Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2008, Article ID 428036, 10 pages
doi:10.1155/2008/428036

Research Article

A Comparison of Detection Performance for Several Track-before-Detect Algorithms

Samuel J. Davey, Mark G. Rutten, and Brian Cheung

Intelligence Surveillance and Reconnaissance Division, Defence Science and Technology Organisation, P.O. Box 1500, Edinburgh, SA 5111, Australia

Correspondence should be addressed to Samuel J. Davey, samuel.davey@dsto.defence.gov.au

Received 30 March 2007; Revised 20 August 2007; Accepted 8 October 2007

Recommended by Yvo Boers

A typical sensor data processing sequence uses a detection algorithm prior to tracking to extract point measurements from the observed sensor data. Track before detect (TBD) is a paradigm which combines target detection and estimation by removing the detection algorithm and supplying the sensor data directly to the tracker. Various different approaches exist for tackling the TBD problem. This article compares the ability of several different approaches to detect low amplitude targets. The following algorithms are considered in this comparison: Bayesian estimation over a discrete grid, dynamic programming, particle filtering methods, and the histogram probabilistic multihypothesis tracker. Algorithms are compared on the basis of detection performance and computation resource requirements.

1. INTRODUCTION

Traditional tracking algorithms are designed assuming that the sensor provides a set of point measurements at each scan. The tracking algorithm links measurements across time and estimates parameters of interest. However, a practical sensor may provide a data image, where each pixel corresponds to the received power in a particular spatial location (e.g., range bins and azimuth beams). In this case, the common approach is to apply a threshold to the data and to treat those cells that exceed the threshold as point measurements, perhaps using interpolation methods to improve accuracy. This is acceptable if the signal-to-noise ratio (SNR) is high. For low SNR targets the threshold must be low to allow sufficient probability of target detection. A low threshold also gives a high rate of false detections which cause the tracker to form false tracks. An alternative approach, referred to as track-before-detect (TBD), is to supply the tracker with all of the sensor data without applying a threshold. This improves track accuracy and allows the tracker to follow low SNR targets.

The main difficulty in the TBD problem is that the measurement, which is the whole sensor image, is a highly nonlinear function of the target state. Typically, the target state describes the kinematic evolution of the target, and may also include its amplitude. However, the sensor provides a map of received scatterer power, which may have a relatively high-intensity response in the location corresponding to the target. One way to deal with this nonlinearity is to discretise the state space. When the state is discrete, then linearity is irrelevant, and estimation techniques such as the hidden Markov model (Baum-Welsh) filter or smoother [1] and the Viterbi algorithm [2] can be applied. Several approaches for TBD have been developed using this method [3–7]. The problem with using a discrete-state space is that it leads to high computation and memory resource requirements.

An alternative to discretising the state is to use a particle filter to solve the nonlinear estimation problem [8–10]. The particle filter uses Monte Carlo techniques to solve the estimation integrals that are analytically intractable. Particle filtering has been used by a number of authors for TBD, for example, [11–13]. It is a numerical approximation technique that uses randomly placed samples instead of fixed samples as is the case for a discretised state space. Particle filtering may be able to achieve similar estimation performance for lower cost by using less sampling points than would be required for a discrete grid.

Another alternative approach is the histogram probabilistic multihypothesis tracker, H-PMHT [14, 15]. A key difference between H-PMHT and the algorithms above is that it uses a parametric representation of the target pdf rather than a numerical one. This reduces the computation load of the algorithm significantly. H-PMHT assumes the superposition of power from the scattering sources and probabilistically associates the received power in each sensor pixel with the target and clutter models. After the association phase it can exploit the estimation algorithms for point measurement tracking. H-PMHT is naturally a multitarget algorithm.

Rather than using the whole sensor image, maximum likelihood probabilistic data association (ML-PDA) reduces the threshold to a low level and then applies a grid-based state model for estimation [16–19]. The association of the high number of measurements is handled using PDA. An alternative version using PMHT for data association has also been used [20]. This algorithm will not be considered in this paper, which instead focuses on algorithms that use the sensor image directly.

In addition to estimating the target state, the TBD algorithm needs to detect the presence or absence of targets. A simple method for this is to extend the state space to include a null state corresponding to the case that there is no target, for example, [6, 11, 21]. In this case, a target is detected when any state other than the null state has the highest probability. A closely related concept is to use a separate Markov chain for the presence or absence of a target as originally introduced for PDA in [22]. This approach has been used for the particle filter [13] and a generalised version was applied to PMHT in [23].

Although there are numerous algorithms for solving the TBD problem, there is currently no TBD benchmark, and existing comparisons between the competing algorithms are limited. The purpose of this article is to compare a number of existing TBD algorithms and to investigate their performance in terms of detection capability, estimation error, and required computation resource. Reference [11] compared the RMS error of a particle-based TBD algorithm with a grid-based Baum-Welsh algorithm. However, that comparison used a single initial target speed (with almost constant velocity) and a single amplitude. A preliminary comparison of the particle filter and H-PMHT algorithms was presented in [24]. This article extends that comparison by including a broader set of algorithms, by using a more realistic measurement model ([24] used Gaussian measurement noise), and by adding diversity in the target behaviour.

This article compares the performance of four TBD algorithms on a radar-like simulation problem as a function of target speed and target amplitude. The target is assumed to be well approximated by a point scatterer, and its contribution to the received sensor image is via a known point-spread function. Although the specific point-spread function used here is the response of Fourier transform windows, problems with extended targets (where the sensor resolution is relatively high) could easily be explored by instead using an appropriate target template, such as in [11].

The algorithms compared are the optimal Bayesian estimator for a discrete-state space, detailed in [6], a Viterbi algorithm, much like that of [4], the particle filter of Rutten et al. [13], and the H-PMHT [14]. The first two algorithms represent maximum a posteriori and maximum likelihood estimation over a fixed grid, the particle filter is a random sampling numerical approximation, and H-PMHT is a parametric approach.

This article is arranged as follows. Section 2 defines the TBD problem, and outlines the target and measurement models used by the various algorithms. Section 3 reviews the different algorithms under test. The performance of the algorithms is investigated via simulations of low SNR targets in Section 4 and Section 5 concludes.

