg\ \min_{i} C^i k \qquad (57)$$

Then, the corresponding master sensor takes the leadership and activates its associated slaves immediately. For the next time step we use the observation vector obtained by this new sensor set.

Proposed tehnique is quite novel and applicable for only a single target. Sophisticated methods for tracking multiple maneuvering targets like Probability Hypothesis Density Filter [14] , [15] can be also examined.

3.4 Simulations and Results

In this section, we will discuss an example of target tracking using our proposed sensor scheduling algorithm. For the simulations the trajectory for a target is generated in a 2-dimensional cartesian coordinate system. Observations are made using sensor scheduling and the particle filtering algorithm is used for target tracking. Throughout the simulations Matlab 7.0 was used.

Initially 64 master sensors and 256 slave sensors are distributed randomly in the area x = (-8000,8000) and y = (-8000,8000). Then, our simple clustering algorithm is applied. Maximum number of slave that a master can give service is assumed to be 7.
Sampling period, Δt, was choosen to be 2 seconds. The process noise intensity factor q in (51) was taken as 0.01 and the initial position of target was taken to be (x,y) = (10,0). The measurement error covariance matrix in (53) was assumed to be 4 by 4 identity matrix noting that the accuracy of position measurements actually defines it. The state covariance matrix in (51) is defined as

$$Q = \begin{bmatrix} 2.67 & 0 & 2 & 0 \\ 0 & 2.67 & 0 & 2 \\ 2 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \tag{58}$$

The last one directly comes from $\Delta t = 2$. For the particle filter algorithm we used a total of 200 particles and 1000 time steps.

