14th European Signal Processing Conference (EUSIPCO 2006), Florence, Italy, September 4-8, 2006, copyright by EURASIP

A PARTICLE FILTER FOR BEACON-FREE NODE LOCATION AND TARGET TRACKING IN SENSOR NETWORKS

Joaquín Míguez, Antonio Artés-Rodríguez

Departamento de Teoría de la Señal y Comunicaciones, Universidad Carlos III de Madrid.
Avenida de la Universidad 30, 28911 Leganés, Madrid, Spain. E-mail: {jmiguez, antonio}@ieee.org.

ABSTRACT

We address the problem of tracking a maneuvering target that moves along a region monitored by a sensor network, whose nodes, including both the sensors and the data fusion center (DFC), are located at unknown positions. Therefore, the node locations and the target track must be estimated jointly without the aid of beacons. We assume that, when the network is started, each sensor is able to detect the presence of other nodes within its range and transmit the resulting binary data to the DFC. After this startup phase, the sensor nodes just measure some physical magnitude related to the target position and/or velocity and transmit it to the DFC. At the DFC, a particle filtering (PF) algorithm is used to integrate all the collected data and produce on-line estimates of both the (static) sensor locations and the (dynamic) target trajectory. The validity of the method is illustrated by computer simulations of a network of power-aware sensors.

1. INTRODUCTION

Sensor networks will soon become ubiquitous because of their suitability for a broad range of emerging applications, such as environmental monitoring, surveillance and security, vehicle navigation, tracking, logistics, etc... For virtually any of these applications, the accurate localization of the sensors is a key task. Indeed, automatic node positioning has been recognized as an enabling technology, since the data measured by a sensor is hardly useful unless it is precisely known *where* it has been collected [1]. Most sensor localization algorithms rely on the availability of beacons, i.e., network nodes with known position that can be taken as reference [1]. Although beacon-free network designs are feasible [2], they usually involve complicated energy-consuming local communications among nodes.

In this paper, we address the problem of using a beacon-free network of sensors with unknown locations to track a maneuvering target. The sensors are assumed to measure some distance-dependent physical magnitude and have two modes of operation. When the network is started, each sensor is able to detect the presence of other nodes within a certain range and transmit the resulting binary data to the DFC. The energy cost of this operation is minimal, because it is only carried out at startup and just a few bits have to be transmitted. Using this information a "picture" of the relative positions of the nodes can be created at the DFC. After the startup, the sensors go into a normal operation mode that consists of measuring some physical magnitude related to the target position and/or velocity and periodically transmit it to the DFC. At the DFC, we propose to use a particle filtering (PF) algorithm to recursively integrate all the collected data and produce on-line estimates of both the (static) sensor locations and the (dynamic) target trajectory. The validity of the method is illustrated by computer simulations of a network of power-aware sensors.

The remaining of the paper is organized as follows. In Section 2, we provide a mathematical model of the class of systems under consideration. The proposed algorithm is described in Section 3. In Section 4 we present illustrative computer simulation results for a network of power-aware sensors. Finally, Section 5 is devoted to the conclusions.

This work was supported by Xunta de Galicia (project PGIDIT04TIC105008PR), Ministerio de Educaci´on y Ciencia of Spain (project 'DOIRAS', id. TIC2003-02602) and the UE (Network of Excellence 'CRUISE', id. IST-4-027738).

2. SYSTEM MODEL

We assume that the target moves along a 2-dimensional region according to the linear model [3]

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{q}u_t, \quad t = 1,2,3,\ldots \tag{1}$$

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, $r_t \in \mathbb{C}$ and $v_t \in \mathbb{C}$, respectively; $\mathbf{A} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}$ is a transition matrix that depends on the observation period, T_s; $\mathbf{q} = [T_s^2/2, T_s]^\top$ and $u_t \sim CN(u_t|0,\sigma_u^2)$ is a complex Gaussian noise term with zero mean and variance σ_u^2. The initial target state, $\mathbf{x}_0$, has a known prior probability density function (pdf), $p(\mathbf{x}_0)$.

The N_s sensors in the network are located at random (but fixed) unknown positions $s_{1:N_s} := \{s_1, s_2, \ldots, s_{N_s}\}$, with known prior pdf, $p(s_{1:N_s})$. During the network startup, each sensor detects any other nodes which are located within a certain range, $\gamma > 0$. In particular, the n-th sensor builds up a $N_s \times 1$ 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

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

where $d^s_{n,k} = |s_n - s_k|$ is the distance between the n-th and k-th sensors, $p_d(\cdot,\cdot)$ is the function that yields the probability of detection and $0 \leq \alpha < 1$ is the false-alarm rate. These decisions are transmitted to the DFC 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.