2. PROBLEM DEFINITION

As in [10, Chapter 11], consider a sensor that collects a sequence of two-dimensional images. When present, a target moves in the plane according to a known statistical process. The algorithms use two different kinds of target model: the Bayesian and Viterbi algorithms use a discrete-valued state space, whereas the particle and H-PMHT algorithms use a continuous valued-state space. The true target state, which is used to generate data for simulation analysis, follows the latter model.

2.1. Target model

For simplicity of notation, assume a discrete time model, with a fixed sampling period, T. The target state at time k, $\mathbf{x}_k$, consists of position and velocity in the plane and the intensity of the returned signal, that is,

$$\mathbf{x}_k = \begin{bmatrix} x_k & \dot{x}_k & y_k & \dot{y}_k & I_k \end{bmatrix}^T. \tag{1}$$

The evolution of the state is modelled by the linear stochastic process

$$\mathbf{x}_k = F\mathbf{x}_{k-1} + v_k, \tag{2}$$

where v_k is a noise process and the transition matrix is given by

$$F = \begin{bmatrix} F_s & 0 & 0 \\ 0 & F_s & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad F_s = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}. \tag{3}$$

The noise process is different for the discrete and continuous-valued state models.

2.1.1. Discrete-valued state

Let $\mathcal{X}$ denote the set of all possible states. Assume that the states are uniformly sampled so that

$$\mathbf{x}_k = \begin{bmatrix} \Delta_x q & \dfrac{\Delta_x}{T} r & \Delta_y s & \dfrac{\Delta_y}{T} t & I_k \end{bmatrix}^T, \tag{4}$$

for some integers q, r, s, and t. The algorithms which use the discrete state do not estimate the intensity, but rely on a marginalised likelihood which is described in Section 2.3.

The process noise, v_k, must also belong to $\mathcal{X}$. To reduce the computation overhead, the probability mass function (pmf) of v_k is chosen so that $p(v_k) = 0$ for all v_k outside a tight region centred on the origin. The implementation for this article restricts v_k to a single step in any one dimension (the pmf is a 81 element matrix).

2.1.2. Continuous-valued state

In this case, the noise process is the usual Gaussian random variable with covariance Q given by

$$Q = \begin{bmatrix} Q_s & 0 & 0 \\ 0 & Q_s & 0 \\ 0 & 0 & q_i T \end{bmatrix}, \qquad Q_s = q_s \begin{bmatrix} T^3/3 & T^2/2 \\ T^2/2 & T \end{bmatrix}, \tag{5}$$

where q_s is the power spectral density of the acceleration noise in the spatial dimensions and q_i is the power spectral density of the noise in the rate of change of target return intensity.

2.2. Measurement model

The measurement at each time is a 2-dimensional image consisting of α cells in the x-dimension and β cells in the y-dimension. An example of the data used for simulation in this paper is shown in Figure 1. The measurement in each pixel of the image at time k, $z_k^{(i,j)}$, is assumed to be the magnitude of a windowed complex sinusoid in Gaussian noise, as in [13]. Thus the pixel value will be Ricean distributed if there is a target present, or Rayleigh distributed if there is no target [25]. The measurement pdf is

$$p\left(z_k^{(i,j)} \mid \mathbf{x}_k\right) = \frac{2z_k^{(i,j)}}{\sigma^2} \exp\left(- \frac{\left[z_k^{(i,j)}\right]^2 + h^{(i,j)}(\mathbf{x}_k)^2}{\sigma^2} \right) \times \mathrm{I}_0\left(\frac{2z_k^{(i,j)} h^{(i,j)}(\mathbf{x}_k)}{\sigma^2} \right) \tag{6}$$

if the target is present or

$$p\left(z_k^{(i,j)}\right) = \frac{2z_k^{(i,j)}}{\sigma^2} \exp\left(- \frac{\left[z_k^{(i,j)}\right]^2}{\sigma^2} \right) \tag{7}$$

if there is no target, where σ^2 is the variance of the measurement noise. The term $h^{(i,j)}(\mathbf{x}_k)$ is the contribution in cell i, j from the target, which depends on the point spread function of the windows, the target location, and the target intensity. $\mathrm{I}_0(\cdot)$ is the modified Bessel function.

The complete measurement at time k is denoted by $z_k = \{z_k^{(i,j)} \mid i = 1,\ldots,\alpha,\ j = 1,\ldots,\beta\}$ and the set of all measurements up to time k is denoted by $Z_k = \{z_l \mid l = 1,\ldots,k\}$.

The target peak SNR quantifies the height of the peak of the target point spread function relative to the noise floor, and represents a measure of how easy it is to detect the target. The point spread function is assumed to be normalised such that the contribution to cell i, j is I_k when the target is located exactly on the sample point for the cell. Thus the peak SNR in dB is given by $20\log\{I_k/\sigma^2\}$.

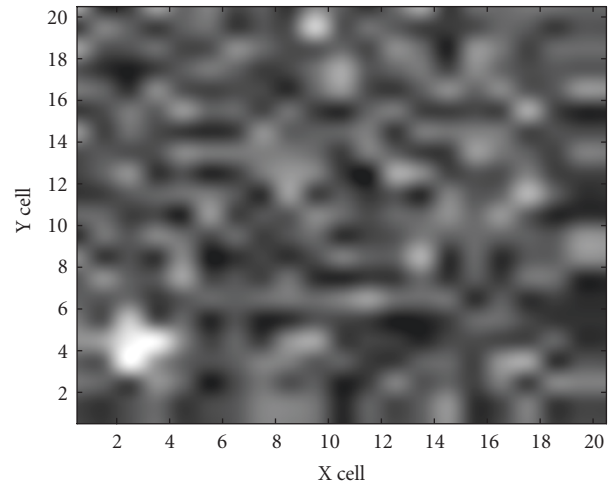

Figure 1: Simulated measurement data with a 12 dB target return at $x = 3.5$ and $y = 3.25$.

