15th European Signal Processing Conference (EUSIPCO 2007), Poznan, Poland, September 3-7, 2007, copyright by EURASIP

DISTRIBUTED SEQUENTIAL MONTE CARLO ALGORITHMS FOR NODE LOCALIZATION AND TARGET TRACKING IN WIRELESS SENSOR NETWORKS

Joaquín Míguez and Antonio Artés-Rodríguez

Department of Signal Theory and Communications, Universidad Carlos III de Madrid
Avenida de la Universidad 30, 28911 Leganés (Madrid), Spain.
E-mail: joaquin.miguez@uc3m.es, antonio@ieee.org

ABSTRACT

We address the problem of tracking a maneuvering target that moves along a region monitored by a wireless sensor network (WSN) whose nodes, including sensors and data fusion centers (DFCs), are located at unknown positions. Therefore, the target trajectory, its velocity and all node locations must be estimated jointly, without assuming the availability of any "beacons" with known location that can be used as a reference. We introduce a new method that comprises: (i) a combination of Monte Carlo optimization and iterated importance sampling to yield and initial population of node locations with high posterior probability (given data collected at the network startup) and (ii) a sequential Monte Carlo (SMC) algorithm for recursively tracking the target position and velocity and sequentially re-generating new populations of node positions as new observations become available. The resulting algorithm is implemented in a distributed fashion. Assuming that the communication capabilities of the DFCs enable them to share some data, each DFC can run an independent SMC algorithm and produce local estimates of the magnitudes of interest. Optimal data fusion is achieved by a linear combination of the local estimates with adequate weights. We illustrate the application of the algorithm in a network of power-aware sensors.

1. INTRODUCTION

Most applications of wireless sensor networks (WSN) rely on the accurate localization of the network nodes [1]. In particular, for network-based navigation and tracking applications it is usually assumed that the sensors, and possibly any data fusion centers (DFCs) in charge of processing the data collected by the network, are placed at *a priori* known locations. Alternatively, if the number of nodes is too large, WSNs are usually equipped with beacons that can be used as a reference to locate the remaining nodes [2]. In both scenarios, the accuracy of node localization depends on some external system that must provide the position of either whole set of nodes or, at least, the beacons [1]. Although beacon-free network designs are feasible [2, 3], they usually involve complicated and energy-consuming local communications among nodes which should, ideally, be very simple.

In this paper, we address the problem of tracking a maneuvering target that moves along a region monitored by a WSN whose nodes, including both the sensors and the DFCs, are located at unknown positions. Therefore, the target trajectory, its velocity and all node locations must be estimated jointly, without assuming the availability of beacons. We introduce a new method that consists of three stages: initialization of the WSN, target tracking and data fusion. At initialization, the network collects information related to the distances among nodes that is collected by the DFCs. We use a combination of random search optimization [4] and iterated importance sampling [5] to produce an initial population of node locations, approximately distributed according to their posterior probability distribution given the available data. This population is the input to the second stage. Target tracking is performed by means of sequential Monte Carlo (SMC) method (i.e., a *particle filter*) [6, 7, 8, 9, 10] that recursively tracks the target position and velocity and improves the node positioning as new observations are collected by the WSN. Assuming the ability of the DFCs to share some amount of data, we also propose a distributed implementation of the particle filter which enables each DFC to run an independent particle filter and obtain local Bayesian estimates of both the target state and the nodes. One advantage of the proposed scheme is that optimal (Bayesian) data fusion can be obtained by linear combination of the local estimates.

The remaining of the paper is organized as follows. After a brief comment on the notation to be used, the system model that we assume in this work is described in detail in Section 2. The proposed algorithm is described in Section 3. In Section 4 we present illustrative computer simulation results for a network of power-aware sensors and, finally, Section 5 is devoted to the conclusions.

1.1 Notation

Scalar magnitudes are denoted as regular letters, e.g., x, N. Vectors and matrices are denoted as lower-case and upper-case bold-face letters, respectively, e.g., vector $\mathbf{x}$ and matrix $\mathbf{X}$. We use $p(\cdot)$ to denote the probability density function (pdf) of a random magnitude. This is an argument-wise notation, i.e., $p(x)$ denotes the pdf of x and $p(y)$ is the pdf of y, possibly different. The conditional pdf of x given the observation of y is written as $p(x|y)$. Sets are denoted using calligraphic letters, e.g., $\mathscr{D}$. Specific sets built from sequences of elements are denoted as $x_{1:N} = \{x_1, \ldots, x_N\}$.

2. SYSTEM MODEL

We assume that the target moves along a 2-dimensional region $\mathscr{C} \subseteq \mathbb{C}^2$ (i.e., a compact subset of the complex plane) according to the linear model [8]

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

where $\mathbf{x}_t = [r_t, v_t]^\top \in \mathbb{C}^2$ is the target state, which includes its position and its velocity at time t, r_t and v_t, respectively; $\mathbf{A} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}$ is a transition matrix that depends on the observation period, T, and $\mathbf{u}_t = [u_{r,t}, u_{v,t}]^\top \sim CN(\mathbf{u}_t|\mathbf{0}, \mathbf{C}_u)$ is a complex Gaussian noise term, with zero mean and known covariance matrix $\mathbf{C}_u = \sigma_u^2 \begin{bmatrix} \frac{1}{4}T^4 & 0 \\ 0 & T^2 \end{bmatrix}$, that accounts for unknown acceleration forces. The initial target state, $\mathbf{x}_0$, has a known prior probability density function (pdf), $p(\mathbf{x}_0) = p(r_0)p(v_0)$, and we assume $p(v_0) = CN(0, \sigma_{v,0}^2)$, i.e., the prior pdf of the velocity random process is complex Gaussian with zero mean and variance, $\sigma_{v,0}^2$.

