Journal of Computer and Knowledge Engineering, Vol. 1, No. 2, 2018.
DOI: 10.22067/cke.v1i2.61798

Particle Filter based Target Tracking in Wireless Sensor Networks using Support Vector Machine

Ahmad Namazi Nik   Abbas Ali Rezaee*

Abstract: Target tracking is estimating the state of moving targets using noisy measurements obtained at a single observation point or node. Particle filters or sequential Monte Carlo methods use a set of weighted state samples, called particles, to approximate the posterior probability distribution in a Bayesian setup. During the past few years, Particle Filters have become very popular because of their ability to process observations represented by nonlinear state-space models where the noise of the model can be non-Gaussian. There are many Particle Filter methods, and almost all of them are based on three operations: particle propagation, weight computation, and resampling. One of the main limitations of the previously proposed schemes is that their implementation in a wireless sensor network demands prohibitive communication capability since they assume that all the sensor observations are available to every processing node in the weight update step. In this paper, we use a machine learning technique called support vector machine to overcome this drawback and improve the energy consumption of sensors. Support Vector Machine (SVM) is a classifier which attempts to find a hyperplane that divides two classes with the largest margin. Given labeled training data, SVM outputs an optimal hyperplane which categorizes new examples. The training examples that are closest to the hyperplane are called support vectors. Using our approach, we could compress sensor observations and only support vectors will be communicated between neighbor sensors which lead to cost reduction in communication. We use LIBSVM library in our work and use MATLAB software to plot the results and compare the proposed protocol with CPF and DPF algorithms. Simulation results show significant reduction in the amount of data transmission over the network.

Keywords: Distributed Particle Filter; Support Vector Machines; Target Tracking; Wireless Sensor Networks**.**

1. Introduction

Target tracking is one of the most important applications of wireless sensor networks. Examples include security and surveillance [1], environmental monitoring [2] and tracking tasks [3]. Target tracking is the estimation of the current state and prediction of future states of a target based on measurements received from a sensor that is observing it. The limited on-board resources of the sensor node and the limited wireless bandwidth are the major constraints of performing target tracking in wireless sensor networks. In order to save resources, target tracking should be implemented in a distributed way. Distributed computation has found very successful applications in sensor networks, particularly when a powerful central unit is not available.

Before particle filtering methods became popular, the Kalman filter was the standard method for solving state space models [4]. The Kalman filter can be applied to optimally solve a linear Gaussian state space model. When linearity or Gaussian conditions do not hold, its variants, i.e. the extended Kalman filter and the unscented Kalman filter, can be used. However, for highly nonlinear and non-Gaussian problems they fail to provide a reasonable estimate.

Particle filtering techniques offer an alternative method. They work online to approximate the marginal distribution of the latent process as observations become available. Importance sampling is used at each point in time in order to approximate the distribution with a set of discrete values, known as particles, each with a corresponding weight. There are several papers and books which have presented detailed reviews of particle filters and their applications [5-12].

In this work we tackle the problem of implementing the DPF algorithm and make use of support vector machine – a well-known machine learning classification method – to compress measurements collected by processing nodes and thus reducing communication costs.

The rest of the paper is organized as follows. In Section 2, a brief review of prior related works on target tracking is presented. In Section 3 we introduce the problem of target tracking in the context of Bayesian filtering and describe the solution to the nonlinear filtering problem with a centralized PF. In Section 4 we provide a formal description of the DPF algorithm. Section 5 introduces support vector machines. In Section 6 we provide details of the proposed method. Simulation and experimental results are presented and discussed in Section 7 and, finally, Section 8 is devoted to conclusions.

2. Related Works

Target tracking has many real life applications such as battlefield surveillance, detection of illegal borders crossing, gas leakage, fire spread, and wildlife monitoring.

Various taxonomies of target tracking algorithms have been proposed in the literature and there is no standardized or predefined classification. Some works have studied tracking algorithms according to the security aspect [13] while others have considered energy efficiency [14], fault tolerance, mobility, accuracy, and so on [15].

A comparative study of target tracking with Kalman Filter, Extended Kalman Filter and Particle Filter using Received Signal Strength measurements has been reported in [16] and their simulation results show that PF has superior

Manuscript received Jaunary 14, 2017; accepted July 28, 2018.
A. Namazi Nik, Department of Information and Communication Technology, Payame Noor University, Tehran, Iran. namazi.a@pnurazavi.ac.ir.
*A. Ali Rezaee Department of Information and Communication Technology, Payame Noor University, Tehran, Iran. a_rezaee@pnu.ac.ir.

performance to the KF and EKF in terms of accuracy and root mean square error (RMSE).

The application of PFs in WSNs is challenging due to the limited resources of WSNs. Centralized particle filters (CPFs) have some problems such as consuming significant energy and vulnerability as a single point of failure. Distributed particle filters (DPFs) were studied as a response to these problems, in particular, to offload the computation from the central unit [17].

Particle filtering for target tracking in WSNs has already attracted some attention, including a body of work in distributed methods [18]. Its relation with agent networks has also been explored in [19].