2.3. Likelihood ratio

In many cases, it is more convenient to deal with the likelihood ratio of the data, rather than the measurement pdf. For the measurement model described above, the likelihood ratio for cell (i, j) is

$$\mathcal{L}\left(z_k^{(i,j)} \mid \mathbf{x}_k\right) \equiv \frac{p\left(z_k^{(i,j)} \mid \mathbf{x}_k\right)}{p\left(z_k^{(i,j)}\right)} = \exp\left(\frac{-h^{(i,j)}(\mathbf{x}_k)^2}{\sigma^2} \right) \mathrm{I}_0\left(\frac{2z_k^{(i,j)} h^{(i,j)}(\mathbf{x}_k)}{\sigma^2} \right). \tag{8}$$

Since the pixels are assumed to be conditionally independent, the likelihood of the whole image is simply the product over the pixels

$$\mathcal{L}(z_k \mid \mathbf{x}_k) = \prod_{i=1}^{\alpha} \prod_{j=1}^{\beta} \mathcal{L}\left(z_k^{(i,j)} \mid \mathbf{x}_k\right). \tag{9}$$

If a prior distribution is assumed for the target intensity, $p(I_k)$, then an intensity independent marginal likelihood is given by

$$\overline{\mathcal{L}}(z_k \mid \mathbf{x}_k) = \int_0^{\infty} \mathcal{L}(z_k \mid \mathbf{x}_k)\, p(I_k)\, dI_k. \tag{10}$$

This integral can be approximated by a summation.

3. ALGORITHMS

3.1. Bayesian estimator

The posterior pdf of the target state can be recursively determined using the well-known Bayesian relationship

$$p(\mathbf{x}_k \mid Z_k) \propto p(z_k \mid \mathbf{x}_k) \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1})\, p(\mathbf{x}_{k-1} \mid Z_{k-1})\, d\mathbf{x}_{k-1}. \tag{11}$$

The Bayesian estimator in this paper is a direct approximation to (11) based on a discretisation of the state space. Choose a uniformly spaced set of states, $\mathcal{X}$ (which is not necessarily related to the discrete measurement function). Equation (11) can then be approximated by

$$p(\mathbf{x}_k \mid Z_k) \approx K\overline{\mathcal{L}}(z_k \mid \mathbf{x}_k) \sum_{\mathbf{x}_{k-1}\in\mathcal{X}} p(\mathbf{x}_k \mid \mathbf{x}_{k-1})\, p(\mathbf{x}_{k-1} \mid Z_{k-1}), \tag{12}$$

where K is a normalising constant. The approximation is exact in the limit as $\mathcal{X}$ approaches $\Re^4$. The first term in (12) is the intensity independent marginal likelihood, defined by (10).

The discrete-state space is augmented with a null state, $\varnothing$, to indicate the possibility that there is no target. Denote the probability of target *death* as P_d, and the probability of target *birth* as P_b. Then the evolution probability in (12) is given by

$$p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) = \begin{cases} 1 - P_b, & \mathbf{x}_k = \varnothing,\ \mathbf{x}_{k-1} = \varnothing, \\ P_d, & \mathbf{x}_k = \varnothing,\ \mathbf{x}_{k-1} \neq \varnothing, \\ P_b/|\mathcal{X}|, & \mathbf{x}_k \neq \varnothing,\ \mathbf{x}_{k-1} = \varnothing, \\ (1 - P_d) & \mathbf{x}_k \neq \varnothing,\ \mathbf{x}_{k-1} \neq \varnothing, \\ \times p(v_k = \mathbf{x}_k - F\mathbf{x}_{k-1}), & \end{cases} \tag{13}$$

where $|\mathcal{X}|$ is the number of discrete states in $\mathcal{X}$.

The parameters P_b and P_d control the detection performance and can be tuned to optimise detection performance. The selection of the state space, $\mathcal{X}$, is a tradeoff between estimation accuracy, which improves with finer resolution, and computation requirement, which increases with $|\mathcal{X}|$. The process noise pmf also affects estimation accuracy, as well as providing some capacity to handle model mismatch between the assumed target model and the true target motion. The algorithm is initialised with $p(\mathbf{x}_0 = \varnothing) = 1$ and $p(\mathbf{x}_0) = 0$ for all $\mathbf{x}_0 \neq \varnothing$.

Once the pdf of the state has been evaluated, a state estimate can be obtained by selecting the state with the highest probability. In the event that this state is the null state, then the algorithm reports that there is no target. To account for the case where the pdf has a peak that is spread over several grid cells, the implementation used in this article finds the highest probability nonnull state and accumulates the probability in the adjacent cells. If the accumulated probability is higher than the null-state probability, then a detection is reported.

3.2. Dynamic programming

The Bayesian algorithm above is a MAP estimator. It recursively defines the probability of the target occupying a particular location by the superposition of all of the possible paths to that position. An alternative is to use a maximum likelihood (ML) estimator. Rather than accumulate the probability from alternate paths, an ML estimator selects the single best path. An ML algorithm for discrete states is the Viterbi algorithm, which has been applied to TBD in [4]. The Viterbi algorithm is a batch processor that finds the most likely sequence of states. One advantage of this is that it always produces an estimate consistent with the dynamic model: if the transition model gives probability zero to the transition from state 1 to state 3, then the Viterbi estimate will never contain a transition from 1 to 3. In contrast, a MAP estimate may contain such a transition. The next algorithm considered is the Viterbi algorithm. The main difference between the algorithm used here and that of Barniv is the extension of the state space to include the possibility that there is no target.

The joint posterior probability of the sequence of states $\mathbf{x}_0 \cdots \mathbf{x}_k$ is given by

$$p(\mathbf{x}_0 \cdots \mathbf{x}_k \mid Z_k) \propto p(\mathbf{x}_0) \prod_{t=1}^{k} \overline{\mathcal{L}}(z_t \mid \mathbf{x}_t)\, p(\mathbf{x}_t \mid \mathbf{x}_{t-1}). \tag{14}$$

The Viterbi algorithm is a recursive scheme for maximising the joint pdf above which has linear complexity in time (unlike the size of the joint-state space, which is $|\mathcal{X}|^{k+1}$). Let $C_k(\mathbf{x}_k)$ denote the Viterbi cost metric, which is a normalised measure of the log-likelihood of the most likely sequence leading into state $\mathbf{x}_k$. As for the Bayesian algorithm, $\mathbf{x}_k = \varnothing$ denotes the case that there is no target. The previous state in the most likely sequence leading into state $\mathbf{x}_k$ is denoted $\theta_{k-1}(\mathbf{x}_k)$. The algorithm proceeds as follows.

(1) Initialise $C_0(\varnothing) = 0$ and $C_0(\mathbf{x}_0) = -\infty$ for all other states.