The network consists of N_s sensors and N_c DFCs. Sensors are located at random (but fixed) unknown positions $s_{1:N_s} := \{s_1, s_2, \ldots, s_{N_s}\}$, $s_i \in \mathbb{C}^2$, with independent and identical uniform prior pdf's, $p(s_i) = U(\mathscr{C})$, $i = 1, \ldots, N_s$, on the 2-dimensional region monitored by the WSN. During the network startup, each sensor detects any other nodes which are located within a certain range, $S_u > 0$. In particular, the n-th sensor builds up an $N_s \times 1$ vector of decisions, $\mathbf{b}_n = [b_{n,1}, \ldots, b_{n,N_s}]^\top$, where (deterministically) $b_{n,n} = 1$

while $b_{n,k} \in \{1,0\}$, $n \neq k$, is a binary random variable with probability mass function (pmf) given by

$$p(b_{n,k} = 1 | s_{1:N_s}) = p_d(d^s_{n,k}, S_u), \quad (2)$$

where $d^s_{n,k} = |s_n - s_k|$ is the distance between the n-th and k-th sensors and $p_d(\cdot,\cdot)$ is the function that yields the probability of detection. At time 0, these decisions are broadcast to the DFCs and we collect them all together in the $N_s \times N_s$ matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_{N_s}]$ for notational convenience.

The locations of the N_c DFCs are denoted as $c_1, \ldots, c_{N_c}$, with $c_i \in \mathscr{C}$ $\forall i$. By convention, the first DFC is assumed to be located at the origin of the monitored region, i.e., $c_1 = 0$. The positions of the remaining DFCs are assumed random (but fixed) and unknown, with complex Gaussian pdf's $p(c_i) = CN(c_i | \mu^c_i, \sigma^2_c)$, $i = 2, \ldots, N_c$. The physical implication of this model is that DFCs are deployed at locations which are *approximately*, but not *exactly*, known. The variance σ^2_c indicates the uncertainty in this prior knowledge.

During the normal operation of the network, the n-th sensor periodically measures some distance-dependent physical magnitude related to the target. The measurement obtained by the n-th sensor at discrete-time $t \geq 1$ is denoted as $y_{n,t} = f_s(d_{n,t}, \varepsilon^y_{n,t})$, where $d_{n,t} = |r_t - s_n|$ is the distance between the target and the sensor, $\varepsilon^y_{n,t}$ is a random perturbation with known pdf and $f_s(\cdot,\cdot)$ is the measurement function. We assume that not every sensor necessarily transmits its observation, $y_{n,t}$, at every time. Indeed, it is often convenient (in order to reduce energy consumption) that only a subset of sensors become active and transmit their measurements. The local decision of a sensor to transmit its data or not depends on the comparison of the measurement, $y_{n,t}$, with some reference value, S_y. We also introduce a certain probability of transmission failure, β. A failure can be caused, e.g., by a strong interference in the channel that prevents adequate reception of the communication signal at the DFC. Thus, at time t only an $N_t \times 1$ vector of observations, $\mathbf{y}_t = [y_{\kappa(1),t}, \ldots, y_{\kappa(N_t),t}]^\top$, where $0 \leq N_t \leq N_s$ and $\kappa(i) \in \{1, \ldots, N_s\}$, $\forall i$, is effectively broadcast to the DFCs (note that *all* DFCs collect *the same* data from the sensors). We assume that the likelihood $p(\mathbf{y}_t | r_t, s_{1:N_s}, c_{1:N_c})$ can be evaluated.

Each DFC has the capability to extract some distance-related magnitude from the communication signals transmitted by the sensors. For simplicity, we consider the same type of measurement carried out at the sensors, hence the n-th DFC also has available, at time $t \geq 0$, the $N_t \times 1$ data vector $\mathbf{z}_{n,t} = [z_{\kappa(1),n,t}, \ldots, z_{\kappa(N_t),n,t}]^\top$, where $z_{i,n,t} = f_s(d^c_{i,n,t}, \varepsilon^z_{i,n,t})$, $d^c_{i,n,t} = |s_i - c_n|$ and $\varepsilon^z_{i,n,t}$ is a random perturbation with known pdf, so that the likelihood $p(\mathbf{z}_{n,t} | s_{1:N_s}, c_n)$ can be computed. Note that $\mathbf{z}_{n,0}$ is defined (unlike $\mathbf{y}_0$), and has dimension $N_0 = N_s$, because during the network startup all sensors broadcast signals to the DFCs.

We assume that the DFCs are equipped with communication devices more sophisticated than those at the sensor nodes and, as a consequence, it is feasible to exchange data among the DFCs. In particular, during network startup one DFC collects a set of $N_c(N_c - 1)$ observations, $q_{i,n} = f_s(d^0_{i,n}, \varepsilon^0_{i,n})$, $i,n \in \{1, \ldots, N_c\}$ (but $i \neq n$), where $d^0_{i,n} = |c_i - c_n|$ and $\varepsilon^0_{i,n}$ is a random perturbation with known pdf, so that $p(q_{i,n} | c_{1:N_c})$ can be evaluated. For compactness, we define the set $\mathscr{Q} = \{q_{i,n}\}^{i \neq n}_{i,n \in \{1,\ldots,N_c\}}$. Moreover, during normal operation of the WSN, each DFC receives sufficient information from the other fusion nodes to build the $N_t \times N_c$ matrix of observations $\mathbf{Z}_t = [\mathbf{z}_{1,t}, \ldots, \mathbf{z}_{N_c,t}]$. Essentially, this means that the DFCs must be capable of *sharing* data.

