Detecting a Small Boat using Histogram PMHT

SAMUEL J. DAVEY

The Histogram Probabilistic Multi-Hypothesis Tracker (H-PMHT) is a parametric track before detect algorithm. This article discusses practical issues faced in producing an operationally relevant implementation of the algorithm. The performance of H-PMHT is demonstrated on a high resolution radar example containing a small boat in challenging clutter and compared with a conventional solution based on image segmentation and the Probabilistic Data Association Filter.

Manuscript received November 21, 2010; revised June 19, 2011 and October 19, 2011; released for publication October 26, 2011.

Refereeing of this contribution was handled by Peter Willett

Author's addresses: Defence Science and Technology Organisation, ISRD 200, PO Box 1500, Edinburgh, South Australia 5111, Australia, E-mail: (Samuel.Davey@DSTO.Defence.gov.au); School of Electrical and Electronic Engineering, The University of Adeliade, Australia.

1. INTRODUCTION

The conventional tracking paradigm is to sequentially apply a single frame detector to each sensor frame and then employ a tracking algorithm to determine which detector outputs originate from targets and which are false alarms. The tracker associates detections from a particular target and estimates parameters of interest for the target [3], [1]. The obvious shortcoming of this approach is that it is impossible for the tracker to recover a target if there are no detector outputs. For low signal strength targets, this implies that the detector threshold must be low and the tracker must attempt to suppress a large number of false alarms. In practice, the tracker can only do so much and for very low signal strength targets, conventional tracking fails [8].

In contrast, track-before-detect (TkBD) algorithms supply the whole predetection sensor frame to the tracker. In essence the tracking problem remains the same, but the measurement function is different. What makes TkBD challenging is that the relationship between the sensor frame and the target state is non-linear and often non-Gaussian, whereas point measurements may often be treated as linear and Gaussian. Apart from a small number of special cases, non-linear non-Gaussian estimation problems cannot be solved with a closed form algorithm. Instead, TkBD algorithms generally use a numerical approximation to make the problem tractable. The numerical approximation may take the form of a fixed discrete grid in state space [17], [2], [21], or a stochastic sampling method may be used (i.e., a particle filter) [19], [4], [18]. Alternatively, the data likelihood ratio may be viewed as an increasing function and the likelihood given a particular state sequence approximated by incoherent or binary integration [14], [20].

The Histogram Probabilistic Multi-Hypothesis Tracker (H-PMHT) is an exception: it is a TkBD algorithm that does not use numerical approximation [22], [24]. H-PMHT uses a unique histogram interpretation of the sensor frame and expectation maximisation (EM) to treat the TkBD problem as one of mixture fitting. The algorithm has the advantage of being capable of handling multiple targets whereas the numerical methods usually assume a single target. In addition, it does not require an assumed statistical distribution for the amplitude of the scene background, which may be difficult to adequately approximate in a realistic application.

Despite its advantages, little work has been published on H-PMHT besides the original algorithm development [22], [24] and its extension to spectral data [23]. Pakfiliz and Efe presented marginalised expressions for a two-dimensional filtering problem and compared H-PMHT with a conventional tracking approach (Interacting Multiple Model Probabilistic Data Association with Amplitude Information) [16]. They showed that H-PMHT provided lower estimation errors. However, the comparison used two independent simulated

data sets rather than producing the point-measurements for the conventional tracker from the sensor frame provided to H-PMHT. Also, the issue of track initiation, which is essentially the whole purpose of TkBD, was not considered. Davey, et al. [8] compared the initiation performance of H-PMHT with other TkBD methods and with sequential detection and tracking using simple straight line scenarios. The comparison showed that H-PMHT provided detection and estimation performance similar to the other TkBD approaches considered but at a fraction of the computation cost. While these results provide strong motivation for using H-PMHT, the scenarios were simplistic and not operationally relevant. Also, only limited implementation information was provided. Recently, Wieneke used a Wishart prior to enable the H-PMHT to estimate the covariance matrix of the target signature [26] and Davey demonstrated how a particle filter could be used for target state estimates when the target signature is highly non-Gaussian [6].