In [20], a fully decentralized particle filtering algorithm for cooperative blind equalization is introduced. The technique is proper, in the sense that it does not make any approximations in the computation of the importance weights of the particles. However, the scheme is applicable only when the state signal is discrete, and would be infeasible in terms of computation and communication among nodes. In [21], the communication load is reduced using quantization and parametric approximations of densities. A similar parametric approach is applied in [18] to further simplify communications.

The work reported in [22] provides a generalized approach for approximating global likelihood through a consensus filter. It approximates log-likelihood by a polynomial function, and the sensors exchange only the coefficients of the polynomial function to compute global likelihood.

The authors in [23] proposed a distributed particle filtering algorithm with the objective of reducing the overhead data that is communicated among the sensors. In their algorithm, the sensors exchange information to collaboratively compute the global likelihood function that encompasses the contribution of the measurements towards building the global posterior density of the unknown location parameters. Each sensor uses its own measurement to compute its local likelihood function and approximates it using a Gaussian function. The sensors then propagate only the mean and covariance of their approximated likelihood functions to other sensors, thereby reducing the communication overhead. The global likelihood function is computed collaboratively from the parameters of the local likelihood functions using an average consensus filter or a forward-backward propagation information exchange strategy.

In [24] a distributed particle filter is designated and it is shown that the difference in accuracy of their proposed DPF and a centralized filter with the same total number of particles is less than 2 cm, while the DPF with four processing nodes is over four times faster than an equivalent centralized version. This equivalently means that the same performance can be obtained on less powerful hardware. The main limitation of that scheme is that every node performing a subset of the computations of the PF should have access to all the observations (i.e., all the measurements collected by the WSN at the current time step) in order to guarantee that the particle weights are proper and, therefore, the resulting estimators are consistent.

3. Nonlinear Filtering in State-Space System

3-1. Bayesian Filtering

Consider the Markov state-space random model with conditionally independent observations [25, 26] described by the triplet:

$$p(x_0), \quad p(x_t|x_{t-1}), \quad p(y_t|x_t), \quad t = 1, 2, \dots \tag{1}$$

We denote the states and the observations up to time t by $x_{0:t} \triangleq \{x_0, \dots, x_t\}$ and $y_{0:t} \triangleq \{y_0, \dots, y_t\}$, respectively. $p(x_0)$ is the prior probability density function (pdf) of the state, the transition density $p(x_t|x_{t-1})$ describes the (random) dynamics of the process x_t and the conditional pdf $p(y_t|x_t)$ describes how the observations are related to the state and it is usually referred to as the likelihood of x_t. The goal of a stochastic filtering algorithm is to recursively estimate the posterior distribution $p(x_t|y_{1:t})$, $t \geq 1$.

Suppose that the required pdf $p(x_{t-1}|y_{1:t-1})$ at time $t-1$ is available. The prediction stage obtains the prior pdf of the state at time t via:

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1} \tag{2}$$

At time step t, an observation $y_t$becomes available, and it may be used to update the prior (update stage) via Bayes' rule:

$$p(x_t|y_{1:t}) \propto p(y_t|x_t)p(x_t|y_{1:t-1}) \tag{3}$$

Eqs. (2) and (3) form the basis for the optimal Bayesian solution [6]. If the system of Eq. (1) is linear and Gaussian then $p(x_t|y_{1:t})$ is Gaussian and can be obtained exactly using the Kalman filter algorithm [27]. If the state space is discrete and finite, exact solutions can also be computed [25]. However, if any of the pdf's in (1) is non-Gaussian, or the system is nonlinear, we have to resort to suboptimal algorithms in order to approximate the filter pdf $p(x_t|y_{1:t})$.

3-2. Particle Filtering

Particle Filters, also known as sequential Monte Carlo methods, are simulation based algorithms that yield estimates of the state based on a random point-mass (or "particle") representation of the probability measure with density $p(x_t|y_{1:t})$ [28-30]. Table 1 shows the standard particle filter algorithm. We refer to it as centralized in order to make explicit that it requires a central unit that collects all the observations together, generates all the particles and processes them together. The resampling step randomly eliminates samples with low importance weights and replicates samples with high importance weights in order to avoid the degeneracy of the importance weights over time [26, 31].

Table 1: The Centralized Particle Filter (CPF) algorithm

```
Initialize: At time $t = 0$
  For $m = 1, \dots, M$
      sample $\boldsymbol{x}_0^{(m)}$ from prior $p(\boldsymbol{x}_0)$

Recursive step: for $t > 0$
  For $m = 1, \dots, M$

      draw $\boldsymbol{x}_t^{(m)} \sim p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}^{(m)})$ and set $\boldsymbol{x}_{0:t}^{(m)} = \{\boldsymbol{x}_t^{(m)}, \boldsymbol{x}_{0:t-1}^{(m)}\}$
      compute importance weights $w_t^{(m)*} = p(\boldsymbol{y}_t|\boldsymbol{x}_t^{(m)})$
  Normalize weights $w_t^{(m)} = w_t^{(m)*}/\sum_{j=1}^{M} w_t^{(j)*}$
  Resample the weighted sample $\{\boldsymbol{x}_{0:t}^{(m)}, w_t^{(m)}\}_{m=1}^{M}$ to obtain
  an unweighted sample $\{\boldsymbol{x}_{0:t}^{(m)}\}_{m=1}^{M}$
```

4. Distributed Particle Filtering

In this paper, we implement a distributed particle filter with nodes that can operate as processing elements (PEs) on a wireless sensor network. Each PE is a low-powered device that has to perform sensing, computation and radio communication tasks while running on batteries. A common assumption in other proposed schemes is that all observations can be readily made available to all PEs in the system [24, 32-33]. Such capacity cannot be taken for granted in a WSN, where the observations are collected locally by the nodes and communications are necessarily constrained because of energy consumption. This issue will be addressed in subsequent sections.