(2) For each scan $k = 1 \cdots k_{\max}$, the unnormalised cost of the null state, c_k^0, is given by

$$c_k^0 = \max_{\mathbf{x}_{k-1}} \{C_{k-1}(\mathbf{x}_{k-1}) + \log p(\varnothing \mid \mathbf{x}_{k-1})\}, \tag{15}$$

which is used to define the normalised cost for all states

$$C_k(\mathbf{x}_k) = \log \overline{\mathcal{L}}(z_t \mid \mathbf{x}_t) + \max_{\mathbf{x}_{k-1}} \{C_{k-1}(\mathbf{x}_{k-1}) + \log p(\mathbf{x}_k \mid \mathbf{x}_{k-1})\} - c_k^0. \tag{16}$$

The previous state in the most likely sequence leading to $\mathbf{x}_k$ is given by

$$\theta_{k-1}(\mathbf{x}_k) = \arg\max \{C_{k-1}(\mathbf{x}_{k-1}) + \log p(\mathbf{x}_k \mid \mathbf{x}_{k-1})\}. \tag{17}$$

(3) The estimated state sequence is found by backtracking

$$\hat{\mathbf{x}}_k = \begin{cases} \arg\max C_k(\mathbf{x}_k), & k = k_{\max}, \\ \theta_k(\hat{\mathbf{x}}_{k+1}), & \text{otherwise.} \end{cases} \tag{18}$$

As for the Bayesian algorithm above, the discrete state grids, P_d and P_b, are tuning parameters. The span of the state space includes a null state, so the algorithm reports that no target is present if the estimated state in (18) is the null state.

3.3. Particle filter

The particle filter used in this paper is based on the algorithm derived in detail in [13]. Like the grid methods above, the

state space is augmented with a null state to allow for automatic track initiation. This algorithm uses terminology similar to that used for target detection with the probabilistic data association filter [22, 26]. That is, a binary *existence* variable, E_k, is defined such that $E_k = 1 \rightarrow \mathbf{x}_k \neq \varnothing$ and $E_k = 0 \rightarrow \mathbf{x}_k = \varnothing$. The algorithm makes a direct approximation of the target-state posterior $p(\mathbf{x}_k \mid E_k = 1, Z_k)$ and the existence probability $p(E_k \mid Z_k)$. Reference [27] demonstrated that this model is significantly more efficient than extending the state vector with a binary existence state.

The algorithm proceeds by constructing two sets of particles. The first set, the birth particles, estimates $p(\mathbf{x}_k \mid E_k = 1, E_{k-1} = 0, Z_k)$, that is the case where the target did not exist in the data at time $k-1$ but does at time k. The second set, the continuing particles, estimates $p(\mathbf{x}_k \mid E_k = 1, E_{k-1} = 1, Z_k)$, which is the case where the target has continued to exist in the data from time $k-1$ to k. Starting with a set of N_c particles $\{\mathbf{x}_{k-1}^i \mid i = 1 \cdots N_c\}$ describing the posterior target state at time $k-1$ and an estimate of the probability of existence at time $k-1$, $\hat{P}_{k-1}$, the algorithm consists of the following steps.

(1) Create a set of N_b birth particles by placing the particles in the highest intensity cells [27]

$$\mathbf{x}_k^{(b)i} \sim q\left(\mathbf{x}_k \mid \mathbf{x}_{k-1} = \varnothing, z_k\right). \tag{19}$$

The unnormalised birth particle weights are calculated using the likelihood ratio (9) and proposal density (19)

$$\widetilde{w}_k^{(b)i} = \frac{\mathcal{L}\left(z_k \mid \mathbf{x}_k^{(b)i}\right) p\left(\mathbf{x}_k^{(b)i} \mid \mathbf{x}_{k-1} = \varnothing\right)}{N_b q\left(\mathbf{x}_k^{(b)i} \mid \mathbf{x}_{k-1} = \varnothing, z_k\right)}. \tag{20}$$

(2) Create a set of N_c continuing particles using the system dynamics (2) as the proposal function, with weights

$$\widetilde{w}_k^{(c)i} = \frac{1}{N_c} \mathcal{L}\left(z_k \mid \mathbf{x}_k^{(c)i}\right). \tag{21}$$

(3) The mixing probabilities are calculated using sums of unnormalised weights

$$\begin{aligned} \widetilde{M}_b &= P_b\left[1 - \hat{P}_{k-1}\right] \sum_{i=1}^{N_b} \widetilde{w}_k^{(b)i}, \\ \widetilde{M}_c &= \left[1 - P_d\right] \hat{P}_{k-1} \sum_{i=1}^{N_c} \widetilde{w}_k^{(c)i}. \end{aligned} \tag{22}$$

(4) The probability of existence at time k can also be calculated in terms of unnormalised weights

$$\hat{P}_k = \frac{\widetilde{M}_b + \widetilde{M}_c}{\widetilde{M}_b + \widetilde{M}_c + P_d \hat{P}_{k-1} + \left[1 - P_b\right]\left[1 - \hat{P}_{k-1}\right]}. \tag{23}$$

(5) The particle weights are normalised

$$\begin{aligned} \hat{w}_k^{(b)i} &= \frac{P_b\left[1 - \hat{P}_{k-1}\right]}{\widetilde{M}_b + \widetilde{M}_c} \widetilde{w}_k^{(b)i}, \\ \hat{w}_k^{(c)i} &= \frac{\left[1 - P_d\right] \hat{P}_{k-1}}{\widetilde{M}_b + \widetilde{M}_c} \widetilde{w}_k^{(c)i}. \end{aligned} \tag{24}$$

The two sets of particles can then be combined into one large set

$$\left\{ \left(\mathbf{x}_k^{(t)i}, \hat{w}_k^{(t)i} \right) \mid i = 1, \ldots, N_t,\ t = c, b \right\}. \tag{25}$$

(6) Resample from $N_b + N_c$ down to N_c particles.

Thus after completing the above steps the particles, $\{\mathbf{x}_k^i \mid i = 1 \cdots N_c\}$, with uniform weights, approximate the posterior target state density at time k, and $\hat{P}_k$ is an estimate of the probability of target existence.

The algorithm declares a target detected when the existence probability, that is, $1 - p(\mathbf{x}_k = \varnothing)$, is above a tunable threshold. The state estimate is then found by taking the mean of the state vectors for each particle.