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 $y_{n,t} = f_s(d_{n,t}, \varepsilon_t)$, where $d_{n,t} = |r_t - s_n|$ is the distance between the

target and the sensor, ε_t is a random perturbation with known pdf and $f_s(\cdot,\cdot)$ is the measurement function. Thus, an $N_t \times 1$ vector of observations, $\mathbf{y}_t := [y_{\kappa(1),t}, \ldots, y_{\kappa(N_t),t}]^\top$, where $\kappa(i) \in \{1,\ldots,N_s\}$, $\forall i$, is transmitted to the DFC. We assume that the likelihood $p(\mathbf{y}_t|\mathbf{x}_t, s_{1:N_s})$ can be evaluated. Note that not every sensor necessarily transmits 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 measurement, hence $N_t \leq N_s$ in general.

By convention, the DFC is assumed to be located at the origin of the monitored region. We also assume that the 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 DFC also has available, at time $t \geq 0$, the $N_t \times 1$ data vector $\mathbf{z}_t = [z_{\kappa(1),t}, \ldots, z_{\kappa(N_t),t}]^\top$, where $z_{n,t} = f_s(|s_n|, \varepsilon_t)$, $d^c_{n,t} = |s_n|$ and ε_t is a random perturbation with known pdf, so that $p(\mathbf{z}_t|s_{1:N_s})$ can be computed. Note that $\mathbf{z}_0$ is defined (unlike $\mathbf{y}_0$), and has dimension $N_0 = N_s$, because during the network startup all sensors transmit signals to the DFC.

Our goal is to jointly estimate the target track $\mathbf{x}_{0:t} := \{\mathbf{x}_0, \ldots, \mathbf{x}_t\}$, and the sensor locations, $s_{1:N_s}$, from the decisions in $\mathbf{B}$ and the sequence of data vectors $\mathbf{y}_{1:t} := \{\mathbf{y}_1, \ldots, \mathbf{y}_t\}$ and $\mathbf{z}_{0:t} := \{\mathbf{z}_0, \ldots, \mathbf{z}_t\}$ without the aid of beacons. Note that, because of the use of distance-aware measurements and the lack of any absolute reference position, the estimates are subject to an inherent rotation ambiguity.

3. ALGORITHM

Eq. (1) and the observations $\{\mathbf{B}, \mathbf{y}_{t\geq 1}, \mathbf{z}_{t\geq 0}\}$ describe a dynamic system in state-space form. It has the peculiarity that the state consists of a time-varying component, $\mathbf{x}_t$, and a static component, $s_{1:N_s}$. For this reason, it is not possible to apply a standard particle filter to perform the desired estimation task. In this paper, we propose to use an auxiliary particle filter (APF) algorithm for state estimation in dynamic systems with unknown fixed parameters, based on the technique orignally proposed in [4].

Another specific feature of the system presented in Section 2 is that, at time $t = 0$, there is a subset of observations, $\{\mathbf{B}, \mathbf{z}_0\}$, that provides information on the sensor locations (but *not* on the target). In order to adequately describe the processing of these observations, it is convenient to consider two steps in the proposed APF algorithm.

3.1 Initialization

A particle filter is a recursive method that approximates a sequence of desired pdf's by means of sets of weighted samples, usually termed *particles* [5]. The pdf of interest at time t is, in our case, $p(\mathbf{x}_t, s_{1:N_s}|\mathbf{B}, \mathbf{z}_{0:t}, \mathbf{y}_{1:t})$, and the associated set of particles is $\Omega_t = \left\{\mathbf{x}_t^{(i)}, s_{1:N_s,t}^{(i)}, w_t^{(i)}\right\}_{i=1}^M$, where M is the number of particles and $w_t^{(i)}$ are normalized importance weights. When Ω_t is properly built, we can approximate the desired pdf as

$$p(\mathbf{x}_t, s_{1:N_s}|\mathbf{B}, \mathbf{z}_{0:t}, \mathbf{y}_{1:t}) \approx p_M(\mathbf{x}_t, s_{1:N_s}|\mathbf{B}, \mathbf{z}_{0:t}, \mathbf{y}_{1:t}) = \sum_{i=1}^M \delta_i(\mathbf{x}_t)\delta_i(s_{1:N_s}) w_t^{(i)}, \quad (3)$$

where $\delta_i(\mathbf{x}_t) = \delta(\mathbf{x}_t - \mathbf{x}_t^{(i)})$ and $\delta_i(s_{1:N_s}) = \delta(s_{1:N_s} - s_{1:N_s,t}^{(i)})$ are Dirac delta functions.