Assume we have N processing nodes in the network and each is capable of running a separate PF with K particles (we ignore any non-processing nodes for now since they do not run particle filters). The total number of particles distributed over the network is M=NK. In particular, after the completion of a full recursive step of the distributed PF at time t-1, the n-th PE should hold the set $\{x_{t-1}^{(n,k)}, w_{t-1}^{(n,k)*}, W_{t-1}^{(n)*}\}_{k=1,\dots,K}$, where $x_{t-1}^{(n,k)}$ is the k-th particle at the n-th PE, $w_{t-1}^{(n,k)*}$ is the corresponding non-normalized importance weight, and $W_{t-1}^{(n)*}$ is the non-normalized aggregated weight of PE n.

Each PF runs locally on a node involves the usual steps of drawing samples, computing weights and resampling. The generation of new particles, the update of the importance weights and the resampling step are taken strictly locally, without interaction between different nodes. To be specific, assume that the transition pdf of model (1) is used as an importance function and that the observation vector y_t is available at every node. Then, at the n-th PE, and for $k = 1, \dots, K$, $x_t^{(n,k)}$ is drawn from the pdf $p(x_t^{(n,k)}|x_{t-1}^{(n,k)})$, and the corresponding nonnormalized weight is computed as $w_t^{(n,k)*} = w_{t-1}^{(n,k)*} p(y_t|x_t^{(n,k)})$.

Hence, the information stored by the n-th node at this point becomes $\{x_t^{(n,k)}, w_t^{(n,k)*}\}_{k=1,\dots,K}$ and the aggregated weight is $W_t^{(n)*} = \sum_{k=1}^{K} w_t^{(n,k)*}$.

Next, a resampling step is taken locally by each PE. Assuming a multinomial resampling algorithm, we assign, for $k = 1, \dots, K$, $x_t^{(n,k)} = x_t^{(n,j)}$ with probability $w_t^{(n,j)}$ and $j \in \{1, \dots, K\}$, where $w_t^{(n,j)} = \frac{w_t^{(n,j)*}}{\sum_{l=1}^{K} w_t^{(n,l)*}}$, $j = 1, \dots, K$, are the locally normalized importance weights. After resampling, the particles at the n-th PE are equally weighted.

In the estimation step, we obtain local estimates of target position at any node as:

$$\hat{x}_t^n = E(x_t|y_{1:t}) = \int x_t p(x_t|y_{1:t})\, dx_t = \sum_{k=1}^{K} w_t^{(n,k)} x_t^{(n,k)} \tag{5}$$

where $w_t^{(n,k)} = w_t^{(n,k)*}/W_t^{(n)*}$, $k = 1, \dots, K$ are the locally normalized importance weights.

Global estimates can be easily computed by a linear combination of local estimates. In order to obtain a global estimate of target position, each node n in the network should transmit its local estimate $\hat{x}_t^n$ and its aggregated weight $W_t^{(n)*}$ to a prescribed node (working as a fusion center) where global estimates can be computed as:

$$\hat{x}_t^{MMSE} = \sum_{k=1}^{K} W_t^{(n)} \hat{x}_t^{(n)} \tag{6}$$

where $W_t^{(n)} = W_t^{(n)*}/\sum_{i=1}^{N} W_t^{(i)*}$ is the globally normalized aggregated weight of the n-th node.

5. Support Vector Machine

Support vector machines discriminate two classes by fitting an optimal linear separating hyperplane to the training samples of two classes in a multidimensional feature space. The optimization problem being solved aims to maximize the margins between the optimal linear separating hyperplane and the closest training samples which are called support vectors (Figure 1). In a linearly non-separable case, the input data are mapped into a high-dimensional space in which the new distribution of the samples enables the fitting of a linear hyperplane [34].

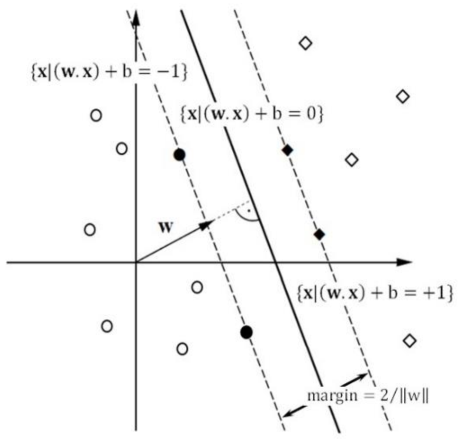


Fig 1. An example of classification of two classes by SVM. The support vectors are filled.

