Hindawi Publishing Corporation
EURASIP Journal on Applied Signal Processing
Volume 2006, Article ID 83042, Pages 1–16
DOI 10.1155/ASP/2006/83042

Particle Filtering Algorithms for Tracking a Maneuvering Target Using a Network of Wireless Dynamic Sensors

Joaquín Míguez and Antonio Artés-Rodríguez

Departamento de Teoría de la Señal y Comunicaciones, Universidad Carlos III de Madrid, Avenida de la Universidad 30, Leganés, 28911 Madrid, Spain

Received 16 June 2005; Revised 24 January 2006; Accepted 30 April 2006

We investigate the problem of tracking a maneuvering target using a wireless sensor network. We assume that the sensors are binary (they transmit '1' for target detection and '0' for target absence) and capable of motion, in order to enable the tracking of targets that move over large regions. The sensor velocity is governed by the tracker, but subject to random perturbations that make the actual sensor locations uncertain. The binary local decisions are transmitted over the network to a fusion center that recursively integrates them in order to sequentially produce estimates of the target position, its velocity, and the sensor locations. We investigate the application of particle filtering techniques (namely, sequential importance sampling, auxiliary particle filtering and cost-reference particle filtering) in order to efficiently perform data fusion, and propose new sampling schemes tailored to the problem under study. The validity of the resulting algorithms is illustrated by means of computer simulations.

Copyright © 2006 J. Míguez and A. Artés-Rodríguez. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. INTRODUCTION

Recently, there has been a surge of interest in the application of networks of wireless microsensors in diverse areas, including manufacturing, health and medicine, transportation, environmental monitoring, scientific instrumentation and others [1–3]. One common feature to these applications is that they involve the detection, classification, and tracking of signals, with the outstanding peculiarity that the large amounts of (possibly multimodal) data acquired by the sensors must be handled and integrated in order to perform the prescribed tasks [1]. Wireless sensor networks (WSN) are usually depicted as a collection of data-acquiring devices (sensors) and one or more fusion centers which are in charge of integrating the data to extract the information of interest.

This paper deals with the problem of tracking a moving target over a region which is monitored by WSN [4]. As well as in most WSN applications, the main constraints are related to the network cost of deployment and operation. It is desirable that the sensors be inexpensive and, as a consequence, devices with limited processing capabilities are commonly used [4]. Moreover, stringent energy consumption restrictions must be met for the continued and reliable operation of networks consisting of battery-supported sensors. In this respect, radio communication is a major power-consumer [5], so it should be kept to a minimum. However, this implies that only a small fraction of the data collected by the sensors can be transmitted to the fusion center, which results in a decrease of the tracking capability. One solution is to deploy dense networks (i.e., networks with a large number of sensors per unit area/volume) in order to boost performance but, with this approach, it may turn out prohibitive to provide adequate coverage for large regions. Another strong requirement which is peculiar to target tracking and target localization [6, 7] applications is the need to accurately estimate the position of the sensors in the network [8], a task which is often included in network calibration [9]. When the number of sensors is large, the accurate estimation of their locations becomes hard.

Bearing in mind the above considerations, we investigate the tracking of a maneuvering target on a 2-dimensional space using a WSN of binary, distance-aware sensors [10–12], that is, we consider sensing devices that can measure some distance-related physical magnitude (e.g., received signal strength) and use it to make a binary decision regarding the presence of the target within a certain range. The resulting bit ("1" if the target is detected within the sensor range, "0" otherwise) is transmitted to the fusion center, where the local decisions are integrated to

recursively estimate the current position and velocity of the target.

Since the target can move over a large region and it may not be possible to deploy such a widespread network with the required sensor density for adequate performance, we propose to use a relatively small number of dynamic sensors. Specifically, we assume that the sensors are deployed randomly (there is uncertainty in the knowledge of their initial positions) and the network is endowed with a control system that allows the tracker to command the sensors to move with a certain speed (including magnitude and phase). The speed of motion assigned to each sensor is determined by the tracker using the target trajectory estimates provided by the fusion center, but the actual sensor movement is also subject to random perturbations. We will discuss several strategies for sensor speed assignment that differ in complexity and performance.

The novel target tracking algorithms that we introduce in this paper are largely based on the sequential Monte Carlo (SMC) methodology, also known as *particle filtering* (PF) [13–16]. PF algorithms are tools for estimating the time-varying state of a dynamic system that cannot be observed directly, but through some related measurements. The a posteriori probability distribution of the state given the measurements is approximated, with any desired accuracy [17], using a probability mass function (pmf) with random support. This pmf is composed of samples in the state-space and associated weights, which can be intuitively seen as approximations of the a posteriori probability of the samples. The PF approach lends itself naturally to the problem at hand, where the unobserved dynamic state consists of the target position and speed together with the sensor locations, and the measurements are the local sensor decisions and the power of the communication signals transmitted by the sensors and received at the fusion center. From this point of view, a PF algorithm performs a data fusion process that consists of approximating the a posteriori distribution of the target and sensor trajectories, as well as any estimators that can be derived from it, given the measurements. We investigate the application of different families of PF techniques (in particular, sequential importance sampling, auxiliary particle filtering and cost-reference particle filtering) and propose new sampling schemes tailored to the problem of joint sensor and target tracking.

The remaining of this paper is organized as follows. Section 2 introduces some background material on PF. The problem of tracking a maneuvering target using dynamic binary sensors is formally stated in Section 3 and a system model suitable for application of the PF methods is derived. Four target-tracking PF algorithms are introduced in Section 4 and, since these procedures are based on the network capability to govern the speed of motion of the sensors, different velocity assignment strategies are discussed in Section 5. Illustrative computer simulation results are presented in Section 6 and, finally, Section 7 is devoted to a brief discussion and concluding remarks.

2. BACKGROUND: BAYESIAN TRACKING AND PARTICLE FILTERING

Let us consider the dynamic system in state-space form

$$x_t = f_x(x_{t-1}, u_t), \quad t = 1, 2, \ldots \text{ (state equation)}, \tag{1}$$

$$y_t = f_y(x_t, \phi, n_t), \quad t = 1, 2, \ldots \text{ (observation equation)}, \tag{2}$$

where x_t is the system state, y_t is the corresponding observation, f_x and f_y are (possibly nonlinear) functions, ϕ denotes the set of unknown fixed parameters, and u_t and n_t are the (possibly non-Gaussian) system noise and observation noise processes, respectively. For the sake of deriving estimation algorithms for x_t, some basic probabilistic assumptions are usually made on (1)–(2). These include the pair-wise statistical independence of x_t, u_t, and n_t, the availability of a known prior probability density function (pdf) for the initial state, $p(x_0)$, and the fixed parameters, $p(\phi)$, and the ability to sample the transition pdf $p(x_t \mid x_{t-1})$ and to evaluate the likelihood function $p(y_t \mid x_{0:t}, y_{1:t-1})$ (which reduces to $p(y_t \mid x_t)$ when all fixed parameters are known, i.e., ϕ is empty).

2.1. Sequential importance sampling

Many problems can be stated as the estimation of the sequence of states $x_{0:t} = \{x_0, \ldots, x_t\}$ given the series of observations $y_{1:t} = \{y_1, \ldots, y_t\}$. From the Bayesian point of view, all relevant information is contained in the smoothing pdf, $p(x_{0:t} \mid y_{1:t})$, but it is in general impossible to find it (or its moments) in a closed form. The sequential importance sampling (SIS) algorithm is a recursive Monte Carlo technique to approximate $p(x_{0:t} \mid y_{1:t})$ by means of a pmf with random support [13, 14, 18]. The algorithm is derived from the recursive decomposition

$$p(x_{0:t} \mid y_{1:t}) \propto p(y_t \mid x_{0:t}, y_{1:t-1})\, p(x_t \mid x_{t-1})\, p(x_{0:t-1} \mid y_{1:t-1}) \tag{3}$$

and the importance sampling (IS) principle [18]. In particular, we are interested in approximating $p(x_{0:t} \mid y_{1:t})$ from its samples, but it is obviously impossible to draw from the latter pdf directly. Instead, we choose an importance pdf (also termed *proposal* or *trial* density) $q(x_{0:t} \mid y_{1:t})$, with a domain that includes that of $p(x_{0:t} \mid y_{1:t})$, and draw M samples $x_{0:t}^{(i)} \sim q(x_{0:t} \mid y_{1:t})$, $i = 1, \ldots, M$. Each sample is weighted as $\tilde{w}_t^{(i)} = p(x_{0:t}^{(i)} \mid y_{1:t}) / q(x_{0:t}^{(i)} \mid y_{1:t})$ and the pair $(x_{0:t}^{(i)}, \tilde{w}_t^{(i)})$ is called a particle.

However, the computational cost of drawing particles in this way grows with time, which is unacceptable for online applications [18]. To avoid this limitation, we assume the importance function to be factored as $q(x_{0:t} \mid y_{1:t}) \propto q(x_t \mid x_{0:t-1}, y_{1:t}) q(x_{0:t-1} \mid y_{1:t-1})$ and, using (3), it turns out that both sampling and weighting of the particles can be carried out recursively. To be specific, given the set $\Omega_{t-1} \triangleq \{x_{0:t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{M}$, at time t we proceed to draw $x_t^{(i)}$ and update

$w_{t-1}^{(i)}$ as

$$x_t^{(i)} \sim q\left(x_t \mid x_{0:t-1}^{(i)}, y_{1:t}\right), \tag{4}$$

$$\widetilde{w}_t^{(i)} = \frac{p\left(y_t \mid x_{0:t}^{(i)}, y_{1:t-1}\right) p\left(x_t^{(i)} \mid x_{t-1}^{(i)}\right)}{q\left(x_t^{(i)} \mid x_{0:t-1}^{(i)}, y_{1:t}\right)} w_{t-1}^{(i)}, \tag{5}$$

$$w_t^{(i)} = \frac{\widetilde{w}_t^{(i)}}{\sum_{k=1}^{M} \widetilde{w}_t^{(k)}}. \tag{6}$$

The weight normalization step in (6) is performed to ensure that the collection of weights yields a proper pmf for finite M. The new set $\Omega_t = \{x_{0:t}^{(i)}, w_t^{(i)}\}_{i=1}^{M}$ readily yields a discrete approximation of the smoothing pdf,

$$p(x_{0:t} \mid y_{1:t}) \approx \widehat{p}(x_{0:t} \mid y_{1:t}) = \sum_{i=1}^{M} w_t^{(i)} \delta\left(x_{0:t} - x_{0:t}^{(i)}\right), \tag{7}$$

where δ is the Dirac delta function, and it can be shown that $\widehat{p}(x_{0:t} \mid y_{1:t}) \to p(x_{0:t} \mid y_{1:t})$, as $M \to \infty$ [17]. Any desired estimators can be easily approximated from $\widehat{p}(x_{0:t} \mid y_{1:t})$, for example, the minimum mean square error (MMSE) estimate of $x_{0:t}$ is $x_{0:t}^{\text{mmse}} = \sum_{i=1}^{M} w_t^{(i)} x_{0:t}^{(i)}$. Equations (4)–(6) constitute the sequential importance sampling (SIS) algorithm.

One major practical limitation of the SIS method is that the variance of the weights increases stochastically over time until one single weight accumulates the unit probability mass, rendering the approximation $\widehat{p}(x_{0:t} \mid y_{1:t})$ useless [14]. This is known as "degeneracy of weights," and it is usually avoided by resampling the particles [14, 18]. Intuitively, resampling amounts to stochastically discarding particles with small weights while replicating those with larger weights. Although several schemes have been proposed [13, 14, 19], we will restrict ourselves to the conceptually simple multinomial resampling algorithm [17, 19], which reduces to drawing M times from the discrete pmf $\widehat{p}(x_{0:t} \mid y_{1:t})$.