The goal is to jointly estimate the target states $\mathbf{x}_{0:t} = \{\mathbf{x}_0, \ldots, \mathbf{x}_t\}$, the sensor locations, $s_{1:N_s}$ and the uknown DFC positions, $c_{2:N_c}$, from the decisions in $\mathbf{B}$, the data in $\mathscr{Q}$ and the sequences of observation arrays $\mathbf{y}_{1:t} = \{\mathbf{y}_1, \ldots, \mathbf{y}_t\}$ and $\mathbf{Z}_{0:t} = \{\mathbf{z}_0, \ldots, \mathbf{z}_t\}$.

3. ALGORITHMS

3.1 Mixture Kalman Filtering

For convenience of exposition, let us begin with the case in which the locations of the DFCs, $c_{1:N_c}$, and the sensors, $s_{1:N_s}$, are known. If we aim at the Bayesian estimation of the sequence of target positions $r_{0:t}$ conditional on the observations $\mathbf{y}_{1:t}$ (given $c_{1:N_c}$ and $s_{1:N_s}$, $\mathscr{Q}$, $\mathbf{B}$ and $\mathbf{Z}_{0:t}$ are not relevant for the estimation problem), all statistical information is contained in the posterior pdf $p(r_{0:t} | \mathbf{y}_{1:t}, s_{1:N_s})$, which can be approximated by means of a particle filter. Specifically, the dynamic model (1) is linear in r_t conditional on v_t, hence the recursive decomposition

$$p(r_{0:t} | \mathbf{y}_{1:t}, s_{1:N_s}) \propto p(\mathbf{y}_t | r_t, s_{1:N_s}) p(r_t | r_{0:t-1}) p(r_{0:t-1} | \mathbf{y}_{1:t-1}, s_{1:N_s}) \quad (3)$$

enables the application of the mixture Kalman filter (MKF) technique [11] to build a point-mass approximation of the posterior pdf,

$$p_M(r_{0:t-1} | \mathbf{y}_{1:t-1}, s_{1:N_s}) = \sum_{i=1}^{M} w^{(i)}_{t-1} \delta(r_{0:t-1} - r^{(i)}_{0:t-1}), \quad (4)$$

where $\delta(\cdot)$ is the Dirac delta funstion, $\{r^{(i)}_{0:t-1}\}^M_{i=1}$ are samples in the space of $r_{0:t-1}$ and $\{w^{(i)}_t\}^M_{i=1}$ are normalized importance weights ($\sum^M_{i=1} w^{(i)}_t = 1$) [6]. The M weighted samples are usually termed *particles* and, given the set $\Omega^{mkf}_{t-1} = \{r^{(i)}_{0:t-1}, w^{(i)}_{t-1}\}^M_{i=1}$, we can apply the sequential importance sampling (SIS) [6] algorithm to recursively compute Ω^{mkf}_t. For $i = 1, \ldots, M$, the following steps are recursively applied:

1. Importance sampling: Draw $r^{(i)}_t \sim p(r_t | r^{(i)}_{0:t-1})$.
2. Weight update: $w^{(i)^*}_t = w_{t-1} p(\mathbf{y}_t | r^{(i)}_t, s_{1:N_s})$ and $w^{(i)}_t = w^{(i)^*}_t / \sum^M_{k=1} w^{(k)^*}_t$.

Resampling steps also need to be applied (although not for each t) to avoid weight degeneracy [6]. Since the likelihood $p(\mathbf{y}_t | r^{(i)}_t, s_{1:N_s})$ can be computed, by assumption of the model, and several methods are available for resampling, the only difficulty in the application of this algorithm is sampling from the prior $p(r_t | r^{(i)}_{0:t-1})$. The latter can be obtained from the Kalman filter (KF) equations [12]. To be specific, let us note that the pair of equations jointly given by (1),

$$v_t = v_{t-1} + u_{v,t} \quad (5)$$

$$\Delta_t = r_t - r_{t-1} = T v_{t-1} + u_{r,t} \quad (6)$$

form a linear-Gaussian system, hence the posterior pdf of v_t, given a specific sequence $r_{0:t}$, is complex Gaussian, $p(v_t | \Delta_{1:t}, r_0) = CN(v_t | \mu^v_t, \sigma^2_{v,t})$, with posterior mean and variance, μ^v_t and $\sigma^2_{v,t}$, respectively, that can be recursively computed using the KF recursion [13]. Moreover, the normalization constant of $p(v_t | \Delta_{1:t}, r_0)$ is $p(\Delta_t | \Delta_{1:t-1}, r_0) = p(r_t | r_{0:t-1})$, which is also complex Gaussian and can be analytically found [11].