Assume some training data S which are a set of n points of the form:

$$S = \{(x_i, y_i) | x_i \in \mathbb{R}^d, y_i \in \{+1, -1\}\} \quad i = 1, \dots, n \tag{7}$$

where $\mathbb{R}^d$ indicates the class to which point x_i belongs and each x_i is a d-dimensional real vector. The goal of SVM is to define a hyperplane which divides S, such that all the points with the same label are on the same side of the hyperplane while maximizing the distance between the two classes +1, -1 and the hyperplane. The boundary can be expressed as $w.x + b = 0$, where w is the normal vector to the hyperplane. The parameter $\frac{b}{\|w\|}$ determines the perpendicular distance from the hyperplane to the origin along the normal vector w and $\|w\|$ is the Euclidean norm of w. The data points nearest to the boundary are used to define the margins between the two classes and are known as support vectors. At the margins, where the support vectors are located, the equations for classes +1 and -1, respectively, are:

$$w.x + b = +1, \quad w.x + b = -1 \tag{7}$$

and the following decision function can be used to classify any data point in either class +1 or -1:

$$f(x) = sign(w.x + b) \tag{8}$$

The margin between the two classes is measured perpendicular to the hyperplane is $\frac{2}{\|w\|}$, so we want to minimize $\|w\|$. In a linearly separable case, the support vector machine looks for the separating hyperplane with the largest margin. Suppose that all the training data satisfy these constraints:

$$w.x_i + b \geq +1 \quad \forall\, x_i \; with \; y_i = +1 \tag{9}$$

$$w.x_i + b \leq -1 \quad \forall\, x_i \; with \; y_i = -1 \tag{10}$$

These can be combined into one inequality:

$$y_i(w.x_i + b) \geq 1 \quad i = 1, 2, \dots, N \tag{11}$$

where N is the number of training sets. According to [28] it is worth to use Lagrangian formulation of the problem. Thus, introducing Lagrange multipliers $\alpha_i \geq 0$, $i = 1, 2, \dots, N$, one for each of the constraints in Eq. (9), we get the following Lagrangian:

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i y_i (w.x_i + b) + \sum_{i=1}^{N} \alpha_i \tag{12}$$

We must now minimize Eq. (10) with respect to w and b, and maximize it with respect to α_i. Thus:

$$\frac{\partial}{\partial w} L(w, b, \alpha) = 0, \quad \frac{\partial}{\partial b} L(w, b, \alpha) = 0 \tag{13}$$

which leads to:

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i, \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{14}$$

Substituting Eq. (12) into Eq. (10) yields the dual quadratic optimization problem:

Maximize

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j x_i . x_j \tag{15}$$

Subject to

$$\alpha_i \geq 0, \quad i = 1, 2, \dots, N, \tag{16}$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{17}$$

On substitution of Eq. (12) into the decision function (6) we obtain an expression which can be evaluated in terms of dot products between the pattern to be classified and the Support Vectors:

$$f(x) = sign\left(\sum_{i=1}^{N} \alpha_i y_i (x_i . x) + b\right) \tag{18}$$

The dot product can therefore be replaced with a nonlinear kernel function, thereby performing large margin separation in the feature-space of the kernel.

6. Using Support Vector Machine with Distributed Particle Filter

We use LIBSVM [35] in our work. LIBSVM is a library for Support Vector Machines and has gained wide popularity in machine learning and many other areas [36].

The Web address of the package is at http://www.csie.ntu.edu.tw/~cjlin/libsvm. Also, we use the MATLAB software to plot the results.

A classification task usually involves separating data into training and testing sets. Each instance in the training set contains one “target value” (i.e. the class labels) and several “attributes” (i.e. the features or observed variables). The goal of SVM is to produce a model (based on the training data) which predicts the target values of the test data given only the test data attributes. Our idea is to make use of support vector machine as a data classification technique in our work to reduce communications among the nodes.

As we mentioned in section 4 in the weight update step we assume that the observation vector y_t is available at every node which involves communications among the nodes. We use SVM to reduce these communications. SVMs only consider points near the margin (support vectors) instead of whole data points. According to our assumption, the observation coming from sensor j at time t, denoted $y_{j,t}$, is modeled as a binary observation. Then our SVM has two classes. Each sensor has two attributes which are equal to the coordinates of its position.

Scaling before applying SVM is very important. The main advantage of scaling is to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges. Another advantage is to avoid numerical difficulties during the calculation. Because kernel values usually depend on the inner products of feature vectors, e.g. the linear kernel and the polynomial kernel, large attribute values might cause numerical problems. In [37] it is recommended to linearly scale each attribute to the range [-1, +1] or [0, 1]. We have to use the same method to scale both training and testing data. For example, suppose that we scaled the first attribute of training data from [-10, +10] to [-1, +1]. If the first attribute of testing data lies in the range [-11; +8], we must scale the testing data to [-1.1, +0.8]. There are four basic kernel functions in SVM, including linear, polynomial, radial basis function (RBF) and sigmoid. In our work we have used RBF kernel in the training step since it has fewer numerical difficulties and has better performance in nonlinear cases.