2.2. Auxiliary particle filtering

One appealing version of the SIS approach is obtained when we integrate the importance sampling and resampling steps using the auxiliary particle filtering (APF) technique [20]. In particular, we define

$$q\left(x_t, k \mid x_{0:t-1}^{(k)}, y_{1:t}\right) \propto \rho_t^{(k)} p\left(x_t \mid x_{t-1}^{(k)}\right) w_{t-1}^{(k)}, \tag{8}$$

where k is an "auxiliary index," $\rho_t^{(k)} = p(y_t \mid \widehat{x}_t^{(k)}, x_{0:t-1}^{(k)})$, and $\widehat{x}_k$ is an estimate of x_k given $x_{t-1}^{(k)}$ (typically the mean of $p(x_t \mid x_{t-1}^{(k)})$). The sampling step is then carried out in two stages,

$$\begin{aligned} k^{(i)} &\sim q_t(k), \quad \text{where } q_t\,(k=j) \propto w_{t-1}^{(j)} \rho_t^{(j)}, \\ x_t^{(i)} &\sim p\left(x_t \mid x_{t-1}^{(k^{(i)})}\right), \quad i = 1, \ldots, M, \end{aligned} \tag{9}$$

that is, we randomly select the indices of the particles with a higher predicted likelihood (according to the estimates $\widehat{x}_t^{(i)}$) and then propagate the selected particles using the transition pdf. The new samples are joined with the former ones according to the auxiliary indices, $x_{0:t}^{(i)} = \{x_{0:t-1}^{(k^{(i)})}, x_t^{(i)}\}$, and the weights are updated as

$$w_t^{(i)} \propto \frac{p\left(y_t \mid x_{0:t}^{(i)}, y_{1:t-1}\right)}{\rho_{t-1}^{(k^{(i)})}}. \tag{10}$$

Note that resampling is implicit in (9). Therefore, it is carried out at each time step, but this scheme has the advantage of exploiting the knowledge of the new observation, y_t, through the predictive likelihood $\rho_t^{(j)}$.

2.3. Cost-reference particle filtering

When a reliable and tractable probabilistic model of system (1)–(2) is not available, the PF approach can still be exploited using the cost-reference particle filtering (CRPF) methodology [21]. Instead of trying to approximate a posterior pdf, let us aim at computing the sequence of states that attains the minimization of an arbitrary cost function with additive structure,

$$C(x_{0:t}, y_{1:t}) = \lambda C(x_{0:t-1}, y_{1:t-1}) + \triangle C(x_t, y_t), \tag{11}$$

where $0 < \lambda < 1$ is a forgetting factor and $\triangle C$ is an incremental cost function. If a risk function is also defined that consists of the cost of a predicted state, that is, $R(x_{t-1}, y_t) = \lambda C(x_{0:t-1}) + \triangle C(f_x(x_{t-1}, 0))$ (zero-mean system noise is assumed), then we can describe the basic steps of a CRPF algorithm.

(1) *Initialization*: draw $\{x_0^{(i)}, C_0^{(i)} = 0\}_{i=1}^{M}$ from an arbitrary pdf. The only constraint is that the particles should not be infinitely far away from the initial state. Notation $C_0^{(i)}$ represents the cost of the ith particle in the absence of observations, that is, $C_0^{(i)} = C(x_0, y_{1:0})$.

(2) *Selection*: given $\Omega_{t-1} = \{x_{t-1}^{(i)}, C_{t-1}^{(i)}\}_{i=1}^{M}$ and the new observation y_t, compute the risks $R_t^{(i)} = R(x_{t-1}^{(i)}, y_t)$ and resample the particles according to the probabilities

$$p\left(x_{t-1}^{(i)}\right) \propto \mu\left(R_t^{(i)}\right), \tag{12}$$

where μ is a nonnegative and monotonically nonincreasing function, termed the generating function. The resampled particles preserve their original costs and yield $\widehat{\Omega}_{t-1} = \{\widehat{x}_{t-1}^{(i)}, \widehat{C}_{t-1}^{(i)}\}_{i=1}^{M}$.

(3) *Propagation*: generate new particles by drawing $x_t^{(i)} \sim p_t(x_t \mid x_{t-1}, y_t)$ from an arbitrary propagation pdf p_t. Assign new weights, $C_t^{(i)} = \lambda \hat{C}_{t-1}^{(i)} + \triangle C(x_t^{(i)}, y_t)$, to build $\Omega_t = \{x_t^{(i)}, C_t^{(i)}\}_{i=1}^{M}$.

Estimation of the state can be performed at any time, either by choosing the minimum cost particle or by assigning probabilities to the particles of the form $\pi_t^{(i)} \propto \mu(C_t^{(i)})$ and then computing the mean cost estimate [21] $\hat{x}_t^{\text{mean}} = \sum_{i=1}^{M} \pi_t^{(i)} x_t^{(i)}$.

3. SYSTEM MODEL

We address the problem of tracking a maneuvering target that moves along a 2-dimensional space using a WSN. It is assumed that each sensor can measure some physical magnitude related to the distance between the target and the sensor itself, and then uses the measurement to determine whether the target is within a predefined range or not (we will henceforth refer to the sensors as "binary"). The distinctive features of the system under study are the following: (a) the sensor locations are not exactly known, so they must be estimated together with the target trajectory, and (b) the sensors are dynamic, that is, they are able of motion with adjustable speed (magnitude and phase). With this setup, it is possible to use a relatively small network to follow a target over a large area by making the sensors move according to the estimated track.

3.1. Signal model

We can describe the tracking problem using a dynamic state-space model formulation similar to (1)–(2). Let $r(\tau)$ and $v(\tau)$ be the continuous-time complex random processes of the target position and velocity, respectively. If measurements are collected by the network every T_s seconds (s), then it is straightforward to derive the discrete-time linear kinematic model [22]

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{Q}\mathbf{u}_t, \tag{13}$$

where $\mathbf{x}_t = [r_t, v_t] \in \mathbb{C}^2$ is the target state at discrete time t, $r_t = r\,(\tau = tT_s)$ and $v_t = v\,(\tau = tT_s)$ are samples of the position and speed random processes, respectively,

$$\mathbf{A} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \tag{14}$$

is the transition matrix, $\mathbf{u}_t \in \mathbb{C}^2$ is a 2-dimensional complex Gaussian process with zero mean and covariance matrix $\sigma_u^2 \mathbf{I}_2$ ($\mathbf{I}_2$ is the 2×2 identity matrix), which we denote as $\mathbf{u}_t \sim N(\mathbf{u}_t \mid \mathbf{0}, \sigma_u^2 \mathbf{I}_2)$, and

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{2} T_s^2 & 0 \\ 0 & T_s \end{bmatrix}. \tag{15}$$

The target probabilistic description is completed with a known a priori pdf for the initial target location and speed, $\mathbf{x}_0 \sim p(\mathbf{x}_0)$, that models the uncertainty on the target state before any observations are collected.

The system state includes the target position and velocity, as well as the sensors positions, and we also assume a linear motion model for the latter. Specifically, there are N_s sensors in the network and the trajectory of the ith one is given by

$$s_{i,t} = s_{i,t-1} + T_s v_{i,t}^s + m_{i,t}, \quad i = 1, 2, \ldots, N_s, \tag{16}$$

where $s_{i,t}$ and $v_{i,t}^s$ are complex values that represent the ith sensor location and speed, respectively, at time t, and $m_{i,t} \sim N(m_{i,t} \mid 0, \sigma_m^2)$ is a complex Gaussian perturbation. We remark that $v_{i,t}^s$ is deterministic and known, since the sensor velocity is assigned by the tracking algorithm. Several strategies for computing $v_{i,t}^s$ given an estimate of $\mathbf{x}_t$ are possible and some of them are discussed in Section 5. As in the case of the target, we assume known prior pdf's, $s_{i,0} \sim p(s_{i,0})$, $i = 1, \ldots, N_s$, that account for the randomness in the initial deployment of the network.

By defining the vector of sensor locations, $\mathbf{s}_t = [s_{1,t}, \ldots, s_{N_s,t}]^\top$, and taking together (13) and (16), we can write the complete system state equation as

$$\begin{aligned} \mathbf{x}_t &= \mathbf{A}\mathbf{x}_{t-1} + \mathbf{Q}\mathbf{u}_t, \\ \mathbf{s}_t &= \mathbf{s}_{t-1} + T_s \mathbf{v}_t^s + \mathbf{m}_t, \end{aligned} \tag{17}$$

where $\mathbf{v}_t^s = [v_{1,t}^s, \ldots, v_{N_s,t}^s]^\top$ and $\mathbf{m}_t \sim N(\mathbf{m}_t \mid \mathbf{0}, \sigma_m^2 \mathbf{I}_{N_s})$. When needed, we will denote the complete system state as $\boldsymbol{\varsigma}_t = [\mathbf{x}_t^\top, \mathbf{s}_t^\top]^\top \in \mathbb{C}^{N_s+2}$.

The observations available for tracking $\mathbf{x}_t$ are the local decisions produced by the sensors, whereas the sensor trajectory itself is estimated from the received power of the signals transmitted by the sensors. To be specific, let $y_{i,t} \in \{0, 1\}$ be the binary output of the ith sensor, with associated probabilities

$$\begin{aligned} p(y_{i,t} = 1 \mid r_t, s_{i,t}) &= p_d(d_{i,t}, \alpha), \\ p(y_{i,t} = 0 \mid r_t, s_{i,t}) &= 1 - p_d(d_{i,t}, \alpha), \end{aligned} \tag{18}$$

where $d_{i,t} = |r_t - s_{i,t}|$ is the distance between the target and the ith sensor location at time t. Therefore, the probability of detection, p_d, is a function of distance and a known parameter $\alpha > 0$ which represents the probability of a false alarm (i.e., the probability of a positive output when there is no target within the sensor range). The specific shape of p_d depends on the physical magnitude that is measured, the sensor range of coverage as well as other practical considerations. Notwithstanding, we assume the following general properties of p_d [10]:

(i) $p_d(d_{i,t}, \alpha) \geq 0$ (since it is a probability),
(ii) $\lim_{d_{i,t} \to \infty} p_d(d_{i,t}, \alpha) = \alpha$ (this is the definition of the false alarm probability), and
(iii) p_d is a monotonically decreasing function of $d_{i,t}$.

The vector of local decisions is subsequently denoted as $\mathbf{y}_t = [y_{1,t}, \ldots, y_{N_s,t}]^\top$.

For the measurements of the received signal powers, we adopt the log-normal noise model commonly used in mobile communication applications [23], that is,

$$\begin{aligned} \pi^s_{i,t} &= 10\log_{10}\left(\frac{P_i}{|s_{i,t} - r^o|^\beta}\right) + l_{i,t} \\ &= 10\log_{10}\left(\frac{1}{|s_{i,t} - r^o|^\beta}\right) + \varpi_i + l_{i,t}, \end{aligned} \tag{19}$$

where P_i is an intermediate power of the communication signal transmitted by the ith sensor (which depends on the transmitted power, the carrier frequency, and other phenomena causing power attenuation, such as multipath fading, except the distance), $\pi^s_{i,t}$ is the power measured at the fusion center, $r^o \in \mathbb{C}$ is the location of the network fusion center, β is an attenuation parameter that depends on the physical environment [23], $l_{i,t} \sim N(l_{i,t} \mid 0, \sigma_l^2)$ is Gaussian noise and ϖ_i is shorthand for the log-power $10\log_{10} P_i$. We assume β is a deterministic and known parameter but, in order to account for unknown power attenuation effects, ϖ_i is modeled as a random variable (depending on the practical setup of the communication system we may wish to define the intermediate powers as random processes, $P_{i,t}$, which can be easily included within the state of the dynamic system and tracked together with $\mathbf{x}_t$, $\mathbf{s}_t$). The vector of measured powers at time t is denoted as $\boldsymbol{\pi}^s_t = [\pi^s_{1,t}, \ldots, \pi^s_{N_s,t}]^\top$.