3.4. Histogram PMHT

The algorithms described so far are general numerical approximation techniques applied to the TBD problem. The final algorithm is an approach specifically developed for TBD. H-PMHT is derived by interpreting the sensor image as a histogram of observations of an underlying random process. The received energy in each cell is quantised, and the resulting integer is treated as a count of the number of measurements that fell within that cell. The sum over all of the cells is the total number of measurements taken. The probability mass function for these discrete measurements is modelled as a multinomial distribution where the probability mass for each cell is the superposition of target and noise contributions. The H-PMHT data association process probabilistically assigns each individual quantum to the target and noise models. For each model, the individual quanta and their assignment weights are combined to form a single *synthetic* measurement and measurement covariance. These are then used by a point-measurement-based estimator. For the special case of linear Gaussian statistics, the synthetic measurement is formed using a weighted arithmetic mean and a Kalman filter can be used as the estimator. The quantisation is an artificial process, and is removed by taking the limit as the quantisation step size approaches zero.

The H-PMHT measurement model is slightly different to the model in Section 2.2. Whereas Section 2.2 explicitly represents the target amplitude, H-PMHT uses a relative power representation. In the H-PMHT model, the mean cell value is given by

$$P_k^{(i,j)} = \sum_{m=0}^{M} \pi_{km} P_{km}^{(i,j)}, \tag{26}$$

where π_{km} is the mixing proportion for model m at time k. Model 0 quantifies the noise contribution, and there are M targets. The cell contribution of model m, $P_{km}^{(i,j)}$, is the integral of the model measurement pdf over cell (i, j). The noise is spatially uniform, so $P_{k0}^{(i,j)} = (\alpha\beta)^{-1}$, that is, one over the number of cells. The target contribution is approximated as a normal density function with variance Σ^2 in both X and Y, that is, $P_{km}^{(i,j)} = N(i; x_k, \Sigma^2) N(j; y_k, \Sigma^2)$.

Existing tracks are updated using a recursive implementation of the H-PMHT algorithm. H-PMHT is an iterative algorithm which alternates between data association and state estimation. The state and mixing proportion estimates at the pth iteration are denoted by $\hat{\mathbf{x}}_{km}^{(p)}$ and $\hat{\pi}_{km}^{(p)}$, respectively. The algorithm is summarised as follows.

(1) Initialise estimates

$$\begin{aligned}\hat{\mathbf{x}}_{km}^{(0)} &= F\hat{\mathbf{x}}_{(k-1)m},\\ \hat{\pi}_{km}^{(0)} &= \hat{\pi}_{(k-1)m}.\end{aligned} \tag{27}$$

(2) Calculate cell probabilities, $P_{km}^{(i,j)}$ and $P_k^{(i,j)}$.

(3) Calculate cell-centroid, $\tilde{z}_{km}^{(i,j)} = [\tilde{x}_{km}^{(i,j)}, \tilde{y}_{km}^{(i,j)}]^T$, with

$$\begin{aligned}\tilde{x}_{km}^{(i,j)} &= \hat{x}_{km}^{(p-1)} + \Sigma^2 N\left(i-\frac{1}{2}; \hat{x}_{km}^{(p-1)}, \Sigma^2\right)\\ &\quad - \Sigma^2 N\left(i+\frac{1}{2}; \hat{x}_{km}^{(p-1)}, \Sigma^2\right),\\ \tilde{y}_{km}^{(i,j)} &= \hat{y}_{km}^{(p-1)} + \Sigma^2 N\left(j-\frac{1}{2}; \hat{y}_{km}^{(p-1)}, \Sigma^2\right)\\ &\quad - \Sigma^2 N\left(j+\frac{1}{2}; \hat{y}_{km}^{(p-1)}, \Sigma^2\right).\end{aligned} \tag{28}$$

(4) Determine synthetic measurements and synthetic measurement covariances (where I is the identity matrix)

$$\begin{aligned}\tilde{z}_{km}^{(p)} &= \frac{\sum_i \sum_j \left[\bar{z}_k^{(i,j)}\left(P_{km}^{(i,j)}/P_k^{(i,j)}\right)\right]\tilde{z}_{km}^{(i,j)}}{\sum_i \sum_j \left[\bar{z}_k^{(i,j)}\left(P_{km}^{(i,j)}/P_k^{(i,j)}\right)\right]},\\ \tilde{R}_{km}^{(p)} &= \frac{\Sigma^2}{\hat{\pi}_{km}^{(p-1)} \sum_i \sum_j \bar{z}_k^{(i,j)}\left(P_{km}^{(i,j)}/P_k^{(i,j)}\right)} I.\end{aligned} \tag{29}$$

(5) Estimate mixing proportions

$$\hat{\pi}_{tm}^{(p)} = \frac{\hat{\pi}_{km}^{(p-1)} \sum_i \sum_j \bar{z}_k^{(i,j)}\left(P_{km}^{(i,j)}/P_k^{(i,j)}\right)}{\sum_{l=0}^{M} \hat{\pi}_{kl}^{(p-1)} \sum_i \sum_j \bar{z}_k^{(i,j)}\left(P_{kl}^{(i,j)}/P_k^{(i,j)}\right)}. \tag{30}$$

(6) Estimate states using Kalman filters, the synthetic measurements, and covariances.

(7) Repeat 2–6 until convergence.

(8) Estimate intensity

$$\hat{I}_{km} = \frac{\hat{\pi}_{km} \sum_i \sum_j \bar{z}_k^{(i,j)}}{\hat{\pi}_{k0}}. \tag{31}$$

Note that the theory demands that the convergence test be based on the expectation-maximisation auxiliary function associated with the algorithm (see [14]). However, in practice this function is costly to evaluate and only required for the convergence test. Instead it is more practical to test for convergence based on the estimates themselves. In the implementation used for this paper, convergence is tested by measuring the change in state estimates from one iteration to the next.

The H-PMHT algorithm described above updates existing tracks, but does not provide a means for initiating new tracks or terminating old tracks. A typical two-stage approach based on the method in [15] is used for this function. The tracker maintains two sets of tracks: established tracks, that the tracker is confident corresponding to targets, and tentative tracks, that the tracker is not confident in. The established tracks are updated first, and they vet the sensor data before it is presented to the tentative tracks. Similarly, the tentative tracks vet the data before it is passed to a new tentative track initiation stage. Established tracks are terminated when the estimated intensity drops below a threshold of −10 dB for two consecutive scans. Tentative tracks are terminated when the estimated intensity drops below 0 dB for two consecutive scans. Tentative tracks are promoted to established tracks if the estimated intensity is greater than 0 dB for more than three scans.