Therefore, the outlined SIS algorithm can be implemented using a bank of M KFs, one per particle. Given $r^{(i)}_{0:t-1}$, it is possible to (recursively and analytically) obtain $p(v_{t-1} | r^{(i)}_{0:t-1}) = CN(v_{t-1} | \mu^{v^{(i)}}_{t-1}, \sigma^{2^{(i)}}_{v,t-1})$ and, as a consequence, $p(r_t | r^{(i)}_{0:t-1}) = CN(r_t - r^{(i)}_{t-1} | T\mu^{v^{(i)}}_{t-1}, \sigma^{2^{(i)}}_{r,t|t-1})$, which can be easily sampled to draw $r^{(i)}_t$ in the importance sampling step. Moreover, both r_t and v_t can be estimated (in the minimum mean square error sense),

$$r^{mmse}_t = \sum_{i=1}^{M} w^{(i)}_t r^{(i)}_t, \quad v^{mmse}_t = \sum_{i=1}^{M} w^{(i)}_t \mu^{v^{(i)}}_t, \quad (7)$$

by combining the outputs of the KF's, hence the name MKF. This methodology was applied to generic tracking problems in [8].

3.2 Distributed MKF

The tracking problem becomes considerably more involved when the node locations, $s_{1:N_s}$ and $c_{2:N_c}$, are unknown. Handling both static and dynamic random magnitudes using SMC methods is, by itself, a nontrivial task because the overall dynamic system is not ergodic [7]. Moreover, in our particular setup, the high dimension of the random fixed parameters (N_s+N_c-1 complex variables need to be estimated) makes an approach based on sampling-only very inefficient and, therefore, we propose to combine Monte Carlo sampling *and* optimization to obtain better performance.

Assume estimates of $c_{2:N_c}$ are available. The basic probabilistic relationship that we exploit to derive the new algorithm in this paper is obtained by means of Bayes theorem and the repeated decomposition of conditional probabilities, namely

$$\begin{aligned} p(r_{0:t},s_{1:N_s}|\mathbf{y}_{1:t},\mathbf{Z}_{0:t},\mathbf{B}) &\propto p(\mathbf{y}_t|r_t,s_{1:N_s})p(\mathbf{Z}_t|s_{1:N_s}) \\ &\times p(r_t|r_{0:t-1})\rho_t(s_{1:N_s})p(r_{0:t-1}|\mathbf{y}_{1:t-1},\mathbf{Z}_{0:t-1},\mathbf{B}) \quad (8)\\ &= \textstyle\prod_{k=1}^{t} p(\mathbf{y}_k|r_k,s_{1:N_s})p(\mathbf{Z}_k|s_{1:N_s})p(r_k|r_{0:k-1}) \\ &\times p(r_0)p(s_{1:N_s}|\mathbf{Z}_0,\mathbf{B}) \quad (9)\end{aligned}$$

where

$$\rho_t(s_{1:N_s}) = p(s_{1:N_s}|r_{0:t-1},\mathbf{y}_{1:t-1},\mathbf{Z}_{0:t-1},\mathbf{B}) \quad (10)$$

is the posterior pdf of $s_{1:N_s}$ at time $t-1$.

Assume that we are able to draw samples from $\rho_t(s_{1:N_s})$. Then, (8) shows that a SMC algorithm can be used to recursively approximate $p(r_{0:t},s_{1:N_s}|\mathbf{y}_{1:t},\mathbf{Z}_{0:t},\mathbf{B})$. Indeed, if at time $t-1$ the set of particles $\Omega_{t-1}^{mkf} = \{r_{0:t-1}^{(i)}, s_{t-1,1:N_s}^{(i)}, w_t^{(i)}\}_{i=1}^M$ is available, then we can compute a point-mass approximation the last factor in (8),

$$\begin{aligned} &p_M(r_{0:t-1}|\mathbf{y}_{1:t-1},\mathbf{Z}_{0:t-1},\mathbf{B}) = \\ &\int \textstyle\sum_{i=1}^M w_{t-1}^{(i)} \delta(r_{0:t-1}-r_{0:t-1}^{(i)})\delta(s_{1:N_s}-s_{t-1,1:N_s}^{(i)})ds_{1:N_s} \quad (11)\\ &= \textstyle\sum_{i=1}^M w_{t-1}^{(i)} \delta(r_{0:t-1}-r_{0:t-1}^{(i)}), \quad (12)\end{aligned}$$

where the integrand in (11) is the approximation of $p(r_{0:t-1},s_{1:N_s}|\mathbf{y}_{1:t-1},\mathbf{Z}_{0:t-1},\mathbf{B})$ built from Ω_{t-1}^{mkf}. Eq. (12) implies that we can start from the set $\tilde{\Omega}_{t-1}^{mkf} = \{r_{0:t-1}^{(i)}, w_t^{(i)}\}_{i=1}^M$ and exploit (8) to build Ω_t^{mkf} via the MKF algorithm. Specifically,

$$\begin{aligned} r_t^{(i)} &\sim p(r_t|r_{0:t-1}^{(i)}) \quad (13)\\ s_{t,1:N_s}^{(i)} &\sim \rho_t(s_{1:N_s}) \quad (14)\\ w_t^{(i)} &\propto w_{t-1}^{(i)} p(\mathbf{y}_t|r_t^{(i)},s_{t,1:N_s}^{(i)})p(\mathbf{Z}_t|s_{t,1:N_s}^{(i)}). \quad (15)\end{aligned}$$

Moreover, (9) shows that, in order to start the recursion, we need to draw initial populations not only from the prior $p(r_0)$, but also from the posterior $p(s_{1:N_s}|\mathbf{Z}_0,\mathbf{B})$.