We use notation $\boldsymbol{\theta}_t = [\mathbf{y}_t^\top, \boldsymbol{\pi}_t^\top]^\top$ for the complete $2N_s \times 1$ observation vector. Notice that the observation function (equivalent to f_y in (2)) for this system is highly nonlinear and it is hard even to write it in some compact form. However, assuming statistical independence among the various noise processes in the state and observation equations, it is straightforward to derive the likelihood function, namely,

$$p(\boldsymbol{\theta}_t \mid \boldsymbol{\varsigma}_{0:t}, \boldsymbol{\theta}_{1:t-1}) = p(\mathbf{y}_t \mid \mathbf{x}_t, \mathbf{s}_t)\, p(\boldsymbol{\pi}^s_t \mid \mathbf{s}_{0:t}, \boldsymbol{\pi}^s_{1:t-1}), \tag{20}$$

where the first factor can be computed by substituting (18) in

$$p(\mathbf{y}_t \mid \mathbf{x}_t, \mathbf{s}_t) = \prod_{k=1}^{N_s} p(y_{k,t} \mid r_t, s_{k,t}). \tag{21}$$

If we assume that the random variables ϖ_k, $k = 1, \ldots, N_s$, have Gaussian a priori densities $N(\varpi_k \mid \overline{\varpi}_{k,0}, \sigma^2_{k,0})$, the second factor in (21),

$$p(\boldsymbol{\pi}^s_t \mid \mathbf{s}_{0:t}, \boldsymbol{\pi}^s_{1:t-1}) = \prod_{k=1}^{N_s} p(\pi^s_{k,t} \mid s_{k,0:t}, \pi^s_{k,1:t-1}), \tag{22}$$

can be calculated recursively using the formulae

$$\widetilde{\pi}^s_{k,t} = \pi^s_{k,t} - 10\log_{10}\left(|s_{k,t} - r^o|^{-\beta}\right), \tag{23}$$

$$\overline{\varpi}_{k,t} = \frac{\sigma_l^2 \overline{\varpi}_{k,t-1} + \overline{\varpi}_{k,t-1}\widetilde{\pi}^s_{k,t}}{\sigma_l^2 + \sigma^2_{k,t-1}}, \tag{24}$$

$$\sigma^2_{k,t} = \frac{\sigma_l^2 \sigma^2_{k,t-1}}{\sigma_l^2 + \sigma^2_{k,t-1}}, \tag{25}$$

$$\begin{aligned} &p(\pi^s_{k,t} \mid s_{k,0:t}, \pi^s_{k,1:t-1}) \\ &\quad \propto \sqrt{\frac{\sigma^2_{k,t}}{\sigma_l^2 \sigma^2_{k,t-1}}} \exp\left\{-\frac{1}{2}\left(\frac{\widetilde{\pi}^{s^2}_{k,t}}{\sigma_l^2} + \frac{\overline{\varpi}^2_{k,t-1}}{\sigma^2_{k,t-1}} - \frac{\overline{\varpi}^2_{k,t}}{\sigma^2_{k,t}}\right)\right\}, \end{aligned} \tag{26}$$

as shown in Appendix 7, where $\overline{\varpi}_{k,t}$ and $\sigma^2_{k,t}$ denote the a posteriori mean and variance of ϖ_k given the power measurements $\pi^s_{k,1:t}$.

3.2. Goals

Our aim is to estimate the trajectory of the target and the sensors, $\mathbf{x}_{0:t}$ and $\mathbf{s}_{0:t}$, respectively, from the series of observations, $\boldsymbol{\theta}_{1:t}$. All information for this problem is contained in the a posteriori smoothing density $p(\boldsymbol{\varsigma}_{0:t} \mid \boldsymbol{\theta}_{1:t})$ and, in particular, the optimal (MMSE) estimator is $\widehat{\boldsymbol{\varsigma}}^{\text{opt}}_{0:t} = E_{p(\boldsymbol{\varsigma}_{0:t}|\boldsymbol{\theta}_{1:t})}[\boldsymbol{\varsigma}_{0:t}]$, where E_p denotes mathematical expectation with respect to the pdf in the subscript.

Because of the strong nonlinearities in the system model, neither the smoothing density nor the MMSE estimator have a closed form and, as a consequence, numerical methods are necessary. In the sequel, we explore the application of the SIS algorithm to approximate $p(\boldsymbol{\varsigma}_{0:t} \mid \boldsymbol{\theta}_{1:t})$ and then derive any desired track estimates.

If the probabilistic assumptions on the model (1)–(2) are not reliable enough (e.g., we may suspect that the noise processes that appear both in the state and observation equations are non-Gaussian), it may be desirable to resort to the more robust CRPF methodology. For this reason, we also extend the instance of the CRPF method proposed in [11] to incorporate the sensor motion and location uncertainty.

4. PF ALGORITHMS FOR MANEUVERING TARGET TRACKING

4.1. SIS algorithms

Assuming that a set of M particles with normalized weights, $\Omega_{t-1} = \{\boldsymbol{\varsigma}^{(i)}_{0:t-1}, w^{(i)}_{t-1}\}_{i=1}^M$, is available (observe the use of notation $\boldsymbol{\varsigma}^{(i)}_t = \begin{bmatrix} \mathbf{x}^{(i)}_t \\ \mathbf{s}^{(i)}_t \end{bmatrix}$) at time $t-1$, the application of the SIS algorithm described in Section 2.1 with a generic importance function consists of two steps:

$$\varsigma_t^{(i)} \sim q\left(\varsigma_t \mid \varsigma_{0:t-1}^{(i)}, \boldsymbol{\theta}_{1:t}\right),$$

$$w_t^{(i)} \propto w_{t-1}^{(i)} \frac{p\left(\boldsymbol{\theta}_t \mid \varsigma_{0:t}^{(i)}, \boldsymbol{\theta}_{1:t-1}\right) p\left(\varsigma_t^{(i)} \mid \varsigma_{t-1}^{(i)}\right)}{q\left(\varsigma_t^{(i)} \mid \varsigma_{0:t-1}^{(i)}, \boldsymbol{\theta}_{1:t}\right)} = w_{t-1}^{(i)} \frac{p\left(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}, \mathbf{s}_t^{(i)}\right) p\left(\boldsymbol{\pi}_t^s \mid \mathbf{s}_{0:t}^{(i)}, \boldsymbol{\pi}_{1:t-1}^s\right) p\left(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(i)}\right) p\left(\mathbf{s}_t^{(i)} \mid \mathbf{s}_{t-1}^{(i)}\right)}{q\left(\varsigma_t^{(i)} \mid \varsigma_{0:t-1}^{(i)}, \boldsymbol{\theta}_{1:t}\right)}, \tag{27}$$

where we have used the obvious fact that $p(\mathbf{x}_t, \mathbf{s}_t \mid \mathbf{x}_{t-1}, \mathbf{s}_{t-1}) = p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) p(\mathbf{s}_t \mid \mathbf{s}_{t-1})$. The simplest form of the SIS algorithm results from precisely taking the prior $p(\varsigma_t \mid \varsigma_{t-1}) = p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) p(\mathbf{s}_t \mid \mathbf{s}_{t-1})$ as a trial density [14]. In such case, the weight update equation is significantly simplified,

$$w_t^{(i)} \propto w_{t-1}^{(i)} p\left(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}, \mathbf{s}_t^{(i)}\right) p\left(\boldsymbol{\pi}_t^s \mid \mathbf{s}_{0:t}^{(i)}, \boldsymbol{\pi}_{1:t-1}^s\right), \tag{28}$$

that is, the weights are sequentially computed according to the likelihoods alone. Note at this point that the computation of $p(\boldsymbol{\pi}_t^s \mid \mathbf{s}_{0:t}^{(i)}, \boldsymbol{\pi}_{1:t-1}^s)$, by means of (26), requires that, for each particle $i = 1, \ldots, M$, we recursively update the posterior mean and variance of the log-powers $\bar{\omega}_k$, $k = 1, \ldots, N_s$, according to (23)–(25). Unfortunately, it is well known that the use of the prior $p(\varsigma_t \mid \varsigma_{t-1})$ as a proposal pdf leads to simple but inefficient algorithms, that usually need a large number of particles to achieve an adequate performance. Our computer simulations (see Section 6) show that this is the case, indeed, and the reason is that new particles are generated "blindly," without regard of the information contained in the new observations, $\boldsymbol{\theta}_t$ [14]. We hereafter use the term standard SIS (SSIS) algorithm for the procedure based on the prior proposal pdf.

In general, the performance of an SIS algorithm is highly dependent on the design of an efficient importance function, that is, one that yields particles in the regions of the state-space with large *a posteriori* probability density [13, 14]. This is normally accomplished by exploiting the latest observations when sampling the particles. In our case, and unless the variance σ_m^2 of the noise process $\mathbf{m}_t$ be too large, we can expect that sampling the sensor positions from the prior, $\mathbf{s}_t^{(i)} \sim p(\mathbf{s}_t \mid \mathbf{s}_{t-1}^{(i)})$, yield acceptable results. This is because the sensor motion given by (16) is governed by the known sequence of speed values $v_{i,t}^s$ and, as a consequence, the PF algorithm remains locked to the sequence of positions $\mathbf{s}_t$ with a high probability. Moreover, the corresponding observations are the measured powers $\boldsymbol{\pi}_t^s$, which are continuous raw data and yield a highly informative likelihood. However, tracking the target state is considerably more involved. Note that the speed $v_{0:t}$ has to be estimated, but the available observations are binary and they do not directly provide any information on the target velocity (only on its position).

Intuitively, we propose to overcome these difficulties by building an importance function that generates trial target states, $\mathbf{x}_t^{(i)}$, with a velocity component directed towards a region of the state-space with a high likelihood for the latest observations, $\mathbf{y}_t$. This scheme provides a remarkable improvement of the SIS algorithm both in terms of efficiency (a lower number of particles is required for a certain degree of estimation accuracy) and robustness (the percentage of track losses becomes negligible), as will be shown by our computer simulations. The details of the procedure are as follows. Let $\{\kappa_n\}_{n=1}^{N_1}$ be the set containing the indices of the sensors that produce a positive decision at time t, that is, $y_{\kappa_1,t} = \cdots = y_{\kappa_{N_1},t} = 1$ and all other sensors transmit a 0 value. Then, for each sample $\mathbf{x}_t^{(i)}$, we deterministically construct a set of N_1 predicted target states of the form

$$\hat{\mathbf{x}}_{t,\kappa_n}^{(i)} = \begin{bmatrix} r_{t-1}^{(i)} + T_s v_{\kappa_n,t}^{(i)} \\ v_{\kappa_n,t}^{(i)} \end{bmatrix}, \tag{29}$$

where

$$v_{\kappa_n,t}^{(i)} = \epsilon\left(s_{\kappa_n,t}^{(i)} - r_{t-1}^{(i)}\right) + (1-\epsilon) v_{t-1}^{(i)} \tag{30}$$

is the speed value that partially forces the target position towards the κ_nth sensor, with $0 < \epsilon < 1$. For each one of the predicted target states, a likelihood is computed,

$$\ell_{\kappa_n,t}^{(i)} = p\left(\mathbf{y}_t \mid \hat{\mathbf{x}}_{t,\kappa_n}^{(i)}, \hat{\mathbf{s}}_t^{(i)}\right), \tag{31}$$

where $\hat{\mathbf{s}}_t^{(i)} = \mathbf{s}_{t-1}^{(i)} + T_s \mathbf{v}_{t-1}^s$ are the predicted sensor positions. From the likelihoods, a mean velocity component is calculated for the ith particle,

$$\overline{v}_{t-1}^{(i)} = \sum_{n=1}^{N_1} v_{\kappa_n,t}^{(i)} \frac{\ell_{\kappa_n,t}^{(i)}}{\sum_{r=1}^{N_1} \ell_{\kappa_r,t}^{(i)}} \tag{32}$$

and, finally, the mean velocity values are used to generate new particles using a Gaussian proposal pdf

$$\mathbf{x}_t^{(i)} \sim N\left(\mathbf{x}_t \mid \mathbf{A}\overline{\mathbf{x}}_{t-1}^{(i)}, \sigma_u^2 \mathbf{Q}\mathbf{Q}^H\right), \tag{33}$$

where $\overline{\mathbf{x}}_{t-1}^{(i)} = \begin{bmatrix} \mathbf{r}_{t-1}^{(i)} \\ \overline{v}_{t-1}^{(i)} \end{bmatrix}$. Sensor locations are still sampled from the prior $p(\mathbf{s}_t \mid \mathbf{s}_{t-1})$.