The tracks vet the sensor data following the method proposed in [15]. This is done by scaling each cell based on the tracker model

$$z_k'^{(i,j)} = z_k^{(i,j)} \frac{1}{\alpha\beta P_k^{(i,j)}}. \tag{32}$$

The result of the vetting process is a sensor image that is suppressed in the location of the existing tracks, but unchanged in other regions. New tentative tracks are formed by finding peaks in the vetted data. When there are peaks within a threshold distance in two consecutive scans, a new tentative track is formed.

3.5. Algorithm tuning

Each of the algorithms has a number of different parameters which need to be selected to ensure good performance. It is of interest to characterise how algorithm performance varies with these parameters. However, it is impractical to investigate these characteristics in this article. For this article, each algorithm has been tuned to give approximately the same false alarm performance, and effort was made to optimise the algorithm code for speed. A more detailed analysis of algorithm performance as a function of various parameters is currently being undertaken by the authors.

4. DETECTION PERFORMANCE

The performance of the various algorithms was investigated by simulating a scenario with a single target. Since this study is concerned with detection performance, only straight line targets were considered. Various scenarios were constructed by selecting a particular target speed and intensity. Each scenario contained 20 scans. Table 1 summarises the different parameters considered. For each scenario, one hundred Monte Carlo trials were performed.

The algorithms which sample the state space on a fixed grid may be affected by the position of the target relative to the grid, that is, whether the target is close to a grid point or mid way between them. In order to average over this potential variation, the initial target position for each Monte Carlo trial was randomly sampled from the range 2.5–3 independently in X and Y. The target heading was also randomly

Table 1: Scenario parameters.

Scenario	1	2	3	4	5	6	7	8	9	10	11	12
speed (pixels/frame)	0.25	0.5	1	2	0.25	0.5	1	2	0.25	0.5	1	2
SNR (dB)	12	12	12	12	6	6	6	6	3	3	3	3

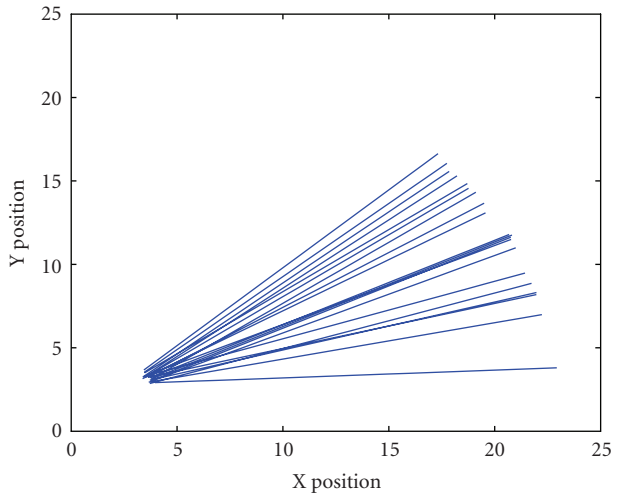

Figure 2: Example scenario, target speed = 1.

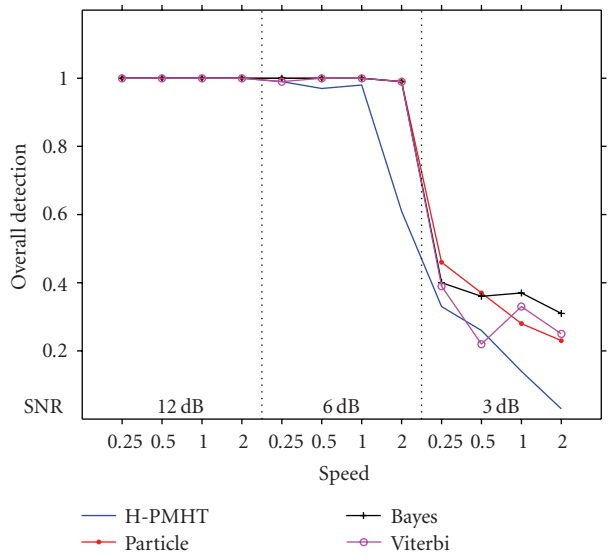

Figure 3: Overall detection probability.

sampled from 0 degrees (East) to 45 degrees (North East). Figure 2 shows an example of 20 Monte Carlo trials with a speed of 1.

False track performance was quantified using a single realisation of a scenario with no target present and 2000 scans. The long duration was chosen to test whether the particle filter algorithm suffered from degeneracy.

Six metrics were used to measure performance as follows.

(1) *Overall detection probability* was defined as the fraction of Monte Carlo trials for which the target was detected*at any time*.

(2) *Instantaneous detection probability* was defined as the total fraction of frames for which the target was detected.

(3) *RMS position error* was averaged over those frames when the target was detected.

(4) *False track count* was the number of false tracks formed in the no-target scenario.

(5) *False track length* was the average number of frames for which these false tracks persist.

(6) *Computation resource* was the total CPU time required to evaluate all of the scenarios. This figure was recorded both in seconds, and as a ratio compared with the fastest algorithm.

The overall detection probability is shown in Figure 3, the instantaneous detection probability in Figure 4, and the RMS position error in Figure 5. In each of the figures, the metric is plotted as a function of scenario number. The horizontal (scenario) axis has two labels: the target speed for the scenario is shown below the axis, and the SNR for the scenario is shown above the axis. Vertical dotted lines delineate the scenarios with a particular SNR value.

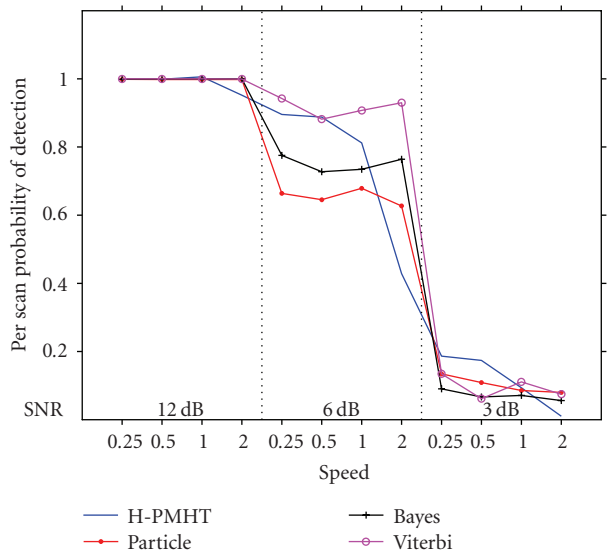

Figure 4: Average instantaneous detection probability.