At time $t = 0$, the desired pdf reduces to

$$p(\mathbf{x}_0, s_{1:N_s}|\mathbf{B}, \mathbf{z}_0) = p(\mathbf{x}_0)p(s_{1:N_s}|\mathbf{B}, \mathbf{z}_0) \propto p(\mathbf{x}_0)p(s_{1:N_s})p(\mathbf{z}_0|s_{1:N_s})\textstyle\prod_{m\neq k} p(b_{n,k}|s_{1:N_s}), \quad (4)$$

and we need to build the first particle set, Ω_0, from scratch. The simplest way to construct Ω_0 is by importance sampling (IS) [6], using the (known) priors $p(\mathbf{x}_0)$ and $p(s_{1:N_s})$ as importance functions, i.e., we

$$\text{draw samples:} \quad \begin{cases} \mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0) \\ s_{1:N_s,0}^{(i)} \sim p(s_{1:N_s}) \end{cases} \quad (5)$$

$$\text{compute weights:} \quad \tilde{w}_0^{(i)} = p(\mathbf{z}_0|s_{1:N_s,0}^{(i)}) \prod_{m\neq k} p(b_{n,k}|s_{1:N_s,0}^{(i)}) \quad (6)$$

$$\text{and normalize:} \quad w_0^{(i)} = \frac{\tilde{w}_0^{(i)}}{\sum_{k=1}^M \tilde{w}_0^{(k)}}. \quad (7)$$

Notice that these initial weights are independent of the target samples, $\mathbf{x}_0^{(i)}$.

3.2 Tracking

The aim is to track the sequence of states $\mathbf{x}_{0:t}$, and improve the estimation of $s_{1:N_t}$ given the measurements $\mathbf{y}_{1:t}$ and $\mathbf{z}_{1:t}$. Using the PF methodology, this is achieved by recursively building Ω_t from Ω_{t-1} when the new observations, $\{\mathbf{y}_t, \mathbf{z}_t\}$, are available.

In this paper, we propose to carry out the recursive update of Ω_t by means of an APF procedure based on the method of [4] and summarized in Table 1. The algorithm is derived from the relationship

$$p(\mathbf{x}_t, s_{1:N_s}|\mathbf{y}_{1:t}, \mathbf{z}_{1:t}, \mathbf{B}) \propto p(\mathbf{y}_t, \mathbf{z}_t|\mathbf{x}_t, s_{1:N_s}) \times \times p(\mathbf{x}_t|s_{1:N_s}, \mathbf{y}_{1:t-1}) p(s_{1:N_s}|\mathbf{y}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{B}) \quad (8)$$

and the approximations

$$p_M(\mathbf{x}_t|s_{1:N_s}, \mathbf{y}_{1:t-1}) = \textstyle\sum_{i=1}^M p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)})\delta_i(s_{1:N_s}) w_{t-1}^{(i)} \quad (9)$$

$$p_M(s_{1:N_s}|\mathbf{y}_{1:t-1}, \mathbf{z}_{1:t-1}) = \textstyle\sum_{i=1}^M w_{t-1}^{(i)} K_i(s_{1:N_s}), \quad (10)$$

where $K_i(\cdot)$ is a symmetric kernel. For the latter, we have chosen

$$K_i(s_{1:N_s}) = CN(s_{1:N_s}|\overline{\mathbf{s}}_{t-1}^{(i)}, h^2\boldsymbol{\Sigma}_{t-1}) \quad (11)$$

where

$$\overline{\mathbf{s}}_{t-1}^{(i)} = [\overline{s}_{1,t-1}^{(i)}, \ldots, \overline{s}_{N_s,t-1}^{(i)}]^\top \quad (12)$$

$$\boldsymbol{\Sigma}_{t-1} = \text{diag}\{\sigma_{1,t-1}^2, \ldots, \sigma_{N_s,t-1}^2\} \quad (13)$$

and $h \in (0,1)$ is a bandwidth factor. The kernel modes are calculated as $\overline{s}_{k,t-1}^{(i)} = a s_{k,t-1}^{(i)} + (1-a)\overline{s}_{k,t-1}$, for $a = \sqrt{1-h^2}$ and $\overline{s}_{k,t-1} = \sum_{k=1}^M w_{t-1}^{(i)} s_{k,t-1}^{(i)}$. The variances, in turn, are found as $\sigma_{k,t-1}^2 = \sum_{l=1}^M w_{t-1}^{(l)} \left| s_{k,t-1}^{(l)} - \overline{s}_{k,t-1} \right|^2$. This choice of