Initialization. For $i = 1, \ldots, M$,

$\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)$, $\mathbf{s}_0^{(i)} \sim p(\mathbf{s}_0)$, choose $\overline{\omega}_{n,0}^{(i)} = \overline{\omega}_{n,0}$ $(n = 1, \ldots, N_s)$, $w_0^{(i)} = 1/M$.

Importance sampling. For $i = 1, \ldots, M$,

(1) for each one of the N_1 sensors with positive decisions and indices $\{\kappa_n\}_{n=1}^{N_1}$, compute predicted target locations

$$v_{\kappa_n,t}^{(i)} = \epsilon\left(s_{\kappa_n,t}^{(i)} - r_{t-1}^{(i)}\right) + (1-\epsilon)v_{t-1}^{(i)}, \quad 0 < \epsilon < 1,$$

$$\widehat{\mathbf{x}}_{t,\kappa_n}^{(i)} = \begin{bmatrix} r_{t-1}^{(i)} + T_s v_{\kappa_n,t}^{(i)} \\ v_{\kappa_n,t}^{(i)} \end{bmatrix};$$

(2) compute predicted likelihoods and use them for averaging the speed

$$\widehat{\mathbf{s}}_t^{(i)} = \mathbf{s}_{t-1}^{(i)} + T_s \mathbf{v}_{t-1}^{s},$$

$$\ell_{\kappa_n,t}^{(i)} = p\left(\mathbf{y}_t \mid \widehat{\mathbf{x}}_{t,\kappa_n}^{(i)}, \widehat{\mathbf{s}}_t^{(i)}\right),$$

$$\overline{v}_{t-1}^{(i)} = \sum_{n=1}^{N_1} v_{\kappa_n,t}^{(i)} \frac{\ell_{\kappa_n,t}^{(i)}}{\sum_{r=1}^{N_1} \ell_{\kappa_r,t}^{(i)}};$$

(3) sample a new target state using the mean speed

$$\overline{\mathbf{x}}_{t-1}^{(i)} = \begin{bmatrix} \mathbf{r}_{t-1}^{(i)} \\ \overline{v}_{t-1}^{(i)} \end{bmatrix},$$

$$\mathbf{x}_t^{(i)} \sim N\left(\mathbf{x}_t \mid \mathbf{A}\overline{\mathbf{x}}_{t-1}^{(i)}, \sigma_u^2 \mathbf{Q}\mathbf{Q}^H\right);$$

(4) draw new sensor locations from the prior pdf,

$$\mathbf{s}_t^{(i)} \sim p\left(\mathbf{s}_t \mid \mathbf{s}_{t-1}^{(i)}\right).$$

Weight update.

(1) Update the mean and variance of the log-powers. For $n = 1, \ldots, N_s$,

$$\widetilde{\pi}_{n,t}^{(i)} = \pi_{n,t}^{s} - 10\log_{10}\left|s_{n,t}^{(i)} - r^o\right|^{-\beta},$$

$$\overline{\omega}_{n,t}^{(i)} = \frac{\sigma_l^2 \overline{\omega}_{n,t-1}^{(i)} + \overline{\omega}_{n,t-1}^{(i)} \widetilde{\pi}_{n,t}^{s(i)}}{\sigma_l^2 + \sigma_{n,t-1}^{2(i)}}, \qquad \sigma_{n,t}^{2(i)} = \frac{\sigma_l^2 \sigma_{n,t-1}^{2(i)}}{\sigma_l^2 + \sigma_{n,t-1}^{2(i)}}.$$

(2) Update the weights

$$w_t^{(i)} \propto w_{t-1}^{(i)} p\left(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}, \mathbf{s}_t^{(i)}\right) p\left(\boldsymbol{\pi}_t^{s} \mid \mathbf{s}_{0:t}^{(i)}, \boldsymbol{\pi}_{1:t-1}^{s}\right) \frac{N\left(\mathbf{x}_t^{(i)} \mid \mathbf{A}\mathbf{x}_{t-1}^{(i)}, \sigma_u^2 \mathbf{Q}\mathbf{Q}^H\right)}{N\left(\mathbf{x}_t^{(i)} \mid \mathbf{A}\overline{\mathbf{x}}_{t-1}^{(i)}, \sigma_u^2 \mathbf{Q}\mathbf{Q}^H\right)}.$$

MMSE estimation.

$$\boldsymbol{\varsigma}_t^{\text{mmse}} = \sum_{i=1}^{M} w_t^{(i)} \boldsymbol{\varsigma}_t^{(i)}.$$

Resampling.

Let $\widehat{M}_{\text{eff}} = (\sum_{i=1}^{M} w_t^{(i)^2})^{-1}$, if $\widehat{M}_{\text{eff}} < \eta M$, $0 < \eta < 1$, then perform multinomial resampling and set $w_t^{(i)} = 1/M$, for all i.

Algorithm 1: MSIS tracking algorithm.

Since we use a different importance function for drawing $\mathbf{x}_t^{(i)}$, $i = 1, \ldots, M$, the weight update equation is not as simple as in the SSIS technique. In particular,

$$w_t^{(i)} \propto w_{t-1}^{(i)} p\left(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}, \mathbf{s}_t^{(i)}\right) p\left(\boldsymbol{\pi}_t^{s} \mid \mathbf{s}_{0:t}^{(i)}, \boldsymbol{\pi}_{1:t-1}^{s}\right) \times \frac{N\left(\mathbf{x}_t^{(i)} \mid \mathbf{A}\mathbf{x}_{t-1}^{(i)}, \sigma_u^2 \mathbf{Q}\mathbf{Q}^H\right)}{N\left(\mathbf{x}_t^{(i)} \mid \mathbf{A}\overline{\mathbf{x}}_{t-1}^{(i)}, \sigma_u^2 \mathbf{Q}\mathbf{Q}^H\right)}, \tag{34}$$

where we have written the prior $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ explicitly as a Gaussian density.

Algorithm 1 shows a summary of the SIS method with the proposed importance function, that we will refer to in the sequel as mean-velocity SIS (MSIS) algorithm. Note that resampling steps are carried out whenever the estimated effective sample size [14], $\widehat{M}_{eff} = (\sum_{i=1}^{M} w_t^{(i)^2})^{-1}$, falls below a certain threshold, ηM (with $0 < \eta < 1$). Intuitively, $\widehat{M}_{\text{eff}}$

Initialization. For $i = 1, \ldots, M$,

$\mathbf{x}_0 \sim p(\mathbf{x}_0)$, $\mathbf{s}_0 \sim p(\mathbf{s}_0)$, choose $\overline{\omega}_{n,0}^{(i)} = \overline{\omega}_{n,0}$ $(n = 1, \ldots, N_s)$, $w_0 = 1/M$.

Auxiliary index sampling. For $i = 1, \ldots, M$,

$$\hat{\mathbf{x}}_t^{(l)} = \mathbf{A}\mathbf{x}_{t-1}^{(l)}, \quad \hat{\mathbf{s}}_t^{(l)} = \mathbf{s}_{t-1}^{(l)} + T_s \mathbf{v}_t^s, \quad l = 1, \ldots, M,$$

$$\hat{\pi}_{n,t}^{(i)} = \pi_{n,t}^s - 10\log_{10}\left|\hat{s}_{n,t}^{(i)} - r^o\right|^{-\beta},$$

$$\overline{\omega}_{n,t}^{(i)} = \frac{\sigma_l^2 \overline{\omega}_{n,t-1}^{(i)} + \overline{\omega}_{n,t-1}^{(i)} \hat{\pi}_{n,t}^{s(i)}}{\sigma_l^2 + \sigma_{n,t-1}^{2(i)}}, \quad \sigma_{n,t}^2 = \frac{\sigma_l^2 \sigma_{n,t-1}^2}{\sigma_l^2 + \sigma_{n,t-1}^2}, \quad \text{for } n = 1, \ldots, N_s,$$

$$\rho_t^{(i)} = p\left(\mathbf{y}_t \mid \hat{\mathbf{x}}_t^{(i)}, \hat{\mathbf{s}}_t^{(i)}\right) p\left(\boldsymbol{\pi}_t^s \mid \mathbf{s}_{0:t-1}^{(i)}, \hat{\mathbf{s}}_t^{(i)}, \boldsymbol{\pi}_{1:t-1}^s\right),$$

$$p\,(k = l) \propto w_{t-1}^{(l)} \rho_t^{(l)},$$

$$k^{(i)} \sim p(k).$$

Target and sensor location sampling. For $i = 1, \ldots, M$,

$$\mathbf{x}_t^{(i)} \sim p\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(k^{(i)})}\right),$$

$$\mathbf{s}_t^{(i)} \sim p\left(\mathbf{s}_t \mid \mathbf{s}_{t-1}^{(k^{(i)})}\right).$$

Weight update. For $i = 1, \ldots, M$,

(1) update the mean and variance of the log-powers. For $n = 1, \ldots, N_s$,

$$\tilde{\pi}_{n,t}^{(i)} = \pi_{n,t}^s - 10\log_{10}\left|s_{n,t}^{(i)} - r^o\right|^{-\beta},$$

$$\overline{\omega}_{n,t}^{(i)} = \frac{\sigma_l^2 \overline{\omega}_{n,t-1}^{(i)} + \overline{\omega}_{n,t-1}^{(i)} \tilde{\pi}_{n,t}^{s(i)}}{\sigma_l^2 + \sigma_{n,t-1}^{2(i)}};$$

(2) update the weights

$$w_t^{(i)} \propto \frac{p\left(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}, \mathbf{s}_t^{(i)}\right) p\left(\boldsymbol{\pi}_t^s \mid \mathbf{s}_{0:t}^{(i)}, \boldsymbol{\pi}_{1:t-1}^s\right)}{\rho_t^{(k^{(i)})}}.$$

MMSE estimation.

$$\boldsymbol{\varsigma}_t^{\text{mmse}} = \sum_{i=1}^{M} w_t^{(i)} \boldsymbol{\varsigma}_t^{(i)}.$$

Algorithm 2: APF tracking algorithm.

is an estimate of how many independent samples from the true smoothing distribution would be necessary to obtain a Monte Carlo approximation with the same accuracy as that given by $\Omega_t = \{\boldsymbol{\varsigma}_t^{(i)}, w_t^{(i)}\}_{i=1}^M$.