As a consequence of the previous analysis, we propose an algorithm consisting of three stages:

1. **Initialization:** We use an optimization method to compute point estimates of the unknown DFC locations, $\hat{c}_{2:N_c}$, draw directly from $p(r_0)$ and use a population Monte Carlo (PMC) method [5] to draw from $p(s_{1:N_s}|\mathbf{Z}_0,\mathbf{B})$.
2. **Tracking:** Find an adequate approximation of $\rho_t(s_{1:N_s})$ and use it to run the MKF algorithm of (13), (14) and (15).
3. **Fusion:** Use the resampling with non-proportional allocation (RNA) method of [10] to distribute the MKF tracking algorithm over the N_c DFCs. In this way, resampling operations are carried out locally at each DFC and global estimates (of r_t, v_t and $s_{1:N_s}$) can be computed by linear combination (fusion) of local estimates.

3.2.1 Initialization

Drawing $r_0^{(i)} \sim p(r_0)$, $i=1,\ldots,M$, is straightforward. In order to find initial point-estimates of $s_{1:N_s}$ and $c_{2:N_c}$, we propose to apply an accelerated random search (ARS) algorithm [4] but, because of the high dimension of the unknowns, it is convenient in practice to address the estimation of $c_{2:N_c}$, on one hand, and each s_n, $n \in \{1,\ldots,N_s\}$, separately. In particular, we propose to compute

$$\begin{aligned} \hat{c}_{2:N_c} &= \arg\max_{c_{2:N_c}} \{p(\mathscr{Q}|c_{2:N_c})p(c_{2:N_c})\} \quad (16)\\ &= \arg\max_{c_{2:N_c}} \left\{ \prod_{i\neq k} p(q_{i,k}|c_i,c_k) \prod_{j=2}^{N_c} CN(c_j|\mu_j^c,\sigma_c^2) \right\} \quad (17)\\ \hat{s}_\ell &= \arg\max_{s_\ell} \left\{ \prod_{n=1}^{N_c} p(z_{\ell,n,0}|s_\ell,\hat{c}_n) \right\}, \quad (18)\end{aligned}$$

for $\ell = 1,\ldots,N_s$ and $\hat{c}_1 = c_1 = 0$. The general form of the ARS technique is outlined in Table 1.

Problem: $\hat{\alpha} = \arg\max_{\alpha\in\mathscr{A}} g(\alpha)$ for some function g.
Denote: $R_{min} > 0$, the "minimum radius"; $R_{max} > R_{min}$, the "maximum radius"; $R_{max} \geq R_n \geq R_{min}$ the radius at the n-th iteration; $\nu > 1$ the "contraction" factor; α_n the solution obtained after the n-th iteration; and
$\mathscr{B}_n = \{\tilde{\alpha} \in \mathscr{A} : ||\tilde{\alpha}-\alpha_n||_2 < R_n\}$,
where $||\cdot||_2$ indicates 2-norm.
Algorithm: given R_n and α_n,
(1) Draw $\tilde{\alpha} \sim U(\mathscr{B}_n)$.
(2) If $g(\tilde{\alpha}) > g(\alpha_n)$ then $\alpha_{n+1} = \tilde{\alpha}$ and $R_{n+1} = R_{max}$, else $\alpha_{n+1} = \alpha_n$ and $R_{n+1} = R_n/\nu$.
(3) If $R_{n+1} < R_{min}$, then $R_{n+1} = R_{max}$.
(4) Go back to (1)

Table 1: Iterative ARS algorithm for a maximization problem. Parameter α is possibly multidimensional (typically, $\alpha \in \mathbb{C}^n$). The algorithm is usually stopped after a given number of iterations without going changing α_n.

In order to draw $s_{0,1:N_s}^{(i)} \sim p(s_{1:N_s}|\mathbf{Z}_0,\mathbf{B})$, we use an iterated Monte Carlo method called PMC [5]. In the first iteration, particles are drawn from independent complex Gaussian proposals built from the ARS estimates and a fixed variance, $\sigma_s^2(0)$, i.e.,

$$s_n^{(i)}(0) \sim CN(s_n|\hat{s}_n,\sigma_s^2(0)), \quad n=1,\ldots,N_s, \quad i=1,\ldots,M, \quad (19)$$

with weights

$$w^{(i)}(0) = \frac{p(\mathbf{Z}_0|s_{1:N_s}^{(i)}(0))p(\mathbf{B}|s_{1:N_s}^{(i)}(0))}{\prod_{n=1}^{N_s} CN(s_n^{(i)}(0)|\hat{s}_n,\sigma_s^2(0))}. \quad (20)$$

After the $(k-1)$th iteration, the weighted particles are $\Omega_{k-1}^{pmc} = \{s_{1:N_s}(k-1)^{(i)}, w^{(i)}(k-1)\}_{i=1}^M$ and importance sampling for the kth iteration is performed as

$$s_n^{(i)}(k) \sim CN(s_n|\overline{s}_n^{(i)}(k-1),\sigma_{s,n}^2(k-1)), \quad (21)$$

where $\overline{s}_n(k-1) = \sum_{i=1}^M w^{(i)}(k-1)s_n^{(i)}(k-1)$, $\overline{s}_n^{(i)}(k-1) = as_n^{(i)}(k-1) + (1-a)\overline{s}_n(k-1)$ and $\sigma_{s,n}^2(k-1) = (1-a^2)\sum_{i=1}^M w^{(i)}(k-1)|s_n^{(i)}(k-1)-\overline{s}_n(k-1)|^2$ for some $0<a<1$, i.e., we build the $(k-1)$th kernel approximation of $p(s_{1:N_s}|\mathbf{Z}_0,\mathbf{B})$ with 'shrinkage' [14] for variance reduction. The corresponding weights are

$$w^{(i)}(k) = \frac{p(\mathbf{Z}_0|s_{1:N_s}^{(i)}(k))p(\mathbf{B}|s_{1:N_s}^{(i)}(k))}{\prod_{n=1}^{N_s} CN(s_n^{(i)}(k)|\overline{s}_n^{(i)}(k-1),\sigma_{s,n}^2(k-1))}. \quad (22)$$

If the algorithm is iterated N times, we obtain a sample of equally-weighted particles $\{s^{(i)}_{0,1:N_s}\}_{i=1}^{M}$, with approximate pdf $p(s_{1:N_s}|\mathbf{Z}_0,\mathbf{B})$ by resampling from Ω_N^{pmc}.