When the training is done, support vectors are generated. Once the support vectors are determined, the rest of the feature set can be discarded, since the support vectors contain all the necessary information for the classifier. We propagate observations corresponding to these support vectors ($\bar{y}_t$) rather than the whole y_t in the network. Then, in the weight update step of our distributed particle filter, every processing element can obtain observations of other sensors by running the final step of the SVM, namely prediction. On the other hand, in the prediction step of SVM, we obtain observation vector y_t from vector $\bar{y}_t$. Table 2 summarizes the DPF algorithm investigated in this paper.

Table 2. Distributed Particle Filter (DPF) algorithm

```
Initialize: At time $t = 0$, for $n = 1, \dots \dots N$
    Draw $\boldsymbol{x}_0^{(n,k)}$, for $k = 1, \dots, K$, from prior $p(\boldsymbol{x}_0)$
    Assign $w_0^{(n,k)*} = \frac{1}{K}$ for all k, set $W_0^{(n)*} = 1$
    Build the set $\{\boldsymbol{x}_0^{(n,k)}, w_0^{(n,k)*}, W_0^{(n)*}\}_{k=1}^{K}$
Recursive step: At time $t > 0$, start from the set $\{\boldsymbol{x}_{t-1}^{(n,k)}, w_{t-1}^{(n,k)*}, W_{t-1}^{(n)*}\}_{k=1}^{K}$. Then, for $n = 1, \dots, N$
    Sampling: Draw $\boldsymbol{x}_t^{(n,k)}$ from $p(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}^{(n,k)})$, for $k = 1, \dots, K$
    Weight update: $w_t^{(n,k)*} = w_{t-1}^{(n,k)*} p(\boldsymbol{y}_t | \boldsymbol{x}_t^{(n,k)})$
    Estimation: compute the desired output, such as the expected value
    Resampling: to obtain the set $\{\boldsymbol{x}_t^{(n,k)}, w_t^{(n,k)*}, W_t^{(n)*}\}_{k=1}^{K}$, where $w_t^{(n,k)*} = W_t^{(n)*}/K$   for $k = 1, \dots, K$
```

7. Simulation and Experimental Results

The goal of our work is to implement a DPF for target tracking in a wireless sensor network and use SVM to compress measurements collected by these sensors. Our experimental scenario is shown in Figure 2. It is a room with 10 nodes (which are equipped with a light sensor) enclosing an area of 4×6 m^2 with a single source of natural light (a window). Modeling environment specifications and translating the disturbances caused by the target in the sensor readings into distance measurements are very complex. Then, instead we emphasize on obtaining binary observations: 1 if the target is in the detection zone and 0 otherwise.

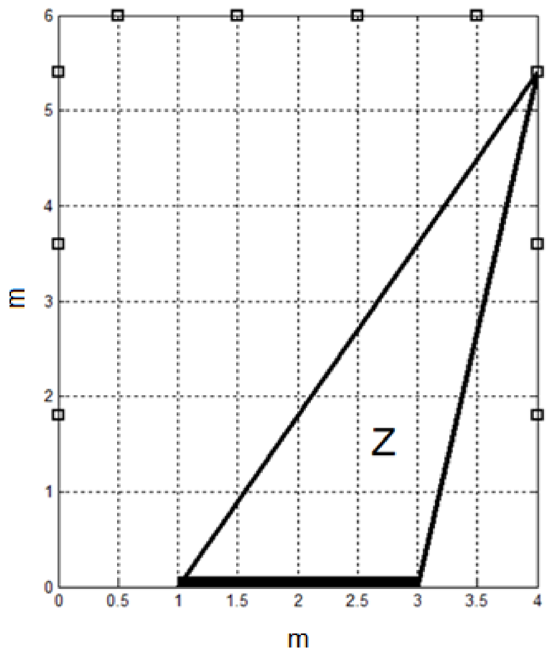


Fig. 2 Tracking scenario of 4×6 m^2. The thick line is the light source. There are 10 nodes equipped with light sensors around the edges, indicated by squares. The entry to the scenario lies at the bottom-right corner.

Table 3 displays values of the relevant simulation and algorithm parameters. The number of processing elements (N) is 4 in our experiments and we use N=1 as the equivalent to a centralized particle filter. Changing N affects other variables, such as the number of sensing-only elements (J-N) and the number of particles per PE (K=M/N). It does not matter which of the nodes are PEs and which are SEs, since we assume a fully connected network. Each node (either PE or SE) produces one binary observation every T_s second.

Figure 3 displays the empirical distribution of errors, and the average error, for 100 simulated paths. Figure 4 plots two selection of these paths along with the path estimated by our SVM-based DPF. The dissensions between true and estimated position tend to happen when the target moves between detection zones. Since the observations are binary and zone-based, rather than distance-based, there are gaps around the edges (see for example the final points in Figure 4). Accuracy also tends to be higher nearer the light source where more detection zones overlap.