4.2. APF

The extension of the APF given by (9)–(10) to the joint tracking of the target and the sensor trajectories yields

$$k^{(i)} \sim p_t(k),$$

$$\boldsymbol{\varsigma}_t^{(i)} \sim p\left(\boldsymbol{\varsigma}_t \mid \boldsymbol{\varsigma}_{t-1}^{(k^{(i)})}\right),$$

$$\boldsymbol{\varsigma}_{0:t}^{(i)} = \left\{\boldsymbol{\varsigma}_{0:t-1}^{(k^{(i)})}, \boldsymbol{\varsigma}_t^{(i)}\right\}, \tag{35}$$

$$w_t^{(i)} \propto \frac{p\left(\mathbf{y}_t \mid \boldsymbol{\varsigma}_t^{(i)}\right) p\left(\boldsymbol{\pi}_t^s \mid \mathbf{s}_{0:t}^{(i)}, \boldsymbol{\pi}_{1:t-1}^s\right)}{\rho_t^{(k^{(i)})}},$$

where $p_t\,(k = l) \propto w_{t-1}^{(l)} \rho_t^{(l)}$, $\rho_t^{(l)} = p(\mathbf{y}_t \mid \hat{\boldsymbol{\varsigma}}_t^{(l)}) p(\boldsymbol{\pi}_t^s \mid \hat{\mathbf{s}}_t^{(l)}, \mathbf{s}_{0:t-1}^{(l)}, \boldsymbol{\pi}_{1:t-1}^s)$, and $\hat{\boldsymbol{\varsigma}}_t^{(l)}$ is a prediction of the complete system state vector at time t computed from $\boldsymbol{\varsigma}_{t-1}^{(l)}$ alone (before sampling). As indicated in Section 2.2, the straightforward way of computing such a prediction is to take the mean of the prior pdf, in particular,

$$\hat{\boldsymbol{\varsigma}}_t^{(i)} = \begin{bmatrix} \hat{\mathbf{x}}_t^{(i)} \\ \hat{\mathbf{s}}_t^{(i)} \end{bmatrix} = E_{p(\boldsymbol{\varsigma}_t \mid \boldsymbol{\varsigma}_{t-1}^{(i)})} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{s}_t \end{bmatrix} = \begin{bmatrix} \mathbf{A}\mathbf{x}_{t-1}^{(i)} \\ \mathbf{s}_{t-1}^{(i)} + T_s \mathbf{v}_t^s \end{bmatrix}. \tag{36}$$

The likelihood computations are carried out as indicated in (20)–(26). A summary of the APF is given in Algorithm 2.

4.3. CRPF

In order to apply the CRPF methodology, we extend the algorithm proposed in [11], originally devised for tracking a single target using a network of fixed binary sensors. Specifically,

we choose an incremental cost function of the form

$$\triangle C(\boldsymbol{\varsigma}_t, \boldsymbol{\theta}_t) = (1-\zeta) D_H(\mathbf{y}_t, y(\boldsymbol{\varsigma}_t)) + \zeta \sum_{n=1}^{N_s} \left| \pi_{n,t}^s - 10\log_{10}\left(|s_{n,t} - r^o|^{-\beta} \right) - \widehat{\omega}_{n,t} \right|, \tag{37}$$

where $0 < \zeta < 1$, $D_H(\mathbf{a}, \mathbf{b})$ denotes the Hamming distance[1] between two vectors of binary symbols, $\mathbf{a}, \mathbf{b} \in \{0,1\}^k$, and $y(\boldsymbol{\varsigma}_t)$ is a vector of "artificial observations" generated deterministically as

$$y_n(\boldsymbol{\varsigma}_t) = \begin{cases} 1, & \text{if } |r_t - s_{n,t}| < \gamma \\ 0, & \text{if } |r_t - s_{n,t}| \geq \gamma \end{cases}, \quad n = 1, \ldots, N_s, \tag{38}$$

where $\gamma > 0$ is the range of coverage of a sensor. The estimate $\widehat{\omega}_{n,t}$ of the log-power ω_n is recursively computed as

$$\widehat{\omega}_{n,t} = \xi \widehat{\omega}_{n,t-1} + (1-\xi)\widetilde{\pi}_{n,t}^s, \tag{39}$$

where $\widehat{\omega}_{n,0} = \overline{\omega}_{n,0}$, $\widetilde{\pi}_{n,t}^s$ is defined in (23), and $0 < \xi < 1$. This choice of cost function enables the algorithm user to adjust the relative weight of the local binary decisions, $\mathbf{y}_t$, and the continuous power measurements, $\boldsymbol{\pi}_t^s$, on the overall cost of particles. Correspondingly, the risk function is constructed as

$$R(\boldsymbol{\varsigma}_{t-1}, \boldsymbol{\theta}_t) = \lambda C(\boldsymbol{\varsigma}_{0:t-1}, \boldsymbol{\theta}_{1:t-1}) + \triangle C\left(\begin{bmatrix} \mathbf{A}\mathbf{x}_{t-1} \\ \mathbf{s}_{t-1} + T_s \mathbf{v}_t^s \end{bmatrix}, \boldsymbol{\theta}_t \right). \tag{40}$$

For the generating function we have chosen

$$\mu\left(C_t^{(i)}\right) = \frac{1}{\left(C_t^{(i)} - \min_k \left\{ C_t^{(k)} \right\}_{k=1}^M + 1/M \right)^3}, \tag{41}$$

which was shown to work well for a related vehicle navigation application in [21] and does not require any complex computations. The selection step is carried out by the standard multinomial resampling using the pmf

$$p\left(\boldsymbol{\varsigma}_{0:t}^{(i)}\right) = \frac{\mu\left(R_t^{(i)}\right)}{\sum_{k=1}^M \mu\left(R_t^{(k)}\right)}, \quad i = 1, \ldots, M, \tag{42}$$

although alternatives exist that enable the parallelization of the CRPF algorithm if necessary [21]. The propagation scheme for $\boldsymbol{\varsigma}_t$ is given by the pair of equations

$$\begin{aligned} \mathbf{x}_t &= \mathbf{A}\mathbf{x}_{t-1} + \nu_x T_s \mathbf{z}_{x,t}, \\ \mathbf{s}_t &= \mathbf{s}_{t-1} + T_s \mathbf{v}_t^s + \nu_s T_s \mathbf{z}_{s,t}, \end{aligned} \tag{43}$$

where $\nu_x, \nu_s > 0$ are adjustable parameters that control the variance of the propagation process, and $\mathbf{z}_{x,t}$, $\mathbf{z}_{s,t}$ are complex Gaussian vectors with zero mean and covariance matrices $\mathbf{I}_2$ and $\mathbf{I}_{N_s}$, respectively. The computer simulations in Section 6 show that the pair (43) provides an appealing trade-off between simplicity (the sampling scheme is similar to the basic SIS algorithm with prior proposal pdf) and performance.

[1] The number of bits that differ between the two arguments.

The resulting instance of the CRPF method for the tracking problem at hand is summarized in Algorithm 3.

5. SENSOR MOTION

For the sake of the derivation of the PF algorithms in the previous section, we have assumed that the sensor speed values contained in $\mathbf{v}_t^s \in \mathbb{C}^{N_s}$ are given. It has been mentioned, however, that it is also a task of the tracker to compute these values and transmit them to the sensors, so that they move to locations which are (in some sense) suitable for detecting the target position and provide informative local decisions.

There are several possible strategies for the assignment of velocities to the sensors. Let us just mention some of them, all assuming that the sensors can move approximately at the same space as the target itself.

(i) *Greedy sensors*: given estimates of the target state and the sensor locations, $\widehat{\mathbf{x}}_t$ and $\widehat{\mathbf{s}}_t$, respectively, the tracker commands each sensor to move towards the target, that is, $v_{i,t+1}^s = [(\widehat{r}_t + T_s \widehat{v}_t) - \widehat{s}_{i,t}]\vartheta_t$, where $\vartheta_t > 0$ is a real scale factor used to adjust the absolute value $|v_{i,t+1}^s|$. The main weakness of this approach is the high probability of loosing the target track, either when the estimates $\widehat{\mathbf{x}}_t$ and $\widehat{\mathbf{s}}_t$ are poor or when the target makes a sharp maneuver that leads it far away from the estimated position $\widehat{r}_t$ in a short period of time.

(ii) *Inclosing the target*: a more robust strategy is to define some distance threshold around the estimated target position, say d_o, and assign the sensor speeds differently depending on whether the sensors are below the threshold or above it. In particular, we can bring far located sensors closer to the target while those already within the d_o threshold move in parallel with the target, that is,

$$v_{i,t+1}^s = \begin{cases} [(\widehat{r}_t + T_s \widehat{v}_t) - \widehat{s}_{i,t}]\vartheta_t, & \text{if } |\widehat{r}_t - \widehat{s}_{i,t}| > d_o, \\ \widehat{v}_t, & \text{if } |\widehat{r}_t - \widehat{s}_{i,t}| < d_o. \end{cases} \tag{44}$$

This is more robust to estimation mismatches and sharp target maneuvers than the greedy approach, but still has some obvious limitations. Indeed, after a few time steps, the network consists of a large number of sensors moving at a distance $\approx d_o$ from the target, and a relatively small number of sensors which are closer to the target position. If d_o is too large, the outer sensors are basically useless (they always produce 0 outputs, except when the target maneuvers away from the estimated track) and the inner ones are comparatively very few, so the network resources are wasted.

(iii) *Uniform coverage*: the aim is to uniformly cover with sensors the region $S(\widehat{r}_t, d_o) := \{x \in \mathbb{C} : |x - \widehat{r}_t| < d_o\}$, where d_o plays again the role of a distance threshold. This strategy overcomes the limitations of the *greedy* and *inclosing* approaches, but it can be computationally complex to implement. In particular, if we wish a statistically uniform coverage of $S(\widehat{r}_t, d_o)$, then it is necessary to perform a sequence of statistical tests on the population of sensor locations. If we seek

Initialization. For $i = 1, \ldots, M$,

draw $\mathbf{x}_0^{(i)}$ and $\mathbf{s}_0^{(i)}$ from a uniform distribution in the region of interest, and set $C_0^{(i)} = 0$, $\widehat{\omega}_{n,0}^{(i)} = \overline{\omega}_{n,0}$, $n = 1, \ldots, N_s$.