3.2.2 Tracking

Given the initial sample $\Omega_0^{mkf} = \{r_0^{(i)}, s_{0,1:N_s}^{(i)}, w_0^{(i)} = \frac{1}{M}\}_{i=1}^{M}$, the MKF algorithm (13)-(14) can be applied if we specify how to approximate $\rho_t(s_{1:N_s})$. One of the simplest choices is to assume a Gaussian distribution built from the particles and weights at time $t-1$, i.e.,

$$\rho_t(s_{1:N_s}) \approx \prod_{n=1}^{N_s} CN(s_n|\overline{s}_{t-1,n}, \sigma^2_{t-1,s,n}), \tag{23}$$

where $\overline{s}_{t-1,n} = \sum_{i=1}^{M} w_{t-1}^{(i)} s_{t-1,n}^{(i)}$ and $\sigma^2_{t-1,s,n} = \sum_{i=1}^{M} w_{t-1}^{(i)} |s_{t-1,n}^{(i)} - \overline{s}_{t-1,n}|^2$. This approximation is easy to sample and still provides an acceptable performance, as will be numerically shown in Section 4.

3.2.3 Fusion

Data fusion can be naturally integrated into the particle filter by using the distributed RNA scheme [10]. Let us split the overall particle set Ω_t^{mkf} into N_c subsets, one per DFC, denoted as $\Omega_{n,t}^{mkf} = \{r_{0:t}^{(n,i)}, s_{t,1:N_s}^{(n,i)}, w_t^{(n,i)}, W_t^{(n)^*}\}_{i=1}^{M_n}$, $n = 1,\ldots,N_c$, and such that $\sum_n M_n = M$. The weights in $\Omega_{n,t}^{mkf}$ are normalized locally, i.e., $\sum_{i=1}^{M_n} w_t^{(n,i)} = 1$, and the sum of the unnormalized weights, $W_t^{(n)^*} = \sum_{i=1}^{M_n} w_t^{(n,i)^*}$ is also kept in order to assess the relative value of each subset (the subsets $\Omega_{n,t}^{M_n}$ are not equally *good* in general).

In the basic RNA scheme, an independent MKF algorithm (13)-(14) is run for each subset $\Omega_{n,t}^{mkf}$ (i.e., for each DFC). This means that resampling is carried out locally (using any desired method) at each DFC and estimates are also computed locally,

$$r_t^{n,mmse} = \sum_{i=1}^{M_n} w_t^{(n,i)} r_t^{(n,i)}, \quad v_t^{n,mmse} = \sum_{i=1}^{M_n} w_t^{(n,i)} \mu_t^{v^{(n,i)}} \tag{24}$$

and

$$s_{1:N_s}^{n,mmse} = \sum_{i=1}^{M_n} w_t^{(n,i)} s_{t,1:N_s}^{(n,i)}. \tag{25}$$

Optimal fusion is performed by combining the local estimates according to the sum-weights, $W_t^{(n)^*}$, i.e., global MMSE estimates are computed as

$$r_t^{mmse} = \frac{\sum_{n=1}^{N_c} W_t^{(n)^*} r_t^{n,mmse}}{\sum_{k=1}^{N_c} W_t^{(k)^*}}, \quad v_t^{n,mmse} = \frac{\sum_{n=1}^{N_c} W_t^{(n)^*} v_t^{n,mmse}}{\sum_{k=1}^{N_c} W_t^{(k)^*}} \tag{26}$$

and

$$s_{1:N_s}^{mmse} = \frac{\sum_{n=1}^{N_c} W_t^{(n)^*} s_{1:N_s}^{n,mmse}}{\sum_{k=1}^{N_c} W_t^{(k)^*}}, \tag{27}$$

in such a way that only the local estimates and the sum-weights need to be transmitted.

One limitation of this approach is that when the subset sizes, M_n, $n = 1,\ldots,N_c$, are not large enough, some particle filters may get relatively impoverished [10], i.e., it may eventually happen that, for some n, $W_t^{(n)^*} << W_t^{(k)^*}$, for all $k \neq n$. In such a case, the corresponding nth DFC becomes "useless", since its local estimates are essentially irrelevant for the computation of the global estimates. A solution to this phenomenon (equivalent to the weight degeneracy in standard particle filters [6]) is to periodically perform a local exchange (LE) of a small number of particles between pairs of DFCs. We propose a simple implementation of LE in which $L < \min_n\{M_n\}$ particles from DFC n are transmitted to DFC $n+1$, for $n = 1,\ldots,N_c-1$ and L particles from DFC N_c are transmitted to DFC 1, i.e., particles are exchanged in a ring configuration.