Table 3. Simulation and algorithm parameters

Variable	Symbol	Value (unit)
Number of PEs	N	4
Number of nodes	J	10
Number of SEs		J-N
Total number of particles	M	100
Number of particles/PE	K	M/N
Number of timesteps	T	20 (s)
Sampling period	T_s	1 (s)

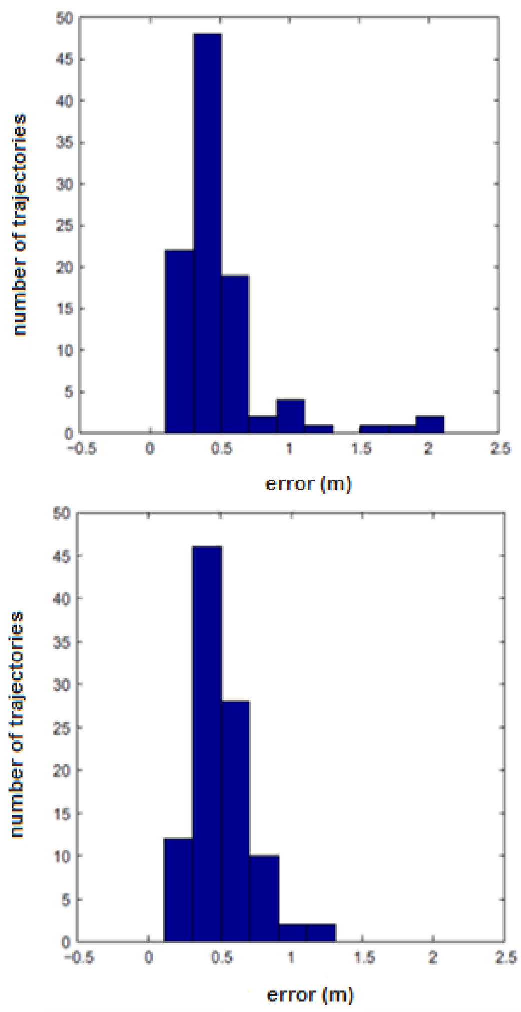
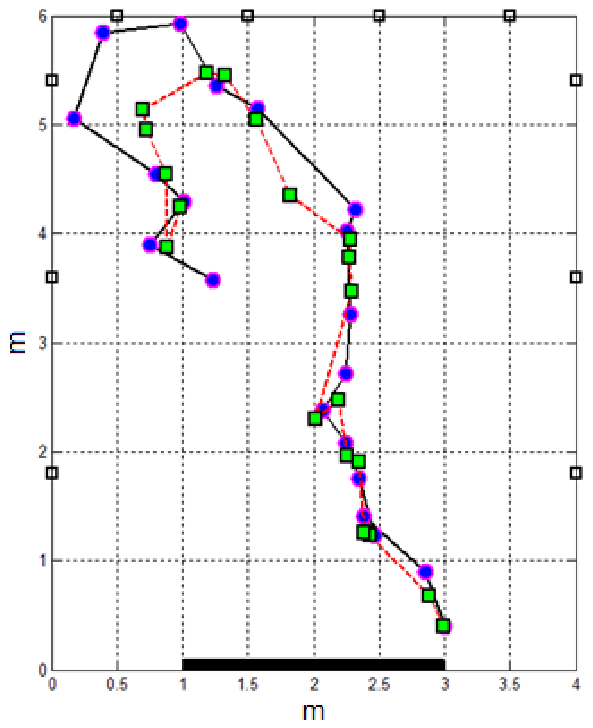


Fig 3. Histogram of position error in meters for both the centralized (up) and our distributed (down) versions of the particle filter over 100 simulated trajectories.



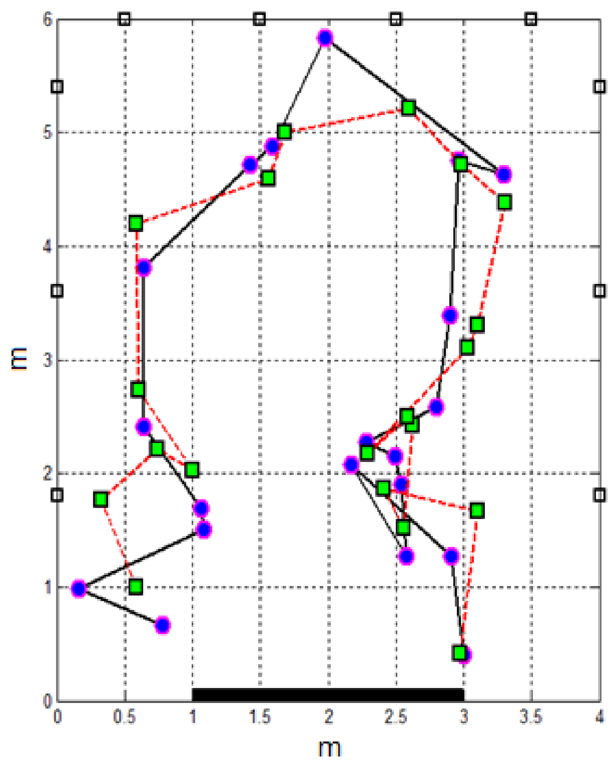


Fig 4. The simulated (black) path for two simulations, and the corresponding SVM-based DPF-estimated path (red); over T=20 time steps.