Selection. For $i = 1, \ldots, M$,

(1) set $\widehat{\pi}_{n,t}^{s^{(i)}} = \pi_{n,t}^{s} - 10\log_{10}|s_{n,t-1}^{(i)} + T_s v_{n,t}^{s} - r^{o}|^{-\beta}$, and $\widehat{\omega}_{n,t}^{(i)} = \xi\widehat{\omega}_{n,t-1}^{(i)} + (1-\xi)\widehat{\pi}_{n,t}^{s^{(i)}}$;

(2) compute the risks

$$R_t^{(i)} = \lambda C_{t-1}^{(i)} + \triangle C\left(\begin{bmatrix}\mathbf{A}\mathbf{x}_{t-1}^{(i)} \\ \mathbf{s}_{t-1}^{(i)} + T_s\mathbf{v}_t^{s}\end{bmatrix}, \boldsymbol{\theta}_t\right);$$

(3) resample the particles according to the probabilities

$$p\left(\boldsymbol{\varsigma}_{t-1}^{(i)}\right) = \frac{\mu\left(R_t^{(i)}\right)}{\sum_{k=1}^{M}\mu\left(R_t^{(k)}\right)}, \quad i = 1, \ldots, M;$$

(4) build the selected particle set

$$\widehat{\Omega}_{t-1} = \left\{\widetilde{\boldsymbol{\varsigma}}_{t-1}^{(i)}, \widetilde{C}_{t-1}^{(i)}\right\}_{i=1}^{M},$$

where

$$\widetilde{C}_{t-1}^{(i)} = C_{t-1}^{(k)}, \quad \widetilde{\omega}_{n,t-1}^{(i)} = \widehat{\omega}_{n,t-1}^{(k)} \quad \text{if } \widetilde{\boldsymbol{\varsigma}}_{t-1}^{(i)} = \boldsymbol{\varsigma}_{t-1}^{(k)}.$$

Propagation. For $i = 1, \ldots, M$,

$$\begin{aligned}
\mathbf{x}_t^{(i)} &\sim N\left(\mathbf{x}_t \mid \mathbf{A}\widetilde{\mathbf{x}}_{t-1}^{(i)}, (\nu_x T_s)^2\mathbf{I}_2\right),\\
\mathbf{s}_t^{(i)} &\sim N\left(\mathbf{s}_t \mid \widetilde{\mathbf{s}}_{t-1}^{(i)} + T_s\mathbf{v}_t^{s}, (\nu_s T_s)^2\mathbf{I}_{N_s}\right),\\
\widetilde{\pi}_{n,t}^{s^{(i)}} &= \pi_{n,t}^{s} - 10\log_{10}\left|s_{n,t}^{(i)} - r^{o}\right|^{-\beta},\\
\widehat{\omega}_{n,t}^{(i)} &= \xi\widetilde{\omega}_{n,t-1}^{(i)} + (1-\xi)\widetilde{\pi}_{n,t}^{s^{(i)}},\\
C_t^{(i)} &= \lambda\widetilde{C}_{t-1}^{(i)} + \triangle C(\boldsymbol{\varsigma}_t^{(i)}, \boldsymbol{\theta}_t).
\end{aligned}$$

Estimation.

$$\boldsymbol{\varsigma}_t^{\text{mean}} = \sum_{i=1}^{M}\frac{\mu\left(C_t^{(i)}\right)}{\sum_{k=1}^{M}\mu\left(C_t^{(k)}\right)}\boldsymbol{\varsigma}_t^{(i)}.$$

Algorithm 3: CRPF tracking algorithm.

a deterministic coverage, for example, a regular grid around the target, the required computations may also grow prohibitively with the number of sensors, N_s.

(iv) *Following the target*: one of the simplest methods, and the one we have adopted for the numerical experiments in Section 6, is to preserve the relative positions of the sensors, which are due to their initial deployment, and simply move the complete network with the latest velocity estimated for the target, that is,

$$v_{i,t+1}^{s} = \widehat{v}_t, \quad \forall i \in \{1, \ldots, N_s\}. \tag{45}$$

If the initial distribution of the sensors is good enough (there are no significant areas which are overpopulated, while others remain poorly covered), then this simple and computationally inexpensive strategy mitigates the drawbacks of the *greedy* and *inclosing* techniques.

6. COMPUTER SIMULATIONS

6.1. Model and algorithm parameters

The simulation of the system model described in Section 3 requires a specific choice of the detection probability function, p_d. Let $\gamma > 0$ be the range of coverage of a single sensor. The local decision probabilities are

$$\begin{aligned}
p(y_{i,t} = 1 \mid r_t, s_{i,t}) &= p_d(d_{i,t}, \alpha) = (1-\alpha)\,\text{Prob}\left\{d_{i,t} + g_{i,t} < \gamma\right\} + \alpha,\\
p(y_{i,t} = 0 \mid r_t, s_{i,t}) &= 1 - p_d(d_{i,t}, \alpha),
\end{aligned} \tag{46}$$

where $\text{Prob}\{\cdot\}$ denotes the probability of the event between braces, $d_{i,t} = |r_t - s_{i,t}|$, $g_{i,t} \sim N(0, \sigma_g^2)$ is a Gaussian perturbation, and α is the probability of false alarm. Therefore, the detection probability can be compactly written as $p_d(d_{i,t}, \alpha) = (1-\alpha)F_N(\gamma - d_{i,t} \mid 0, \sigma_g^2) + \alpha$, where $F_N(\cdot \mid \mu, \sigma^2)$ is the Gaussian cumulative distribution function with mean μ and variance σ^2.

Table 1: Fixed parameters of the dynamic system used throughout the computer simulation experiments.

Description	Parameter	Value
Observation period	T_s	1 s
Target noise variance	σ_u^2	1
Sensor noise variance	σ_m^2	$\frac{1}{3}$
Sensor range	γ	44.91 m
False alarm probability	α	10^{-3}
Distance noise variance	σ_g^2	1
Location fusion center	r^o	0
Log-powers (dB)	ϖ_n	$\varpi_n \sim N(\varpi_n \mid \overline{\varpi}_{n,0}, \sigma_{n,0}^2)$
(Mean)	$\overline{\varpi}_{n,0}$	$10\log_{10}(0.8)$
(Variance)	$\sigma_{n,0}^2$	$\frac{1}{8}$
Attenuation exponent	β	2
Log-normal noise variance	σ_l^2	$\frac{1}{3}$
No. of sensors	N_s	95

The a priori pdf's

$$p(\mathbf{x}_0) = N(0, 16\mathbf{I}_2), \quad p(\mathbf{s}_0) = N(0, 16\mathbf{I}_{N_s}) \tag{47}$$

yield the random initial position of the target and the initial deployment of the sensors.[2] As indicated in Section 5, we assume that all sensors are commanded to move in parallel with the estimated target trajectory, that is, $v_{i,t}^s = \widehat{v}_{t-1}$, where $\widehat{v}_{t-1}$ is the estimated target velocity. All other system parameters which are needed for simulation are provided in Table 1.

The specification of the SSIS and APF tracking algorithms can be deduced from the model parameters of Table 1, but the MSIS and CRPF procedures have some adjustable coefficients, which are indicated in Table 2. For the time being, the choice of these parameters is heuristic, that is, the selected values are the consequence of several simulation trials to optimize the algorithms in the sense of achieving a trade-off between robustness to lost tracks and estimation accuracy.

6.2. Numerical results

Figure 1 shows an example of application of the proposed algorithms to track a maneuvering target during 10 minutes using $M = 800$ particles. It can be observed that the MSIS and CRPF methods remain locked to the target track during the whole simulation period, and the APF algorithm follows the target trajectory during most of the time, but the SSIS procedure looses the track when a maneuver starts. This is coherent with the well-known inefficiency of the a priori importance function, which often requires a very large number of particles for adequate performance. It also justifies the effort in developing the alternative algorithms MSIS and CRPF.

An instance of a sensor track for the same simulation experiment is presented in Figure 2 (upper plot). It is seen that the true and estimated tracks remain locked (even for the SSIS algorithm, that looses the target but attains good estimates of the sensor locations). In the lower plot of Figure 2, we observe how the estimates of the corresponding log-power converge to the true value, ϖ_n. Note that $\overline{\varpi}_{n,t}$ and $\widehat{\varpi}_{n,t}$ (for the SIS-type and CRPF algorithms, resp.) are, indeed, estimates of ϖ_n.

Table 2: Adjustable parameters in the MSIS and CRPF techniques.

Algorithm	Parameter	Value
MSIS	ϵ	$\frac{1}{10}$
MSIS	η	$\frac{1}{2}$
CRPF	ζ	$\frac{1}{100}$
CRPF	ν_x	3
CRPF	ν_s	$\frac{4}{5}$
CRPF	λ	0.9
CRPF	ξ	0.98

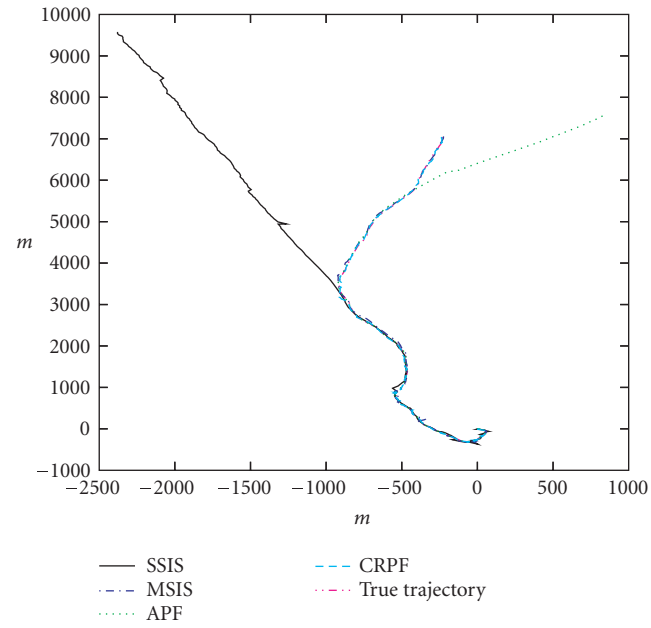

Figure 1: Example of tracking a target with the four proposed PF algorithms, all using $M = 800$ particles and $N_s = 95$ sensors, during 10 minutes.

[2] The trackers require that these prior densities be available (so that the sensors can initially collect useful data). Obviously, the more a priori uncertainty on the initial target location and speed, the larger the variance of the random variables in $\mathbf{x}_0$ and $\mathbf{s}_0$ should be.

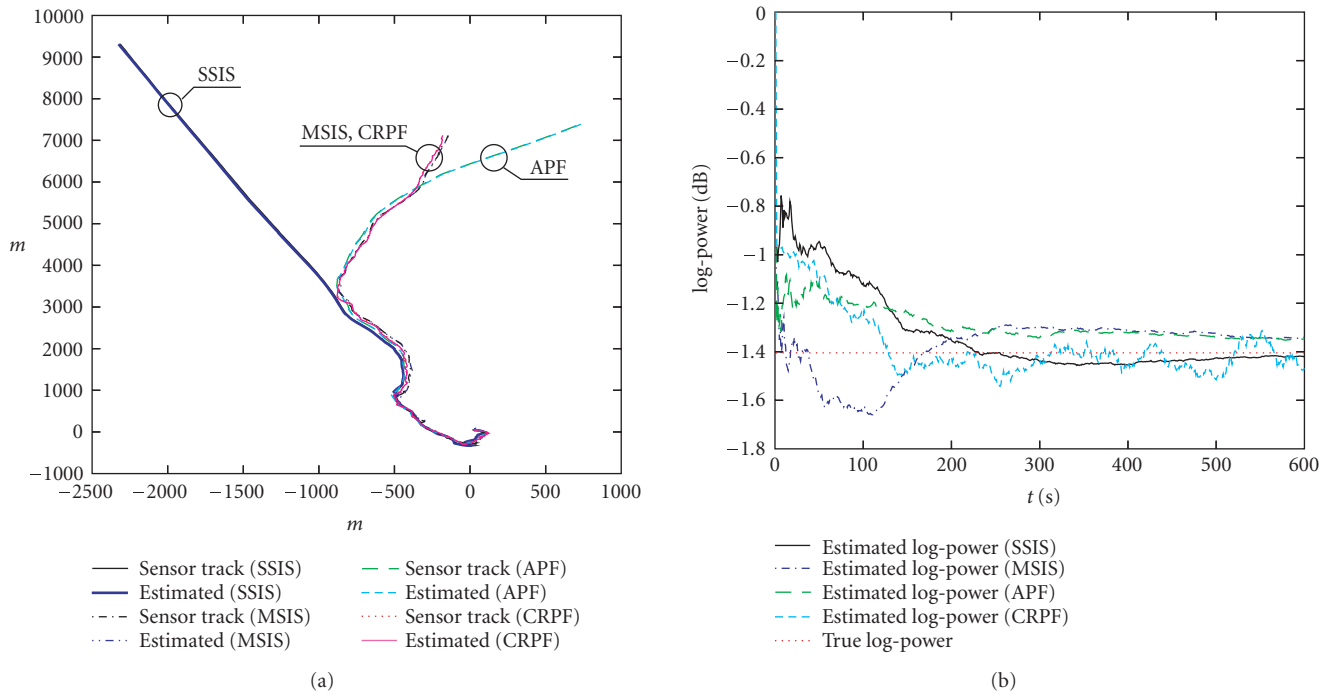

FIGURE 2: (a) Example of tracking a sensor trajectory. (b) Example of estimating the log-power ($\bar{\omega}_n$) of a communication signal. The algorithms employed in the simulation are the SSIS, MSIS, APF, and CRPF techniques, all using $M = 800$ particles to track the target during 10 minutes. The network consists of $N_s = 95$ sensors.