4. SIMULATIONS

In order to provide illustrative numerical results, we have particularized the model of Section 2 to a network of power-aware sensors. Specifically, the measurement functions $f_s(\cdot,\cdot)$ has the form

$$f_s(d,\varepsilon) = 10\log_{10}\left(\frac{1}{d^2} + \eta\right) + \varepsilon, \quad \text{(dB)} \tag{28}$$

where $\eta = 10^{-4}$ accounts for the sensitivity of the measurement device (-40 dB). The n-th sensor transmits its measurement, $y_{n,t}$, only if it corresponds to a distance $d_{n,t} < S_y = 48.86$ m (i.e., $y_{n,t} > -33.78$ dB) and otherwise remains silent. A transmission failure can also occur, with probability $\beta = 10^{-3}$. The observational noise, ε, is zero mean Gaussian but, depending on whether the power observation is carried out at a sensor node or at a DFC node, its variance is assumed different. In particular $\varepsilon_{n,t}^{y} \sim N(0,2)$, for sensors, and, for DFCs, $\varepsilon_{i,n,t}^{z}$ and $\varepsilon_{i,n}^{0}$ are identically distributed according to the Gaussian pdf $N(0,10^{-2})$. Therefore, the likelihoods, namely,

$$p(\mathbf{y}_t|r_t^{(m)}, s_{1:N_s}^{(m)}) = \prod_{l=1}^{N_t} N(y_{\kappa(l),t}|r_t^{(m)}, s_{\kappa(l)}^{(m)}) \tag{29}$$

$$p(\mathbf{Z}_t|s_{1:N_s}^{(m)}, c_1, \hat{c}_{2:N_c}) = \prod_{i=1,n=1}^{N_t,N_c} p(z_{\kappa(i),n,t}|s_{\kappa(i)}^{(m)}, \hat{c}_n) \tag{30}$$

are Gaussian with known mean and variance.

At time zero, the sensors detect all other nodes which are closer than $S_u = 48.86$ m. Since observations are obtained from function $f_s(\cdot,\cdot)$, with the parameters already described for the sensors, the probability of detection is

$$p_d(d_{n,k}^s, S_u) = \Phi_N\left(\frac{\overline{P}_{n,k} - P_u}{\sqrt{10^{-2}}}\right), \tag{31}$$

where $\overline{P}_{n,k} = f_s(d_{n,k}^s, 0)$, $P_u = f_s(S_u, 0)$ and 10^{-2} is the variance of the observational noise.

The state priors are $p(r_0) = CN(r_0|0,10)$ and $p(v_0|0,0.1)$ and the state equation parameters are $T = \frac{1}{2}$ s and $\sigma_u^2 = \frac{1}{5}$. There are $N_c = 4$ DFCs and $N_s = 23$ sensors in the network. We assume $c_1 = 0$, while the others have complex Gaussian priors with equal variance $\sigma_c^2 = 25$ and means $-50 + j35$, $45 - j37$ and $36 + j45$ (where $j = \sqrt{-1}$), respectively. This prior pdf's are used to randomly draw initial estimates of $c_{2:4}$ which are used as inputs to the ARS algorithm that solves (17), the other parameters being $\nu = 2$, $R_{max} = 15$, $R_{min} = 10^{-4}$. The ARS algorithm for problem (18) receives as inputs a sensor position drawn from $U(\mathscr{C})$ (where $\mathscr{C}$ is the square centered at 0 with sides of length 200 m), $R_{max} = 200$, $r_{min} = 10^{-4}$ and $\nu = 2$. The ARS procedures are iterated 3000 times for (17) and 1000 times for each (18). The estimates $\hat{s}_{1:N_s}$ are then used to build the first proposal pdf in the PMC procedure. The corresponding variance is $\sigma_s^2(0) = \frac{1}{2}$ and the subsequent proposals are computed by shrinkage, with parameter $a = 0.7$. We iterate the PMC algorithm 15 times with 3000 particles.

Resampling, via the RNA scheme, is performed every 5 time steps of the tracking algorithms. The latter are run with $M = 3000$ particles and each DFC is assigned $M_n = M/N_c = 750$ particles (for $n = 1,2,3,4$). We assume a local exchange of particles every 4 resampling steps, with $L = 8$ particles being transmitted from DFC n to DFC $n+1$ and from DFC N_c to DFC 1.

Figure 1 shows the results of a typical simulation run with 245 discrete-time steps (122.5 s of simulate time) using: (a) the MKF algorithm with known node locations (both DFCs and sensors) and using the RNA scheme for its distributed implementation over the 4 DFCs, which is labeled 'DMKF'; and (b) the MKF algorithm described in Section 3.2, that jointly estimates the target state and the

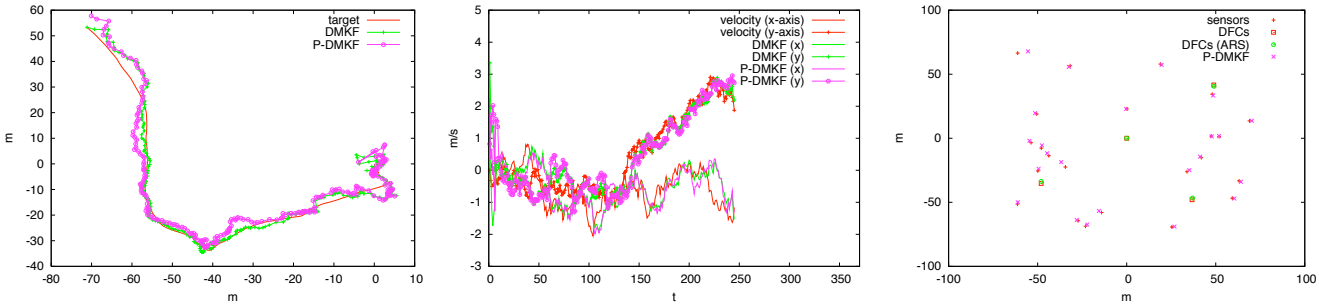

Figure 1: Example of results obtained with the DMKF and P-DMKF algorithms for the estimation of the target trajectory and velocity. P-DMKF also estimates the node locations (both sensors and DFCs). **Left:** Estimate of the target trajectory during 122.5 s (245 discrete-time steps). **Middle:** Estimate of the target velocity on the real (x) and imaginary (y) axis. **Right:** Estimates of the node positions.