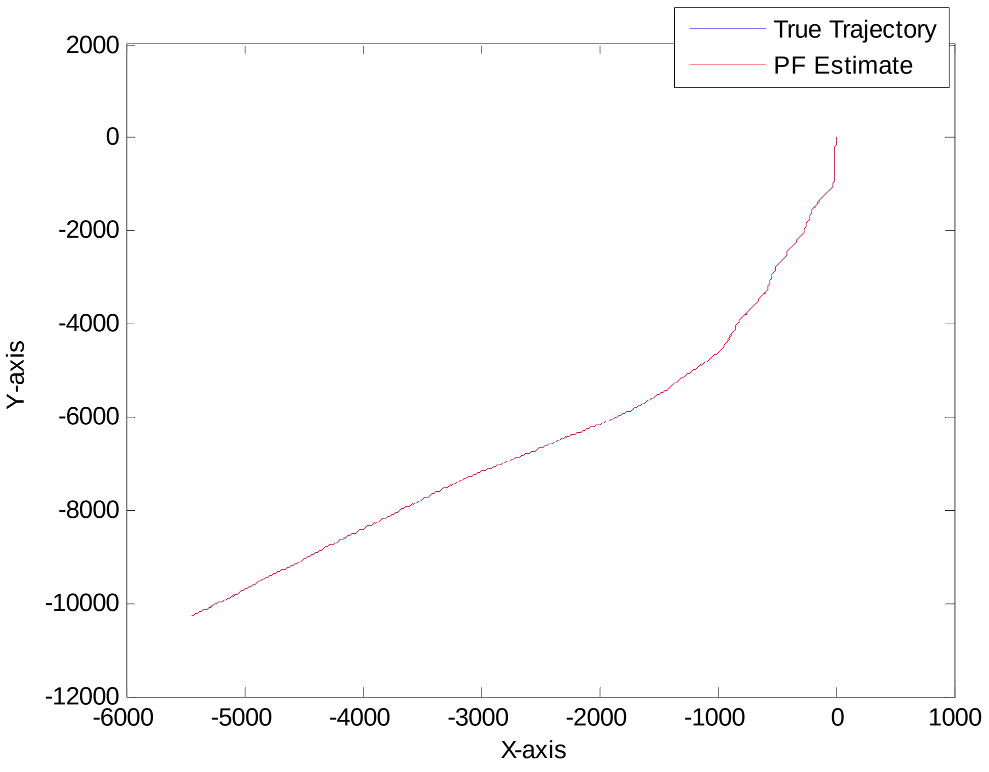

Figure 8 : True trajectory and the estimated trajectory

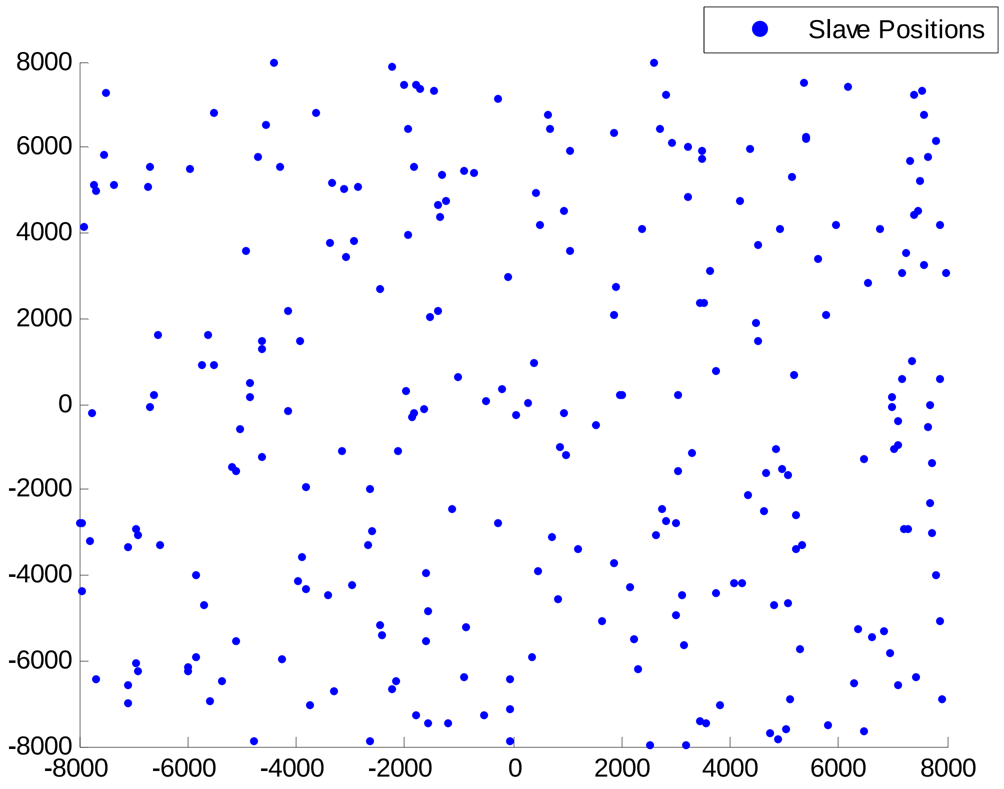