Figure 5 displays the amount of saving in the volume of propagating information for updating particle weights, using the proposed method, for 100 simulated paths. The horizontal axis shows the simulation run and the vertical axis shows the amount of propagating observations (in percent) on the network compared to the case when SVM is not used. The results show that using the proposed scheme, only %51.9 of sensor observations are propagated on the network, compared to the work done in [24], that leads to saving energy consumption of sensors.

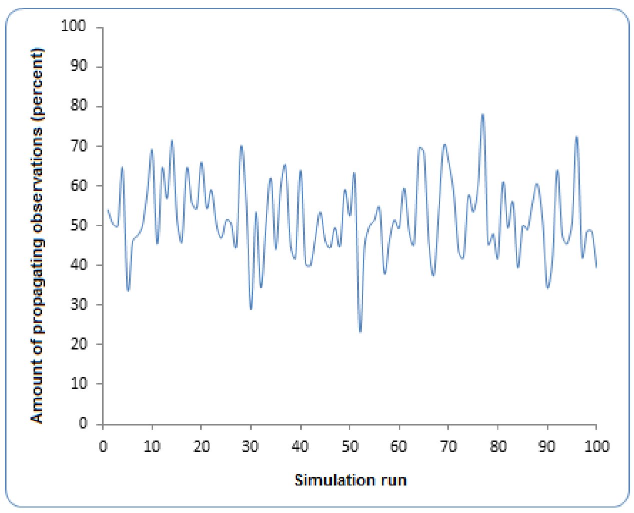


Fig 5. The amount of saving in the volume of propagating information for updating particle weights, using our proposed method, for 100 simulated paths.

8. Conclusion

In this paper, we have described the implementation of a distributed particle filter for target tracking in a wireless sensor network. One of the main limitations of similar works is the need to make all sensor observations available to every processing node. To overcome this limitation, we have used support vector machine to compress sensor observations. Simulation results show that the difference in accuracy of the proposed scheme and centralized particle filter and also distributed particle filter are insignificant, whereas by combining SVM with DPF we have reduced communications among the nodes around %48. Since SVMs only consider points near the margin (support vectors) instead of whole data points, they are suitable for data compression. SVMs can produce accurate and robust classification results on a sound theoretical basis, even when input data are non-monotone and non-linearly separable. The biggest limitation of the support vector approach lies in choice of the kernel function. In our work, we have used RBF kernel in the training step since it has fewer numerical difficulties and has better performance in nonlinear cases.

References

[1] E. Cayirci, H. Tezcan, Y. Dogan, and V. Coskun, "Wireless sensor networks for underwater survelliance systems", *Ad Hoc Networks*, vol. 4, pp. 431-446, 2006.

[2] S. Santini, B. Ostermaier, and A. Vitaletti, "First experiences using wireless sensor networks for noise pollution monitoring", *presented at the Proceedings of the workshop on Real-world wireless sensor networks*, Glasgow, Scotland, 2008.

[3] H.-W. Tsai, C.-P. Chu, and T.-S. Chen, "Mobile object tracking in wireless sensor networks", *Computer Communications,* vol. 30, pp. 1811-1825, 6/8/ 2007.

[4] A. Ribeiro, G. B. Giannakis, and S. I. Roumeliotis, "SOI-KF: Distributed Kalman Filtering With Low-Cost Communications Using the Sign of Innovations", *IEEE Transactions on Signal Processing,* vol. 54, pp. 4782-4795, 2006.

[5] A. Dhital, P. Closas, and C. Fernández-Prades, "Bayesian filtering for indoor localization and tracking in wireless sensor networks", *EURASIP Journal on Wireless Communications and Networking,* vol. 2012, pp. 1-13, 2012.

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

[7] P. M. Djuric, M. Vemula, and M. F. Bugallo, "Target Tracking by Particle Filtering in Binary Sensor Networks", *IEEE Transactions on Signal Processing,* vol. 56, pp. 2229-2238, 2008.

[8] Y. Huang, W. Liang, H.-b. Yu, and Y. Xiao, "Target tracking based on a distributed particle filter in underwater sensor networks", *Wireless Communications and Mobile Computing,* vol. 8, pp. 1023-1033, 2008.

[9] S. Sarkka, "Bayesian Filtering and Smoothing", *Cambridge University Press,* 2013.

[10] H. Q. Liu, H. C. So, F. K. W. Chan, and K. W. K. Lui, "Distributed particle filter for target tracking in sensor networks", *Progress In Electromagnetics Research C,* vol. 11, pp. 171-182, 2009.

[11] K. Achutegui, L. Martino, J. Rodas, C. J. Escudero, and J. Miguez, "A multi-model particle filtering algorithm for indoor tracking of mobile terminals using RSS data", *in Control Applications,* (CCA) & Intelligent Control, (ISIC), 2009 IEEE, 2009, pp. 1702-1707.

[12] O. Hlinka, F. Hlawatsch, and P. M. Djuric, "Distributed particle filtering in agent networks: A survey, classification, and comparison", *IEEE Signal Processing Magazine,* vol. 30, pp. 61-81, 2013.