This article builds on the comparisons already available and addresses whether the algorithm still performs well with real sensor data, which inevitably does not match the modeling assumptions. It shows that the advantages illustrated on thumbnail images in [8] do extrapolate to larger, more relevant image sizes. The performance of the algorithm is investigated as a function of the image size, the number of targets and of the dynamic range in target echo strength. A matrix-vector form of the algorithm is derived for the case of a two dimensional sensor image frame and this is shown to significantly improve implementation efficiency. An efficient method for incorporating non-Gaussian targets is developed and compared with the computationally demanding particle filter approach introduced in [6]. Finally the H-PMHT is then demonstrated on an experimental data set containing a small boat collected by Defence Research and Development Canada (DRDC) [13]. The performance of H-PMHT on the experimental data is compared with a conventional tracking approach consisting of an image segmentation detector and two alternate trackers: a Probabilistic Data Association Filter (PDAF) and a point-measurement PMHT. The main purpose of this example is not a quantitative comparison, but rather to qualitatively investigate the scaling performance of the algorithm to high resolution radar imagery and the sensitivity of the algorithm to mismatch in the target and clutter models.

The article is arranged as follows: a review of the H-PMHT is presented in Section 2; Section 3 illustrates the performance scaling of the algorithm as a function of the measurement image volume, number of targets and diversity of targets, and derives an efficient implementation representation for two dimensional sensor images; Section 4 presents a track management method for H-PMHT; Section 5 introduces a new method for non-Gaussian targets; and Section 6 compares the performance of H-PMHT and conventional tracking on a challenging maritime surveillance experiment.

2. REVIEW OF H-PMHT

The H-PMHT algorithm, as introduced in [22], is now reviewed. 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 probability that an individual histogram shot falls in cell l is

$$\frac{P_l(\mathbf{X}_t)}{P(\mathbf{X}_t)} \tag{1}$$

where

$$P(\mathbf{X}_t) \equiv \sum_l P_l(\mathbf{X}_t) \tag{2}$$

and

$$P_l(\mathbf{X}_t) = \int_{B_l} f(\tau \mid \mathbf{X}_t) d\tau. \tag{3}$$

The spatial extent of cell l (which is of arbitrary dimensionality) is B_l and $\mathbf{X}_t$ is the system state vector at time t, i.e., it summarises all of the targets.

The underlying continuous spatial density, $f(\tau \mid \mathbf{X}_t)$ is the superposition of a background clutter model and M target components

$$f(\tau \mid \mathbf{X}_t) \equiv f(\tau \mid \mathbf{X}_t; \Pi_t) = \pi_t^0 G_0(\tau) + \sum_{m=1}^{M} \pi_t^m G^m(\tau \mid \mathbf{x}_t^m) \tag{4}$$

where $\mathbf{x}_t^m$ is the state of the mth target and the mixing proportions form a probability vector, i.e., $\pi_t^m \geq 0$ and

$$\sum_{m=0}^{M} \pi_t^m = 1. \tag{5}$$

The mixing proportions can be interpreted as the relative power of each target. In the simplest case, the background clutter model is uniform and $G_0(\tau)$ is a constant. In spatially non-uniform clutter, mapping approaches such as [5], [10] may be used. The target component $G^m(\tau \mid \mathbf{x}_t^m)$ may be common across targets, e.g. it could represent the point spread function (psf) for a sensor observing point scatterers, or each target may have a unique signature, such as with high resolution optical sensors. Either way, it is assumed to have a known functional form with potentially unknown parameters.

H-PMHT is an EM algorithm that treats the assignment of histogram shots to model components and the precise location of shots as missing data. In addition, it allows for unobserved cells. These are notionally sensor pixels for which no data was collected. One use for this concept is in tracking targets near the edge of the sensor frame [24]. The data from these unobserved

cells is also treated as missing data. Assuming an existing estimate of the system state, $\mathbf{X}_t^{(i)}$, and the mixing proportions, $\Pi_t^{(i)}$, H-PMHT determines the probability of the missing data (E-step) and then refines the state estimate (M-step).