In order to provide a better characterization of the relative performance of the different tracking algorithms, we have carried out extensive simulations to estimate (i) the probability of achieving a correct track during 4 minutes of target motion and (ii) the average absolute deviation both in position and velocity, for correct tracks only, of each one of the algorithms. Since the performance is expected to improve when increasing the number of particles, we have carried out 200 independent simulation trials (each one with a different random trajectory) in which the tracking algorithms use $M = 100$ particles, 200 simulations using $M = 200$ particles, and another 200 trials with $M = 400$ (600 experiments, overall). For each group of simulation runs, we have calculated the mean absolute deviation

$$e_t^r = \frac{1}{200} \sum_{k=1}^{200} \left| \hat{r}_{k,t} - r_{k,t} \right|, \tag{48}$$

where $\hat{r}_{k,t}$ is the estimated position at time t provided by the tracker in the kth simulation and $r_{k,t}$ is the corresponding true position in the same simulation trial. This error is compared with a threshold of 40 m. Correct tracks are those with $(1/T_{\max}) \sum_{t=0}^{T_{\max}} e_t^r < 40$ m ($T_{\max} = 240$ s for $T_s = 1$ s). For each tracker, and each value of M, we have calculated the percentage of correct tracks.

Figure 3 shows the results attained by the SSIS, MSIS, APF, and CRPF algorithms as a function of the number of particles M. It can be seen that all techniques have a sound behavior as they improve their performance with increasing M, but the CRPF procedure and, particularly, the MSIS algorithm are much more robust than the other two methods.

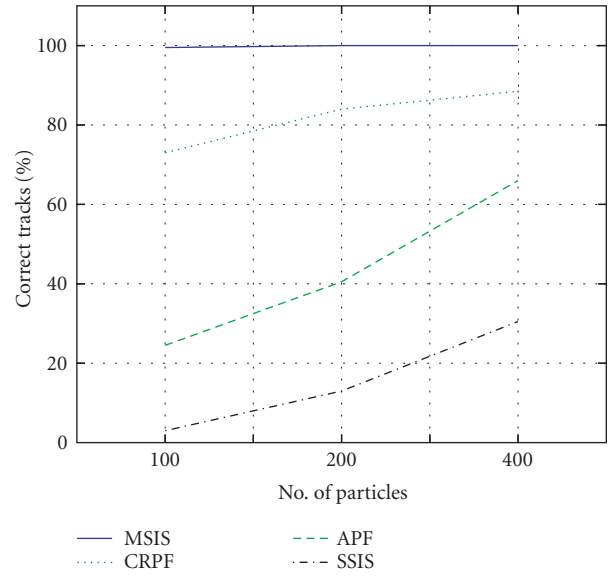

FIGURE 3: Proportion of locked (successful) target tracks versus the number of particles, M, after 4 minutes for the SSIS, MSIS, APF, and CRPF algorithms, all using an $N_s = 95$ sensor network.

To verify the accuracy of the algorithms, we have also estimated the mean absolute deviation of the successful tracks for each method, both in terms of position and speed.

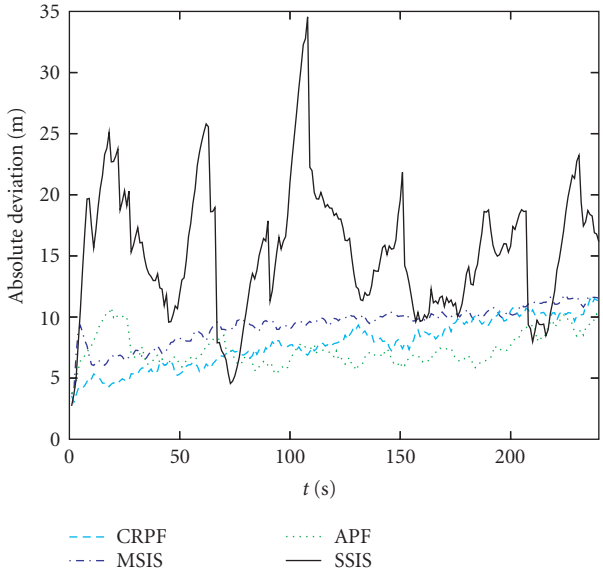

FIGURE 4: Mean absolute deviation in the position estimated by the PF tracking algorithms (with $M = 100$ particles) versus time (4 minutes). Only successful tracks have been used to compute the mean. The network consisted of $N_s = 95$ sensors.

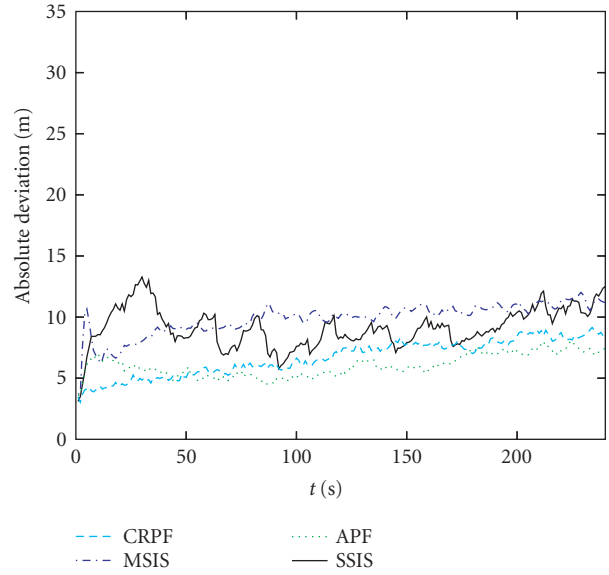

FIGURE 5: Mean absolute deviation in the position estimated by the PF tracking algorithms (with $M = 400$ particles) versus time (4 minutes). Only successful tracks have been used to compute the mean. The network consisted of $N_s = 95$ sensors.

Assume C simulation trials have yielded successful results, and let $\hat{r}_{k,t}$ and $\hat{v}_{k,t}$ be the estimates generated by the tracker in the kth of these experiments, then the mean absolute deviations are

$$\overline{e}_t^r = \frac{1}{C}\sum_{k=1}^{C} |\hat{r}_{k,t} - r_{k,t}|, \qquad \overline{e}_t^v = \frac{1}{C}\sum_{k=1}^{C} |\hat{v}_{k,t} - v_{k,t}|. \quad (49)$$

Figures 4–5 plot $\overline{e}_t^r$ versus time for $M = 100$ and $M = 400$, respectively. It can be seen that the most accurate algorithm is the APF technique, which outperforms the more robust CRPF and MSIS techniques. The reason is that the enhanced robustness of the latter methods is attained at the expense of a larger variance in the generation of new particles, that is, the algorithms explore a larger region when proposing new target state points, hence they have a better chance of keeping locked when the target maneuvers but, in general, they yield state estimates which are less accurate than those of the APF.

Finally, Figures 6–7 show the error signals $\overline{e}_t^v$ for $M = 100$ and $M = 400$, respectively. The relative performance of the algorithms is similar to what is obtained for $\overline{e}_t^r$, and for the same reasons. It is worth pointing out that, in all Figures 4–7, the CRPF algorithm exhibits a slightly better performance than the MSIS method.

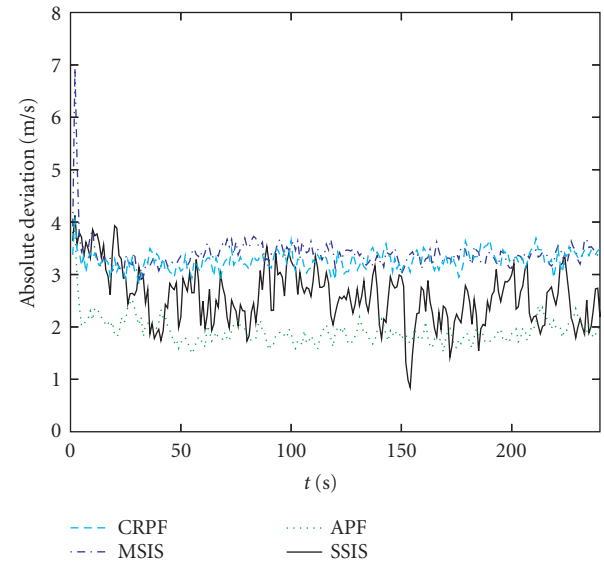

FIGURE 6: Mean absolute deviation in the speed estimated by the PF tracking algorithms (with $M = 100$ particles) versus time (4 minutes). Only successful tracks have been used to compute the mean. The network consisted of $N_s = 95$ sensors.

7. DISCUSSION

We have addressed the problem of tracking a maneuvering target that moves over a large 2-dimensional space using a limited-size network of dynamic sensors (with motion capabilities) that produce local binary decisions regarding the presence or the absence of the target in a given coverage range. The proposed approach is based on the particle filtering methodology, and we have derived four different algorithms, SSIS, MSIS, APF, and CRPF, with distinct features in terms of robustness, complexity, and tracking accuracy. Computer simulation results have been presented to illustrate the validity of the trackers.

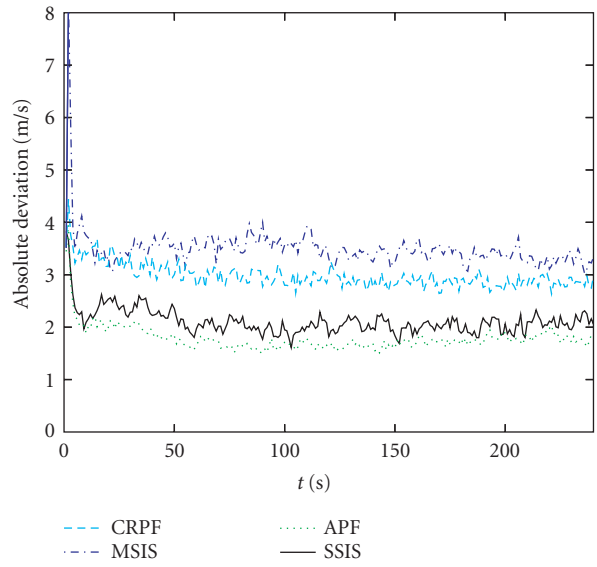

FIGURE 7: Mean absolute deviation in the speed estimated by the PF tracking algorithms, with $M = 400$ particles, versus time (4 minutes). Only successful tracks have been used to compute the mean. The network consisted of $N_s = 95$ sensors.

The proposed algorithms allow to achieve different performance goals. The CRPF and, particularly, MSIS algorithms have been demonstrated through our computer simulations to be very robust in terms of the percentage of successful tracks. However, it has also been shown that the less robust APF procedure is noticeably more accurate, both in terms of target position and target velocity tracking. It must be noted that the performance of the algorithms is fundamentally dependent on the number of particles, M, that can be processed, and all techniques become simultaneously more accurate and more robust as M grows. Therefore, if large numbers of particles can be handled, the APF is likely to be the method of choice, while the MSIS and CRPF techniques should be used when there are stringent constraints on the value of M. It should also be noted that, for equal M, the complexity of the CRPF algorithm is lower and it also lends itself to parallel hardware implementations with straightforward modifications [21] (unlike the APF and MSIS techniques).