Table 2 shows the false track count and the computation resource. For comparison, a probabilistic data association filter with amplitude information (PDAF-AI) [28–30] was also run on the false track scenario. PDAF-AI uses point measurements and includes amplitude as a nonkinematic measurement feature. The PDAF-AI algorithm was run assuming

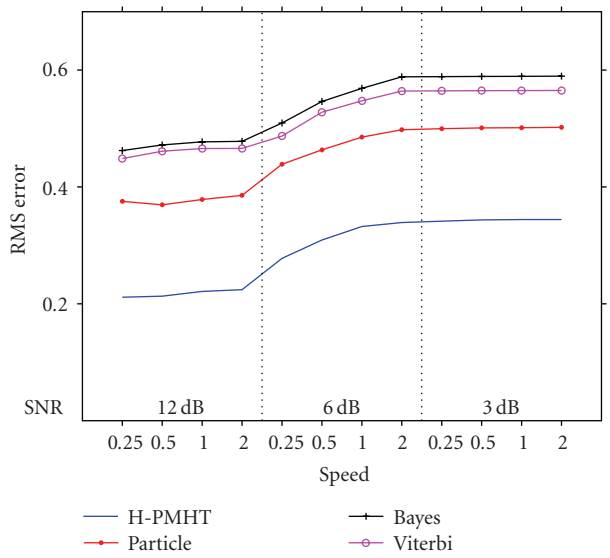

Figure 5: RMS position estimation error.

a known target SNR and a detection threshold to give ninety percent probability of detection for that SNR. The false track performance at the SNR values of interest is shown in Table 2. The false track performance of the PDAF-AI is clearly unacceptable below 12 dB. For the 3 dB case, the rate of false tracks is lower because the false tracks persist for much longer. The other performance metrics were not considered for PDAF-AI since the false track performance was so poor.

All of the TBD algorithms were able to easily detect targets at 12 dB, and they all detected every target. The H-PMHT had a small delay in detecting some of the high speed (2 pixels per frame) targets. These targets proved to be difficult for the H-PMHT algorithm because of the track initialisation method that it used. The algorithm formed tentative tracks and then rejected or accepted the tentative tracks based on the estimated power of the track. For this strategy to work, the track formation logic must reliably form tentative tracks close to the true target state. Two consecutive measurements were used to initialise the tentative tracks, and the approach gave poor speed initialisation. This degraded the H-PMHT's ability to detect high-speed targets.

For the 6 dB targets, in the centre region of Figures 3 and 4, the numerical approximation techniques continued to detect almost all of the targets, but the H-PMHT only detected about half of the high speed targets. The Viterbi algorithm gave the best instantaneous detection result because it was allowed to back-track. The H-PMHT also gave high instantaneous detection for the lower speed cases because the tentative track history was included.

At 3 dB, all of the algorithms showed degraded detection performance and found less than half of the targets. Again, the H-PMHT performed poorly for high speed targets. It may be counterintuitive that the Viterbi algorithm performed generally worse than the other numerical approximations. However, the smoothing it performs does not improve overall detection performance. It improves state estimation when the target is detected and allows the estimator to back-track once the target has been detected, but it does not increase the overall number of targets that are found.

The false track performance of all of the algorithms is comparable, since this was a requirement of the algorithm tuning. Thus, the overall conclusion is that the numerical approximation techniques considered give similar detection performance. If batch processing is acceptable, then back-tracking can provide some improvement in the percentage of time that a target is tracked. Note that for batch processing, the Bayesian filter used here could be replaced with a Baum-Welsh such as in [11]. The H-PMHT gave worse performance than the numerical approaches when the target had high speed. However, this may be alleviated with an improved initialisation scheme.

The RMS estimation error curves shown in Figure 5 shows an obvious trend. The error increases as the target speed increases and as the amplitude is reduced. For all cases, the H-PMHT error is significantly lower than particle filter error, which is in turn better than the grid approximations. The smoothing used in the Viterbi algorithm reduced the RMS error a little over the Bayesian filter. As may be expected, the error from the grid-based algorithms was approximately half the grid size.

The computation resource required by the different algorithms shows a more marked difference than the detection performance. As Table 2 shows, the H-PMHT was more than an order of magnitude faster than the particle filter, and more than two orders of magnitude faster than the grid-based algorithms. Significant effort was spent in optimising all of the algorithms. For the H-PMHT, there was no specific computation bottleneck: both the association and filtering codes took similar resource. In contrast, the numerical approximation algorithms were all limited by the likelihood calculations. These incurred the vast majority of the processing cost, even though they used external library code. The H-PMHT was much faster because it does not perform likelihood computations.

In summary, the H-PMHT gave slightly worse detection results than the other algorithms, but at a fraction of the computation cost. Given that H-PMHT is already a multi-target algorithm, whereas the others assume only a single target, the results here suggest that a robust initialisation scheme would make it the algorithm of choice.

The results presented in this article have not addressed some important issues. In particular, the grid spacing for the Bayesian filter and the Viterbi algorithm was fixed at the sensor resolution. Whether this is the best choice was not thoroughly investigated. However, anecdotal trials demonstrated that doubling the grid resolution gave little improvement in detection performance and marginal improvement in RMS estimation error for four times the computation cost. Given the already high cost for these algorithms, it was decided not to pursue finer resolution at this stage. The number of particles in the particle filter is also a parameter that may be varied. This was not explored in this comparison since earlier work has demonstrated that reducing the number of particles degrades performance too much [24].

Table 2: Algorithm performance.

Algorithm	False number	False length	cpu time (s)	cpu time (ratio)
H-PMHT	0	*	420	1
particle	0	*	16600	39
Bayes	0	*	85600	204
Viterbi	11	2.27	88000	210
PDAF-AI (12 dB)	0	*	*	*
PDAF-AI (6 dB)	1412	145	*	*
PDAF-AI (3 dB)	1028	185	*	*

5. CONCLUSION

The detection performance of four alternative track-before-detect algorithms has been investigated for a range of target SNR values and speeds. For most of the scenarios the difference in detection performance was minor, except for targets with high speed, which were not tracked well by H-PMHT. Both of the grid-based algorithms were very costly in terms of computation resource, whereas the particle filter was significantly faster. The H-PMHT was by far the fastest, requiring two orders of magnitude less computation resource than the grid-based algorithms. The difference in computation cost is largely due to the cost of calculating the likelihood ratio of the data: H-PMHT does not use the likelihood ratio, and the particle filter calculates it over fewer points than the grid algorithms.