Denote the per-cell proportion of the contribution of target m at time t as

$$P_l^m(\mathbf{x}_t^{m(i)}) = \int_{B_l} G^m(\tau \mid x_t^{m(i)}) d\tau \tag{6}$$

then

$$P_l(\mathbf{X}_t^{(i)}) = \sum_{m=0}^{M} \pi_t^{m(i)} P_l^m(\mathbf{x}_t^{m(i)}). \tag{7}$$

The observed power in cell l at time t is denoted z_{tl}. Define

$$\|Z_t\| = \sum_l z_{tl} \tag{8}$$

namely the L_1 norm, and let

$$\bar{z}_{tl} = \begin{cases} \dfrac{z_{tl}}{P_l(\mathbf{X}_t^{(i)})} & l \in \mathcal{L} \\ \dfrac{\|Z_t\|}{P(\mathbf{X}_t^{(i)})} & l \in \bar{\mathcal{L}} \end{cases} \tag{9}$$

where $\mathcal{L}$ is the set of all observed cells and $\bar{\mathcal{L}}$ is the set of all unobserved cells, which may be empty. $\mathcal{S}$ is the union of sets $\mathcal{L}$ and $\bar{\mathcal{L}}$.

The parameters to be estimated are the mixing proportions, π_t^m and the target states, $\mathbf{x}_t^m$. The updated mixing proportion estimate is given by

$$\pi_t^{m(i+1)} = \frac{p_t^{m(i)}}{\sum_{m=0}^{M} p_t^{m(i)}} \tag{10}$$

where

$$p_t^{m(i)} = \pi_t^{m(i)} \sum_{l \in \mathcal{S}} \bar{z}_{tl} P_l^m(x_t^{m(i)}). \tag{11}$$

The state estimate requires the maximisation of the function

$$Q_{mX} = \sum_{t=1}^{T} \frac{\|Z_t\|}{P(\mathbf{X}_t^{(i)})} \log\{p(\mathbf{x}_t^m \mid \mathbf{x}_{t-1}^m)\} + \sum_{t=1}^{T} \sum_{l \in \mathcal{S}} \pi_t^{m(i)} \bar{z}_{tl} \int_{B_l} G^m(\tau \mid x_t^{m(i)}) \log\{G^m(\tau \mid x_t^m)\} d\tau. \tag{12}$$

For the case of linear Gaussian statistics, Streit [22] demonstrated that this maximisation problem is equivalent to a point measurement filtering problem with synthetic measurements

$$\tilde{z}_t^{m(i)} = \frac{\pi_t^{m(i)}}{p_t^{m(i)}} \sum_{l \in \mathcal{S}} \bar{z}_{tl} P_l^m(x_t^{m(i)}) \tilde{z}_{tl}^{m(i)} \tag{13}$$

where the *cell-level centroid*, $\tilde{z}_t^{m(i)}$, is given by

$$\tilde{z}_{tl}^{m(i)} = \frac{1}{P_l^m(x_t^{m(i)})} \int_{B_l} \tau G^m(\tau \mid x_t^{m(i)}) d\tau. \tag{14}$$

The associated synthetic measurement covariance is

$$\tilde{R}_t^m = \frac{1}{p_t^{m(i)}} R_t^m \tag{15}$$

and the synthetic process covariance is

$$\tilde{Q}_t^m = \frac{P(X_t^i)}{\|Z_t\|} Q_t^m. \tag{16}$$

A Kalman Filter can be used to solve this point measurement filtering problem.

The H-PMHT algorithm then consists of iteratively equating:

- cell probabilities, (6) and (7)
- expected measurements, (9)
- cell-level centroids, (14)
- synthetic measurements and covariance matrices, (13), (15) and (16)
- mixing proportion estimates, (10)
- target state estimates, by filtering the synthetic measurements and covariances

3. EFFICIENT IMPLEMENTATION

The majority of the H-PMHT literature does not discuss the computation resource required for the algorithm. The exception is the comparison work in [8], which shows that the algorithm is inexpensive compared with other TkBD methods on thumbnail images. However, it is intuitive that a direct implementation of the equations reviewed in the previous section will have a computation cost that scales linearly with the number of pixels in the data image. This is consistent with results reported by Vo, et al. [25] where a non-linear growth was found as a function of resolution (the authors of [25] suggest that this growth is exponential, although one should expect it to be quadratic). This growth is clearly undesirable and potentially disastrous for video imagery, which can be expected to contain millions of pixels.