[13] A. Oracevic and S. Ozdemir, "A survey of secure target tracking algorithms for wireless sensor networks", *in Proceedings of the World Congress on Computer Applications and Information Systems (WCCAIS '14),* pp. 1–6, IEEE, Hammamet, Tunisia, January 2014.

[14] O. Demigha, W.-K. Hidouci, and T. Ahmed, "On energy efficiency in collaborative target tracking in wireless sensor network: a review", *IEEE Communications Surveys and Tutorials,* vol. 15, no. 3, pp. 1210–1222, 2013.

[15] A. Ez-Zaidi, S. Rakrak, "A Comparative Study of Target Tracking Approaches in Wireless Sensor Networks", *Journal of Sensors,* vol. 2016, p. 11, 2016.

[16] M. W. Khan, N. Salman, A. Ali, A. M. Khan, and A. H. Kemp, "A comparative study of target tracking with Kalman filter, extended Kalman filter and particle filter using received signal strength measurements", *in Emerging Technologies (ICET),* 2015 International Conference on. IEEE, 2015.

[17] B. Jiang, B. Ravindran, "Completely Distributed Particle Filters for Target Tracking in Sensor Networks", *IEEE International Parallel & Distributed Processing Symposium (IPDPS),* pp. 334-344, 2011.

[18] O. Hlinka, O. Sluciak, F. Hlawatsch, P. Djuric, M. Rupp, "Distributed Gaussian particle filtering using likelihood consensus", *in: International Conference on Acoustics,* Speech and Signal Processing, pp. 3756–3759, May 2011.

[19] O. Hlinka, F. Hlawatsch, P. Djuric, "Distributed particle filtering in agent networks", *IEEE Signal Process.* Mag. pp. 61-81 (January) (2013).

[20] Claudio J. Bordin, Marcelo G. S. Bruno, "Cooperative bling equalization of frequency-selective channels in sensor networks using decentralized particle filtering", *in: 42nd Asilomar Conference on Signals, Systems and Computers,* pp. 1198–1201, October 2008.

[21] Mark Coates, "Distributed particle filters for sensor networks", *in: The International Conference on Information Processing in Sensor Networks,* (IPSN),

pp. 99–107, April 2004.

[22] O. Hlinka, P. Djuric, F. Hlawatsch, "Consensus-based distributed Particle Filtering with distributed proposal adaptation", *IEEE Trans. Signal Process.*, vol 62, pp. 3029–3041, 2014.

[23] T. Ghirmai, "Distributed Particle Filter for Target Tracking: With Reduced Sensor Communications", Sensors, 16, 1454, 2016.

[24] J. Read, K. Achutegui, and J. Míguez, “A distributed particle filter for nonlinear tracking in wireless sensor networks”, *Signal Processing,* vol. 98, pp. 121-134, 2014.

[25] A. Doucet, N. d. Freitas, and N. Gordon, “Sequential Monte Carlo Methods in Practice” Springer, 2001.

[26] B. Ristic, S. Arulampalam, and N. Gordon, “Beyond the Kalman Filter: Particle Filters for Tracking Applications”, Artech House, 2004.

[27] R. E. Kalman, “A New Approach to Linear Filtering and Prediction Problems”, *Journal of Basic Engineering,* vol. 82, pp. 35-45, 1960.

[28] A. Bain and D. Crisan, “Fundamentals of Stochastic Filtering”, Springer, 2009.

[29] O. Cappe, S. J. Godsill, and E. Moulines, “An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo”, *Proceedings of the IEEE,* vol. 95, pp. 899-924, 2007.

[30] Djuric, x, P. M., J. H. Kotecha, Z. Jianqui, H. Yufei, et al., “Particle filtering”, *IEEE Signal Processing Magazine,* vol. 20, pp. 19-38, 2003.

[31] L. Tiancheng, M. Bolic, and P. M. Djuric, “Resampling Methods for Particle Filtering: Classification, implementation, and strategies”, *IEEE Signal Processing Magazine,* vol. 32, pp. 70-86, 2015.

[32] M. Bolic, P. M. Djuric, and H. Sangjin, “Resampling algorithms and architectures for distributed particle filters”, *IEEE Transactions on Signal Processing,* vol. 53, pp. 2442-2450, 2005.

[33] J. Míguez, “Analysis of parallelizable resampling algorithms for particle filtering”, *Signal Processing,* vol. 87, pp. 3155-3174, 2007.

[34] C. C. Burges, “A Tutorial on Support Vector Machines for Pattern Recognition”, *Data Mining and Knowledge Discovery,* vol. 2, pp. 121-167, 1998/06/01 1998.

[35] https://www.csie.ntu.edu.tw/~cjlin/libsvm.

[36] C.-c. Chang and C.-J. Lin, “LIBSVM: a library for support vector machines”, *ACM Transactions on Intelligent Systems and Technology,* vol. 2, 2011.

[37] C.-w. Hsu, C.-c. Chang, and C.-j. Lin, “A practical guide to support vector classification”, 2010.