Given $\Omega_{t-1} = \left\{ \left(\mathbf{x}_{t-1}, s_{1:N_s,t-1}\right)^{(i)}, w_{t-1}^{(i)} \right\}_{i=1}^{M}$:

(1) Compute $\tilde{\mathbf{x}}_t^{(i)} = \mathbf{A}\mathbf{x}_{t-1}^{(i)}$, $i = 1,...,M$.

(2) Draw indices $\ell^{(i)} \sim q_t(\ell)$, $i = 1,...,M$, where $q_t(\ell) \propto w_{t-1}^{(\ell)^2} p(\mathbf{y}_t|\tilde{\mathbf{x}}_t^{(\ell)}, s_{1:Ns,t-1}^{(\ell)}) p(\mathbf{z}_t|s_{1:N_s,t-1}^{(\ell)})$.

(3) Draw $s_{1:N_s,t}^{(i)} \sim q_t(s_{1:N_s}|\ell^{(i)})$, where $q_t(s_{1:N_s}|\ell^{(i)}) = CN(s_{1:N_s}|\overline{\mathbf{s}}_{t-1}^{(\ell^{(i)})}, h^2\mathbf{\Sigma}_{t-1})$, for $i = 1,...,M$.

(4) Draw target states $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(\ell^{(i)})})$, $i = 1,...M$, and build the trajectory $\mathbf{x}_{0:t}^{(i)} = \{\mathbf{x}_{0:t-1}^{(\ell^{(i)})}, \mathbf{x}_t^{(i)}\}$.

(5) Update the weights, for $i = 1,...,M$, $w_t^{(i)} \propto \frac{p(\mathbf{y}_t|\mathbf{x}_t^{(i)}, s_{1:N_s,t}^{(i)}) p(\mathbf{z}_t|s_{1:N_s,t}^{(i)})}{p(\mathbf{y}_t|\tilde{\mathbf{x}}_t^{(\ell^{(i)})}, s_{1:N_s,t}^{(\ell^{(i)})}) p(\mathbf{z}_t|s_{1:N_s,t}^{(\ell^{(i)})})}$.

(6) MAP estimation, $i_o = \arg\min_{i\in\{1,...,M\}}\{w_t^{(i)}\}$, $\left(\mathbf{x}_t^{MAP}, s_{1:N_s,t}^{MAP}\right) = \left(\mathbf{x}_{0:t}, s_{1:N_s,t}\right)^{(i_o)}$.

Table 1: APF algorithm for joint estimation of the target trajectory, $\mathbf{x}_{0:t}$, and the fixed node locations, $s_{1:N_s}$, from the measurements collected at the DFC.

$\overline{\mathbf{s}}_{t-1}^{(i)}$ and $\mathbf{\Sigma}_{t-1}$ ensures that the mean and the marginal variance of every fixed parameter given by the kernel approximation (10) is equal to the corresponding mean and marginal variance given by the weights [4].

One difficulty with the approximations (9) and (10) is that they involve mixtures of a typically large number (M) of pdf's. We avoid this limitation by incorporating a discrete auxiliary random variable $\ell \in \{1,...,M\}$ that indicates the terms in (9) and (10) to be selected [7]. In particular, we define

$$p(\mathbf{x}_t, s_{1:N_s}, \ell|\mathbf{y}_{1:t}, \mathbf{z}_{0:t}, \mathbf{B}) \propto p(\mathbf{y}_t, \mathbf{z}_t|\mathbf{x}_t, s_{1:N_s}) \times \times p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(\ell)}) w_{t-1}^{(\ell)^2} K_\ell(s_{1:N_s}, r_n^o). \quad (14)$$

Using (14) we can easily draw particles and compute weights by applying the principle of IS. In particular, we define the importance pdf

$$q_t(\mathbf{x}_t, s_{1:N_s}, \ell) \propto q_t(\ell) q_t(s_{1:N_s}|\ell) p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(\ell)}) \quad (15)$$

that we use for drawing new particles and then update the weights as

$$w_t^{(i)} \propto \frac{p\left(\left(\mathbf{x}_t, s_{1:N_s,t}, \ell\right)^{(i)}|\mathbf{y}_{1:t}, \mathbf{z}_{0:t}\right)}{q_t\left(\left(\mathbf{x}_t, s_{1:N_s,t}, \ell\right)^{(i)}\right)}, i = 1,\ldots,M. \quad (16)$$

The auxiliary variables, $\ell^{(i)}$, are discarded before proceeding to time $t+1$. See Table 1 for the details.