Figure 9 : Slave Positions

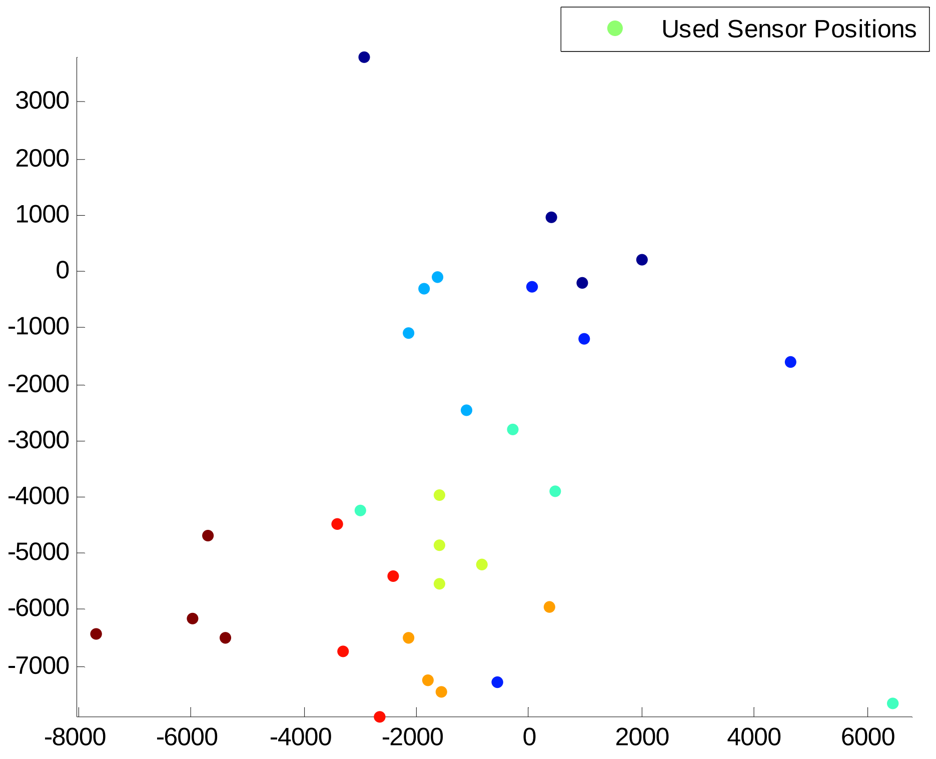

Figure 10: Activated sensors throughout the track. Each color indicates a cluster.