We now demonstrate that with a little finesse, this quadratic complexity can be tamed, and introduce other measures aimed at improved efficiency.

3.1. Pixel Gating

The standard H-PMHT equations as presented in Section 2 show a linear complexity in the number of targets and the number of pixels in the sensor image. Put simply, there are a number of operations which (at first blush) should be performed for each target over every pixel. However, intuition suggests that pixels very distant from the target state estimate will have negligible influence.

This is nothing more than an expression of the ubiquitous method of gating for point-measurement estimators. For example, under Probabilistic Data Association or point-measurement PMHT, one could calculate an association probability for every measurement in the

frame, but many of these measurements are so far away from the target that we know their association probability will be zero: there is no need to waste the computer's time in calculating these numbers.

To make this discussion more crisp, let $\mathcal{G}^m$ be the set of cells with non-negligible probability for target m, namely $\mathcal{G}^m \equiv \{l : P_l^m(x_t^m) > \epsilon\}$ for some arbitrary small ϵ. Equation (11) can then be replaced with

$$p_t^{m(i)} = \pi_t^{m(i)} \sum_{l \in \mathcal{G}^m} \bar{z}_{tl} P_l^m(x_t^{m(i)}) \tag{17}$$

and similarly for (13).

It is clear that for a fixed target, the cardinality of $\mathcal{G}^m$ is independent of the cardinality of $\mathcal{S}$, ignoring edge effects. That is, the complexity is independent of the image size for a fixed target size.

3.2. Clustering

Another common method for reducing complexity in multi-target trackers is clustering [3]. For algorithms whose computation complexity is more than linear in the number of targets, it makes sense to divide the total set of targets into subsets of potentially interacting targets such that each subset is independent of the others. These subsets are commonly called clusters and the process of forming them clustering. For H-PMHT the complexity is linear in the number of targets, so there would seem to be no real advantage in performing clustering.

While clustering will not improve the computation cost for H-PMHT, it will make a difference to the memory requirements. Assuming the gating approach described above, there will be a collection of gates, $\mathcal{G}^m$. Each track only needs to operate over the pixels in its own gate. However, there are some actions that operate over the whole image. Specifically, the overall cell probability $P_l(\mathbf{X}_t)$ should be determined for the region $\bigcup \mathcal{G}^m$, and some of the track management functions to follow operate over the whole image. If clustering is used, then the amount of memory required for large arrays would be reduced. The performance improvement gained can be significant for large images where memory caching plays an important role in overall speed.

Clustering methods are well known in the context of point-measurement multi-target tracking and are not pursued in this paper.

3.3. Partial EM Update

H-PMHT is an EM algorithm that iterates data association and state estimation stages until convergence. The proper way to measure this convergence is through the EM auxiliary function, which can be expressed as

$$Q(\mathbf{X}, \Pi \mid \mathbf{X}^{(i)}, \Pi^{(i)}) = \sum_{t=1}^{T} Q_{t\Pi} + \sum_{m=0}^{M} Q_X^m \tag{18}$$

where the term $Q_{t\Pi}$ depends only on the mixing proportions, and each term Q_X^m depend on the state sequence of a single target [22]. Importantly, the target state sequences are not coupled, so each Q_X^m can be independently optimised and the estimation stage is a bank of parallel single target estimators.

EM iterations should be repeated until (18) converges, i.e., all target states and the mixing proportions are estimated repeatedly until convergence. The $Q_{t\Pi}$ term couples all the targets together through normalisation, so this term will not converge until all of the target states converge. However, it is likely that each target will converge at a different rate. In particular, when a track has been spuriously initiated without the support of a real target, the state estimates will usually converge very quickly, since the data association stage finds no support for the track and it essentially dead-reckons. In contrast, tracks that are initiated on a real target response may take many iterations to recover from initialisation error, particularly if the target velocity is high. So it is likely that much effort would be spent repeating the estimation of target states for some tracks that have already converged while the algorithm waits for other tracks that have not converged.