We finally note that, given Ω_t, it is straightforward to produce estimates of the target trajectory and the node locations (in particular, it is enough to select the particle with the largest weight, as shown in Table 1). Thus, at any given time t, the DFC can produce an approximate MAP estimate of $\mathbf{x}_{0:t}$ and $s_{1:N_s}$.

4. COMPUTER SIMULATIONS

In order to provide illustrative numerical results, we have particularized the model of Section 2 to a network of power-aware sensors. Thus, the measurement functions f_s becomes

$$f_s(d,\varepsilon) = 10\log_{10}\left(\frac{1}{d^2} + \eta\right) + \varepsilon_t, \quad \text{(dB)} \quad (17)$$

where ε_t is zero-mean Gaussian and $\eta = 10^{-6}$ accounts for the power of the background noise (-60 dB). The n-th sensor transmits its measurement, $y_{n,t}$, only if it corresponds to a distance $d_{n,t} < 25$ m (i.e., $y_{n,t} > P_u = -27.95$ dB) and otherwise remains silent (for battery saving). Since ε_t is Gaussian, the associated likelihoods can be easily derived, namely

$$p(\mathbf{y}_t|\tilde{\mathbf{x}}_t, \tilde{s}_{1:N_s}) = \prod_{i=1}^{N_t} N\left(y_{\kappa(i),t}|\, f_s(|\tilde{r}_t - \tilde{s}_{\kappa(i)}|, 0), \frac{1}{2}\right) \quad (18)$$

$$p(\mathbf{z}_t|\tilde{s}_{1:N_s}) = \prod_{i=1}^{N_t} N\left(z_{\kappa(i),t}|\, f_s(|\tilde{s}_{\kappa(i)}|, 0), \frac{1}{10}\right). \quad (19)$$

The likelihoods computed from the binary data in $\mathbf{B}$ are also derived from power measurements. In particular, for $n \neq k$,

$$p_d(d_{n,k}^s, \alpha) = \alpha + (1-\alpha)\left(1 - \Phi\left(\frac{P_u - f_s(|s_n - s_k|, 0)}{\sqrt{1/10}}\right)\right), \quad (20)$$

where $\Phi(\cdot)$ is the standard Gaussian cumulative distribution function, $P_u = -27.95$ dB and $\alpha = 10^{-4}$.

The monitored region is a square of side $L = 100$ m and there are $N_s = 16$ sensors in the network, *a priori* distributed as $s_n \sim CN(s_n|\mu_n, \sigma_s^2)$, where $\sigma_s^2 = 49$ and the means μ_n, $n = 1,...,N_s$, are points in a regular square grid. The state prior is $p(\mathbf{x}_0) = CN\left(\mathbf{x}_0|[0,0]^\top, \text{diag}\{100, \frac{1}{10}\}\right)$ and the system noise variance is $\sigma_u^2 = \frac{1}{4}$.

Figure 1 shows the results of a typical simulation run with observation period $T_s = 0.25$ s, 45 s of total simulated time and $M = 2000$ particles in the APF. Plot (a) depicts the target trajectory on the complex plane and the track obtained by the APF. Plots (b) and (c) show the estimates of the target velocity on the real and imaginary axis, respectively, and plot (d) shows the estimated positions of the 16 sensors (together with the true sensor locations).

In order to assess the average performance of the proposed method, we have also carried out 50 independent computer simulations (each one with a different, and statistically independent, sensor deployment and target trajectory) and computed the mean absolute error (MAE) in the estimation of the target position, its velocity and the sensor locations. The results are presented in Table 2 and illustrate the effectiveness of the APF tracking algorithm. Beware that the resulting estimates may still be rotated with respect to the true trajectory and node locations, although this problem is mitigated when a non-uniform prior pdf $p(s_{1:N_s})$ is available (as it is the case in this example).