Another issue that affects the performance of the trackers is the strategy for sensor motion, that is, how to move the network to effectively follow the target in a way that allows to obtain informative measurements. Some strategies have been discussed and we have demonstrated that the simplest one (in terms of necessary extra computations) yields acceptable results.

APPENDIX

DERIVATION OF $p(\pi_{n,t} \mid s_{n,0:t}, \pi^s_{n,1:t-1})$

We need to show that, assuming ω_n has a Gaussian a priori density $N(\omega_n \mid \overline{\omega}_{n,0}, \sigma^2_{n,0})$, then its posterior pdf is also Gaussian, $p(\omega_n \mid s_{n,0:t}, \pi_{n,1:t}) = N(\omega_n \mid \overline{\omega}_{n,t}, \sigma^2_{n,t})$, with parameters recursively calculated as

$$\overline{\omega}_{n,t} = \frac{\sigma_l^2 \overline{\omega}_{n,t-1} + \overline{\omega}_{n,t-1} \widetilde{\pi}^s_{n,t}}{\sigma_l^2 + \sigma^2_{n,t-1}}, \tag{A.1}$$

$$\sigma^2_{n,t} = \frac{\sigma_l^2 \sigma^2_{n,t-1}}{\sigma_l^2 + \sigma^2_{n,t-1}}, \tag{A.2}$$

where $\widetilde{\pi}^s_{n,t} = \pi^s_{n,t} - 10\log_{10}(|s_{n,t} - r^o|^{-\beta})$, and, as a consequence, the likelihood function $p(\pi^s_{n,t} \mid s_{n,0:t}, \pi^s_{n,1:t-1})$ has the closed form

$$p(\pi^s_{n,t} \mid s_{n,0:t}, \pi^s_{n,1:t-1}) \propto \sqrt{\frac{\sigma^2_{n,t}}{\sigma_l^2 \sigma^2_{n,t-1}}} \exp\left\{ -\frac{1}{2}\left(\frac{\widetilde{\pi}^{s^2}_{n,t}}{\sigma_l^2} + \frac{\overline{\omega}^2_{n,t-1}}{\sigma^2_{n,t-1}} - \frac{\overline{\omega}^2_{n,t}}{\sigma^2_{n,t}} \right) \right\}. \tag{A.3}$$

In order to prove the above results, consider the desired likelihood as a marginal density, that is,

$$p(\pi^s_{n,t} \mid s_{n,0:t}, \pi^s_{n,1:t-1}) = \int_{\mathbb{R}} p(\pi^s_{n,t} \mid \omega_n, s_{n,t})\, p(\omega_n \mid s_{n,0:t-1}, \pi^s_{n,1:t-1})\, d\omega_n. \tag{A.4}$$

We note that

$$p(\omega_n \mid s_{n,0:t}, \pi^s_{n,1:t}) \propto p(\pi^s_{n,t} \mid \omega_n, s_{n,t})\, p(\omega_n \mid s_{n,0:t-1}, \pi^s_{n,1:t-1}) \tag{A.5}$$

and, since both

$$\begin{aligned} p(\pi^s_{n,t} \mid \omega_n, s_{n,t}) &= N(\widetilde{\pi}^s_{n,t} \mid \omega_n, \sigma_l^2), \\ p(\omega_n \mid s_{n,0}, \pi^s_{n,1:0}) &= p(\omega_n) = N(\omega_n \mid \overline{\omega}_{n,0}, \sigma^2_{n,0}) \end{aligned} \tag{A.6}$$

are Gaussian, we conclude that

$$p(\omega_n \mid s_{n,0:t}, \pi^s_{n,1:t}) = N(\omega_n \mid \overline{\omega}_{n,t}, \sigma^2_{n,t}) \tag{A.7}$$

for all t, that is, ω_n is *a posteriori* Gaussian given $s_{n,0:t}$ and $\pi^s_{n,1:t}$. As a consequence, we can rewrite (A.4) as

$$p(\pi^s_{n,t} \mid s_{n,0:t}, \pi^s_{n,1:t-1}) \propto \frac{1}{\sqrt{\sigma_l^2 \sigma^2_{n,t-1}}} \int_{\mathbb{R}} \exp\left\{ -\frac{1}{2}\left(\frac{(\widetilde{\pi}^s_{n,t} - \omega_n)^2}{\sigma_l^2} + \frac{(\omega_n - \overline{\omega}_{n,t-1})^2}{\sigma^2_{n,t-1}} \right) \right\} d\omega_n \tag{A.8}$$

and elementary algebraic manipulations of (A.8) lead to

$$\frac{(\widetilde{\pi}^s_{n,t} - \omega_n)^2}{\sigma_l^2} + \frac{(\omega_n - \overline{\omega}_{n,t-1})^2}{\sigma^2_{n,t-1}} = \frac{(\omega_n - \overline{\omega}_{n,t})^2}{\sigma^2_{n,t}} + \frac{\widetilde{\pi}^{s^2}_{n,t}}{\sigma_l^2} + \frac{\overline{\omega}^2_{n,t-1}}{\sigma_{n,t-1}} - \frac{\overline{\omega}^2_{n,t}}{\sigma^2_{n,t}}, \tag{A.9}$$

where $\overline{\omega}_{n,t}$ and $\sigma^2_{n,t}$ are obtained as shown by (A.1) and (A.2), respectively. Finally, substituting (A.9) into (A.8) yields the likelihood of (A.3).

ACKNOWLEDGMENTS

This work has been partially supported by *Xunta de Galicia* (project PGIDIT04TIC105008PR), the Ministry of Education and Science of Spain (project DOIRAS, TIC2003-02602), the European Commission (Network of Excellence "CRUISE," IST-4-027738), and Comunidad de Madrid (project PRO-MULTIDIS-CM, S0505/TIC/0223).

REFERENCES

[1] R. C. Luo, C.-C. Yih, and K. L. Su, "Multisensor fusion and integration: approaches, applications and future research directions," *IEEE Sensors Journal*, vol. 2, no. 2, pp. 107–119, 2002.

[2] C.-Y. Chong and S. P. Kumar, "Sensor networks: evolution, opportunities, and challenges," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247–1256, 2003.

[3] F. Zhao and L. Guibas, *Wireless Sensor Networks*, Morgan Kaufman, New York, NY, USA, 2004.

[4] R. R. Brooks, P. Ramanathan, and A. M. Sayeed, "Distributed target classification and tracking in sensor networks," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1163–1171, 2003.

[5] L. Doherty, B. A. Warneke, B. E. Boser, and K. S. J. Pister, "Energy and performance considerations for smart dust," *International Journal of Parallel and Distributed Systems and Networks*, vol. 4, no. 3, pp. 121–133, 2001.

[6] N. Patwari, A. O. Hero III, M. Perkins, N. S. Correal, and R. J. O'Dea, "Relative location estimation in wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 51, no. 8, pp. 2137–2148, 2003.

[7] A. Savvides, L. Girod, M. B. Srivastava, and D. Estrin, "Localization in sensor networks," in *Wireless Sensor Networks*, C. S. Raghavendra, K. M. Sivalingham, and T. Znati, Eds., Kluwer Academic, Boston, Mass, USA, 2004.

[8] A. Savvides, C.-C. Han, and M. B. Srivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MOBICOM '01)*, pp. 166–179, Rome, Italy, July 2001.

[9] A. T. Ihler, J. W. Fisher III, R. L. Moses, and A. S. Willsky, "Nonparametric belief propagation for self-calibration in sensor networks," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN '04)*, pp. 225–233, Berkeley, Calif, USA, April 2004.

[10] A. Artés-Rodríguez, "Decentralized detection in sensor networks using range information," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '04)*, vol. 2, pp. 265–268, Montreal, Quebec, Canada, May 2004.

[11] J. Míguez, M. F. Bugallo, and P. M. Djurić, "Decision fusion for distributed target tracking using cost reference particle filtering," in *Proceedings of 13th European Signal Processing Conference (EUSIPCO '05)*, Antalya, Turkey, September 2005.

[12] P. M. Djurić, M. Vemula, M. F. Bugallo, and J. Míguez, "Non-cooperative localization of binary sensors," in *Proceedings of 13th IEEE Workshop on Statistical Signal Processing (SSP '05)*, Bordeaux, France, July 2005.

[13] J. S. Liu and R. Chen, "Sequential Monte Carlo methods for dynamic systems," *Journal of the American Statistical Association*, vol. 93, no. 443, pp. 1032–1044, 1998.

[14] 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.

[15] M. S. Arulumpalam, S. Maskell, N. Gordon, and T. Klapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[16] P. M. Djurić, J. H. Kotecha, J. Zhang, et al., "Particle filtering," *IEEE Signal Processing Magazine*, vol. 20, no. 5, pp. 19–38, 2003.

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

[18] A. Doucet, N. de Freitas, and N. Gordon, "An introduction to sequential Monte Carlo methods," in *Sequential Monte Carlo Methods in Practice*, A. Doucet, N. de Freitas, and N. Gordon, Eds., chapter 1, pp. 4–14, Springer, New York, NY, USA, 2001.

[19] R. Douc, O. Cappé, and E. Moulines, "Comparison of resampling schemes for particle filtering," in *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis (ISPA '05)*, pp. 64–69, Zagreb, Croatia, September 2005.

[20] M. K. Pitt and N. Shephard, "Auxiliary variable based particle filters," in *Sequential Monte Carlo Methods in Practice*, A. Doucet, N. de Freitas, and N. Gordon, Eds., chapter 13, pp. 273–293, Springer, New York, NY, USA, 2001.

[21] J. Míguez, M. F. Bugallo, and P. M. Djurić, "A new class of particle filters for random dynamic systems with unknown statistics," *EURASIP Journal on Applied Signal Processing*, vol. 2004, no. 15, pp. 2278–2294, 2004.

[22] F. Gustafsson, F. Gunnarsson, N. Bergman, et al., "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 425–437, 2002.

[23] T. S. Rappaport, *Wireless Communications: Principles and Practice*, Prentice-Hall, Upper Saddle River, NJ, USA, 2nd edition, 2001.

Joaquín Míguez was born in Ferrol (Galicia, Spain) in 1974. He obtained the Licenciado en Informática (M.S.) and Doctor en Informática (Ph.D.) degrees from Universidade da Coruña (Spain), in 1997 and 2000, respectively. Late in 2000, he joined Departamento de Electrónica e Sistemas, Universidade da Coruña, where he became an Associate Professor in July 2003. From April to December 2001 he was a Visiting Scholar in the Department of Electrical and Computer Engineering, Stony Brook University, and, since October 2004, he has been with Departamento de Teoría de la Señal y Comunicaciones, Universidad Carlos III de Madrid (Spain). His research interests are in the field of statistical signal processing, with emphasis in the topics of Bayesian analysis, sequential Monte Carlo methods, adaptive filtering, stochastic optimization, and their applications to multiuser communications, smart antenna systems, sensor networks, target tracking, and vehicle positioning and navigation.

Antonio Artés-Rodríguez received the Ingeniero de Telecomunicación and Doctor Ingeniero de Telecomunicación degrees, both from Universidad Politécnica de Madrid, Spain, in 1988 and 1992, respectively. He is currently a Professor at Departamento de Teoría de la Señal y Comunicaciones, Universidad Carlos III de Madrid, Spain. Prior to this, he has occupied different teaching and research positions at Universidad de Vigo, Universidad Politécnica de Madrid, and Universidad de Alcalá, all of them in Spain. He has participated in more than 40 projects and contracts and he has coauthored more than 100 journal and international conference papers. His research interests include signal processing, learning, and information theory methods, and its application to sensor networks, communications, and medical applications.