The comparison also considered the RMS position estimation error of the algorithms, for which H-PMHT was clearly the best. The two grid-based algorithms had an RMS error approximately double that of H-PMHT, and the particle filter RMS error was part-way between.

The scenarios considered used only a single target, whereas many practical situations required the detection of multiple targets. The H-PMHT is a multitarget algorithm, so is already capable of handling such a problem, but the other algorithms require extension to the multitarget problem. Future work is planned to consider multitarget detection and tracking.

REFERENCES

[1] L. Rabiner and B. Jang, "An introduction to hidden markov models," *IEEE ASSP Magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[2] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[3] S. C. Pohlig, "An algorithm for detection of moving optical targets," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 25, no. 1, pp. 56–63, 1989.

[4] Y. Barniv, "Dynamic programming algorithm for detecting dim moving targets," in *Multitarget-Multisensor Tracking: Advanced Applications*, Y. Bar-Shalom, Ed., chapter 4, Artech House, Norwood, Mass, USA, 1990.

[5] S. M. Tonbsen and Y. Bar-Shalom, "Maximum likelihood track-before-detect with fluctuating target amplitude," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, no. 3, pp. 796–809, 1998.

[6] L. D. Stone, C. A. Barlow, and T. L. Corwin, *Bayesian Multiple Target Tracking*, Artech House, Norwood, Mass, USA, 1999.

[7] M. G. S. Bruno and J. M. F. Moura, "Multiframe detector/tracker: optimal performance," *Transactions on Aerospace and Electronic Systems*, vol. 37, no. 3, pp. 925–945, 2001.

[8] D. J. Salmond and H. Birch, "A particle filter for track-before-detect," in *In Proceedings of the American Control Conference*, vol. 5, pp. 3755–3760, Arlington, Va, USA, June 2001.

[9] Y. Boers and J. N. Driessen, "Particle filter based detection for tracking," in *Proceedings of the American Control Conference*, pp. 4393–4397, Arlington, Va, USA, June 2001.

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

[11] M. G. S. Bruno, "Bayesian methods for multiaspect target tracking in image sequences," *IEEE Transactions on Signal Processing*, vol. 52, no. 7, pp. 1848–1861, 2004.

[12] H. Driessen and Y. Boers, "An efficient particle filter for nonlinear jump Markov systems," in *Proceedings of the IEE Seminar on Target Tracking: Algorithms and Applications*, Sussex, UK, March 2004.

[13] M. G. Rutten, N. J. Gordon, and S. Maskell, "Recursive track-before-detect with target amplitude fluctuations," *IEE Proceedings on Radar, Sonar and Navigation*, vol. 152, no. 5, pp. 345–322, 2005.

[14] R. L. Streit, "Tracking on intensity-modulated data streams," Tech. Rep. 11221, NUWC, Newport, RI, USA, May 2000.

[15] R. L. Streit, M. L. Graham, and M. J. Walsh, "Multitarget tracking of distributed targets using histogram-PMHT," *Digital Signal Processing*, vol. 12, no. 2-3, pp. 394–404, 2002.

[16] C. Jauffret and Y. Bar-Shalom, "Track formation with bearing and frequency measurements in clutter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 26, no. 6, pp. 999–1010, 1990.

[17] T. Kirubarajan and Y. Bar-Shalom, "Low observable target motion analysis using amplitude information," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 32, no. 4, pp. 1367–1384, 1996.

[18] W. R. Blanding, P. K. Willett, and Y. Bar-Shalom, "Off-line and real-time methods for ML-PDA track validation," *IEEE Transactions on Signal Processing*, vol. 55, no. 5, pp. 1994–2006, 2007.

[19] W. R. Blanding, P. K. Willett, Y. Bar-Shalom, and R. S. Lynch, "Directed subspace search ML-PDA with application to active sonar tracking," to appear in *IEEE Transactions on Aerospace and Electronic Systems*.

[20] P. Willett and S. Coraluppi, "MLPDA and MLPMHT applied to some MSTWG data," in *Proceedings of the 9th International Conference on Information Fusion*, July 2006.

[21] R. L. Streit and R. F. Barrett, "Frequency line tracking using hidden markov models," *IEEE Transactions on Acoustics, Speach, and Signal Processing*, vol. 38, no. 4, pp. 586–598, 1990.

[22] S. B. Colegrove, A. W. Davis, and J. K. Ayliffe, "Track initiation and nearest neighbours incorporated into probabilistic data association," *Journal of Electrical and Electronics Engineers, Australia*, vol. 6, no. 3, pp. 191–198, 1986.

[23] S. J. Davey and D. A. Gray, "Integrated track maintenance for the pmht via the hysteresis model," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 1, pp. 93–111, 2007.

[24] S. J. Davey and M. G. Rutten, "A comparison of three algorithms for tracking dim targets," in *Conference Proceedings of Information, Decision, and Control (IDC '07)*, pp. 342–347, Adelaide, Australia, February 2007.

[25] M. I. Skolnik, *Introduction to Radar Systems*, McGraw-Hill, 3rd edition, 2001.

[26] D. Mušicki, R. Evans, and S. Stankovic, "Integrated probabilistic data association," *IEEE Transactions on Automatic Control*, vol. 39, no. 6, pp. 1237–1240, 1994.

[27] M. G. Rutten, B. Ristic, and N. J. Gordon, "A comparison of particle filters for recursive track-before-detect," in *Proceedings of the 8th International Conference on Information Fusion*, vol. 1, pp. 169–175, Philadelphia, Pa, USA, July 2005.

[28] D. Lerro and Y. Bar-Shalom, "Automatic track formation with target amplitude information," in *Proceedings of the Oceans Conference Record (OCEANS '91)*, vol. 3, pp. 1460–1467, October 1991.

[29] D. Lerro and Y. Bar-Shalom, "Comparison of tracking/association methods for low SNR targets," in *Proceedings of the Oceans Conference Record (OCEANS '92)*, pp. 443–448, October 1992.

[30] D. Lerro and Y. Bar-Shalom, "Interacting multiple model tracking with target amplitude feature," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 29, no. 2, pp. 494–509, 1993.