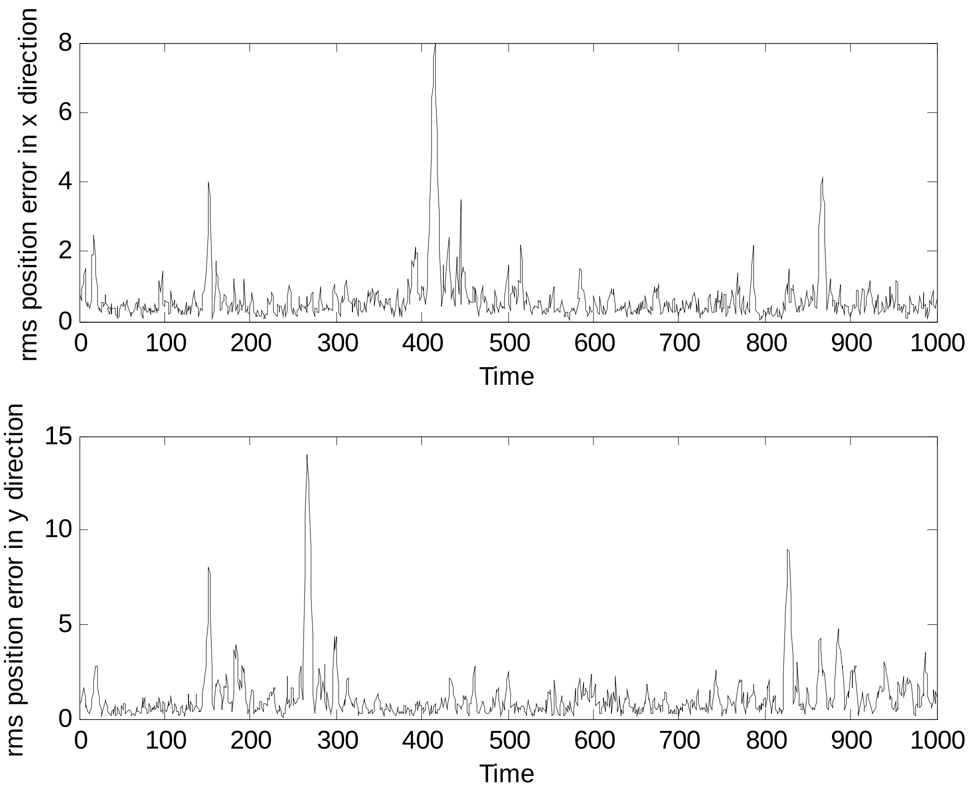

Figure 11 : rms position errors

CHAPTER 4

CONCLUSION AND REMARKS

In this thesis we have presented a recursive Bayesian formulation for target tracking and proposed a simple sensor scheduling technique in order to reduce power consumption of the system. We discussed in detail the Bayesian approach to target tracking. In particular, we have formulated the target tracking problem using state-space equations. Tracking was considered as a sequential estimation problem and particle filtering algorithm was implemented. In order to schedule the sensors in our region of interest we have compared the position estimate of our target and the master sensors. Approach can be named “Closest Master Activate” since the closest master takes the leadership and its associated slaves were used to obtain observations for the next time step. We observed that our scheduling results are quite satisfactory. Over all power consumption of the system is extremely low when compared to the case where no scheduling is done.

References

[1] *M.S. Arulampalam, S. Maskell, N.Gordon, T. Clapp "A tutorial on Particle Filters for Online Nonlinear/non-Gaussian Bayesian Tracking" IEEE Trans on Signal Processing, vol 50, pp 174-188, February 2002.*

[2] *R.E. Kalman "A new approach to linear filtering and prediction problem", Trans. ASME, Ser. D, J Basic Eng., vol.82, pp. 34-45, 1960*

[3] *A. Doucet, S.Godsill, C.Andrieu, "On Sequential Monte Carlo Sampling Methods for Bayesian Filtering" , Statist. Comput., 10, 197-208, 2000*

[4] *J. Carpenter, P.Clifford, P. Fearnhead, "An Improved Particle Filter for Non-linear Problems", IEEE proc. Radar, Sonar and Navigation , vol. 146, pp. 2-7, February 1999*

[5] *N. Xiong, P. Svensson, "Multi-sensor management for information fusion: issues and approaches," Information fusion 3, (2002), 163-186*

[6] *X. R. Li, V.P.Jilkov, "Survey of maneuvering Target Tracking", IEEE Trans. On Aerospace and electronics systems," vol. 39, No.4, October 2003*

[7] *S. Ghiasi, A.Srivastava, X. Yang, M. Sarrafzadeh "Optimal Energy Aware Clustering in Sensor Networks," Sensors 2002, 2, 258-269, July 2002*

[8] *D. Crisan, A.Doucet,"A Survey of Convergence Results on Particle Filtering Methods for Practitioners," IEEE Trans. On Signal Processing, Vol 50, No 3, March 2002*

[9] *M.K. Pitt, N.Shephard, “Filtering via Simulation: Auxilary Particle Filters,” Journal of the American Statistical Association, Vol 94, No 446, 1999*

[10] *T. Schön, R. Karlsson, F. Gustafsson, “The Marginalized Particle Filter in Practice,” Proceedings 2006 IEEE Aerospace conference, 2006*

[11] *R. Merwe, A. Doucet, N. Freitas, and E. Wan, "The unscented particle filter," Technical Report CUED/F-INFENG/TR 380, Cambridge University Engineering Department, 2000.*

[12] *Y. Rui and Y. Chen, "Better Proposal Distributions: Object Tracking Using Unscented Particle Filter," Proc. IEEE Conf. Computer Vision and Pattern Recognition, vol. II, pp. 786-793, 2001.*

[13] *E. Arnaud, E. Memin, “An Efficient Rao Blackwellized Particle Filter for Object Tracking, ” IEEE International Conference on Image Processing 2005. ICIP 2005., vol 2, pp 426-9, September 2005.*

[14] *B. Vo, S. Singh, A. Doucet “Sequential Monte Carlo Implementation of the PHD Filter for Multi Target Tracking,”, Proceedings of the Sixth International Conference of, Information Fusion 2003, vol 2, Issue 2003, pp 792-799, 2003.*

[15] *R. P. S. Mahler, “Multitarget Bayes Filtering via First-Order Multitarget Moments”, IEEE Trans. On Aerospace and electronics systems,” vol 39,
No 4, pp 1152-1178, May 2003*