node locations, using a PMC procedure in the initialization stage, which we label as 'P-DMKF'. In the left plot, it is observed that both algorithms are able to closely track a highly nonlinear trajectory. The velocity estimates, both in the real and imaginary axis, are shown in the center plot, with similarly accurate performance. Finally, the right plot shows the true node locations and the estimated positions. It is seen that the combination of the ARS, PMC and distributed MKF methods provide reasonably accurate estimates of the sensors and DFCs locations.

In order to estimate the average performance of the proposed method, we have carried out 20 independent computer simulations (each one with a different, and random, network deployment and target trajectory) and computed the mean absolute error (MAE) in the estimation of the target position, r_t, and its velocity, v_t. The results are presented in Table 2 and quantitatively illustrate the effectiveness of the method.

	r_t	v_t
DMKF	1.5067 m	0.5718 m/s
P-DMKF	2.6411 m	0.6245 m/s

Table 2: Mean absolute error (MAE) in the estimation of the target position, r_t, given in m; the target velocity, v_t, in m/s, using the DMKF and P-DMKF algorithms.

5. CONCLUSIONS

We have proposed a novel SMC algorithm to jointly estimate the positions of the nodes of a WSN (including both the sensors and the DFCs) and track a target that moves along the region monitored by the network. The proposed method does not require the aid of beacons in order to locate the network nodes. Instead, it resorts to a novel combination of Monte Carlo optimization and iterated sampling procedures in order to generate an initial population of node locations with sufficient quality. Starting from this population, an MKF algorithm is subsequently used to recursively track the target *and* sequentially generate new samples of node positions as new data become available. Moreover, we have proposed a distributed implementation of the tracker using the RNA technique. The latter makes it possible to split the set of particles among the network DFCs, let each individual DFC propagate-and-resample its subset of particles locally (i.e., run an independent particle filtering algorithm) and only exchange a limited amount of data to produce fused (global) estimates of the desired magnitudes. We have presented computer simulation results that illustrate the effectiveness of the proposed method with a network of power-aware sensors.

6. ACKNOWLEDGEMENTS

This work has been supported by *Ministerio de Educación y Ciencia* of Spain (MONIN, ref. TEC-2006-13514-C02-01/TCM), *Comunidad de Madrid* (PROMULTIDIS-CM, ref. S-0505/TIC/0233) and *Ministerio de Industria, Turismo y Comercio* of Spain (m:Ciudad, ref. FIT-330503-2006-2, *Plan Avanz@*).

REFERENCES

[1] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero III, R. L. Moses, and N. S. Correal, "Locating the nodes," *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 54–69, July 2005.

[2] G. Sun, J. Chen, W. Guo, and K. J. R. Liu, "Signal processing techniques in network-aided positioning," *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 12–23, July 2005.

[3] M. Vemula, M.F. Bugallo, and P.M. Djurić, "Fusion of information for sensor self-localization by a Monte Carlo method," in *Proceedings of ICIF*, July 2006.

[4] M. J. Appel, R. Labarre, and D. Radulovic, "On accelerated random search," *SIAM Journal of Optimization*, vol. 14, no. 3, pp. 708–730, 2003.

[5] O. Cappé, A. Gullin, J. M. Marin, and C. P. Robert, "Population monte carlo," *Journal of Computational and Graphical Statistics*, vol. 13, no. 4, pp. 907–929, 2004.

[6] A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo Sampling methods for Bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, 2000.

[7] D. Crisan and A. Doucet, "A survey of convergence results on particle filtering," *IEEE Transactions Signal Processing*, vol. 50, no. 3, pp. 736–746, March 2002.

[8] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation and tracking," *IEEE Transactions Signal Processing*, vol. 50, no. 2, pp. 425–437, February 2002.

[9] P. M. Djurić, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Míguez, "Particle filtering," *IEEE Signal Processing Magazine*, vol. 20, no. 5, pp. 19–38, September 2003.

[10] M. Bolić, P. M. Djurić, and S. Hong, "Resampling algorithms and architectures for distributed particle filters," *IEEE Transactions Signal Processing*, vol. 53, no. 7, pp. 2442–2450, July 2005.

[11] R. Chen and J. S. Liu, "Mixture Kalman filters," *Journal of the Royal Statistics Society B*, vol. 62, pp. 493–508, 2000.

[12] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960.

[13] S. Haykin, *Adaptive Filter Theory, 4th Edition*, Prentice Hall, Information and System Sciences Series, 2001.

[14] J. Liu and M. West, "Combined parameter and state estimation in simulation-based filtering," in *Sequential Monte Carlo Methods in Practice*, A. Doucet, N. de Freitas, and N. Gordon, Eds., chapter 10, pp. 197–223. Springer, 2001.