5. CONCLUSIONS

We have proposed a sequential Monte Carlo algorithm that enables the joint estimation of the node locations and a maneuvering target track in a region monitored by a sensor network. The algorithm does not require the aid of beacons.

r_t	v_t	s_n
2.8383 m	0.6100 m/s	5.5083 m

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; and the position of a sensor, s_n, $n \in \{1,...,N_s\}$, in m. Simulation parameters: $M = 2000$, $T_s = 0.5$ s, $N_s = 16$.

Instead, it exploits the capability of the sensors to detect the presence of other network nodes within their range during the network startup phase. This information is used to provide a suitable initial guess of the sensor locations. When the *a priori* pdf of the sensors is informative (e.g., non-uniform), it is sufficient to draw M samples from it and weight them according to their likelihood. In more complex scenarios, however, the number of samples required for initialization may be prohibitive and it may be more efficient to use some global optimization method to find sensor positions with high likelihood, as suggested in [8]. After initialization, the proposed auxiliary particle filter recursively provides MAP estimates of the target position and velocity, and improves the estimates of the sensor locations as new observations are collected. The validity of the method has been illustrated by computer simulations of a network of power-aware sensors.

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] 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.

[4] 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.

[5] 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.

[6] M. H. DeGroot and M. J. Schervish, *Probability and Statistics, 3rd ed.*, Addison-Wesley, New York, 2002.

[7] 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, 2001.

[8] J. Míguez and A. Artés-Rodríguez, "A Monte Carlo method for joint node location and maneuvering target tracking in a sensor network," in *Proceedings of the 31st IEEE ICASSP*, May 2006.

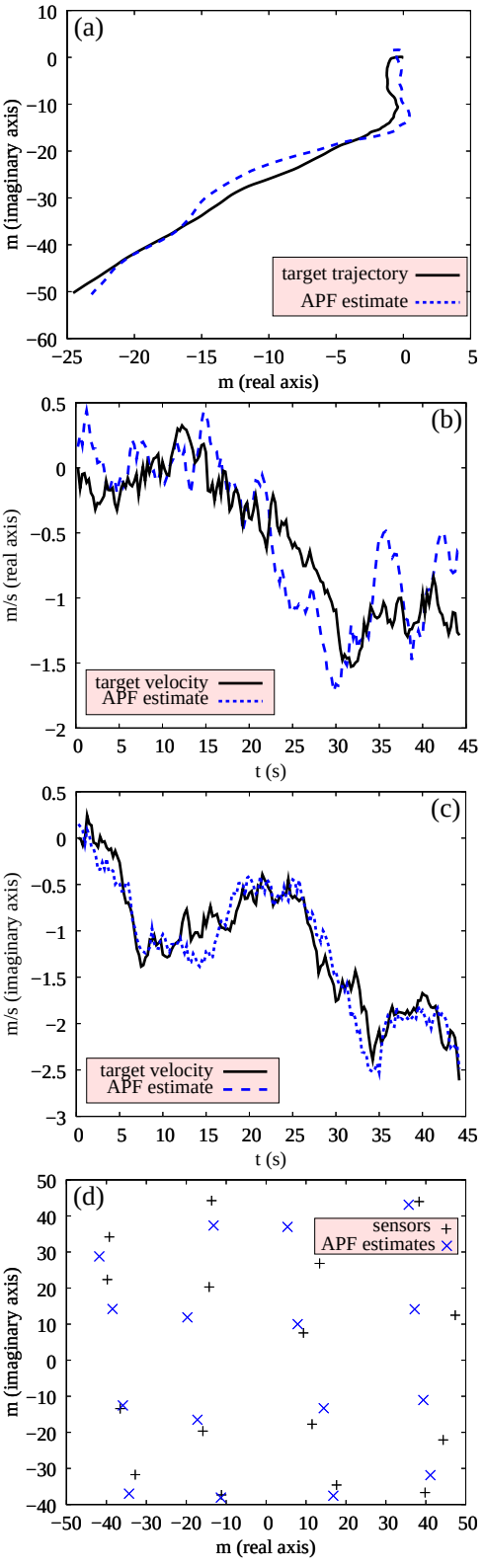

Figure 1: Simulation parameters: $T_s = 0.25$ s, $N_s = 16$ sensors, $M = 2000$ particles, 45 s simulation time. **(a)** Estimate of the target trajectory. **(b)** Estimate of the target velocity on the real axis. **(c)** Estimate of the target velocity on the imaginary axis. **(d)** Estimates of the sensor locations.