An intuitive way to reduce the redundant reestimation of converged states is to use partial EM steps. Under a partial EM step, only a subset of the states are iterated. Because of the factorised form of the auxiliary function (18), the only way that a change in the state of target m will effect the target auxiliary component for a different target p, Q_X^p, is if the change in the state of target m affects the data association stage for target p. Under the assumption that this will not happen if target p has already converged, then one may employ a partial EM iteration that keeps the state of p fixed and refines the state of m. This not only saves on the computation required for the estimator of p, but also the computation required for the cell probabilities, expected measurements, centroids and synthetic measurements. Adopting partial EM iterations in this manner can reduce the overall algorithm resource requirement by as much as an order of magnitude because the majority of the tracks carried by the algorithm are spurious and converge very quickly.

In practice the convergence test for the algorithm will limit the number of iterations based on the maximum tolerable processing delay per frame. By using partial updates, the time taken for each iteration is reduced as the iteration count increases, since fewer tracks still require refinement. The result is that a much greater maximum number of iterations may be used without significant impact on the overall algorithm speed.

The partial-EM H-PMHT consists of the following steps:

- For converged tracks, fix the cell probabilities using the value determined on the final iteration before convergence.

- For non-converged tracks, determine the new cell probabilities for this iteration using (6).
- Calculate the overall cell probabilities (7).
- For non-converged tracks determine:
- expected measurements, (9)
- cell-level centroids, (14)
- synthetic measurements and covariance matrices, (13), (15) and (16)
- target state estimates, by filtering the synthetic measurements and covariances
- For all tracks, determine new mixing proportion estimates, (10)

In principle the convergence test should be based on the cost function element Q_X^m. However, the only reason to determine this quantity is to test for convergence, and this is relatively expensive. Therefore the implementation used in this paper used a convergence test based on the target state: when the difference between the state estimates for consecutive iterations dropped below a threshold, the track was deemed converged. A maximum of ten iterations was also applied.

3.4. Separable Measurement Function

The H-PMHT derivation uses single indexing of the sensor cells. Single indexing does not limit the sensor dimensionality, but it makes it more difficult to decouple the sensor dimensions if they are independent. For the following, assume a two dimensional sensor with an independent grid such that

$$\int_{B_l} G(\tau \mid \mathbf{x}) d\tau \equiv \int_{B_X^i} \int_{B_Y^j} G(u, v \mid \mathbf{x}) dv\, du \tag{19}$$

where the arbitrary index l corresponds to the same cell as the arbitrary double index i, j.

Assume also a separable point spread function:

$$G(u, v \mid \mathbf{x}) \equiv g_X(u \mid \mathbf{x}) g_Y(v \mid \mathbf{x}). \tag{20}$$

The standard H-PMHT equations presented in the previous section can solve this problem, but it's desirable to exploit the structure of the target function to achieve a factorised algorithm. The results derived here are equivalent to the algorithm presented in [16]. Here more detail is shown to justify the end result and compact vector notation is adopted to help make compact intuitive expressions and to point towards efficient implementation.

In the following discussion, the time subscripts and iteration indices are suppressed to simplify notation.

The per-cell target probabilities (6) directly factorise into a product of two integrals:

$$\begin{aligned} P_{ij}^m(\mathbf{x}_t^m) &= \left\{ \int_{B_X^i} g_X(u \mid \mathbf{x}^m) du \right\} \left\{ \int_{B_Y^j} g_Y(v \mid \mathbf{x}^m) dv \right\} \\ &\equiv P_X^m(i) P_Y^m(j). \end{aligned} \tag{21}$$

Define the stacked vectors

$$\mathbf{P}_X^m = [P_X^m(1), P_X^m(2), \ldots P_X^m(N_X)]^{\mathrm{T}} \tag{22}$$

where N_X is the number of cells in the X-dimension. Similarly define $\mathbf{P}_Y^m$.

Let P_X be a matrix of the X per-cell contributions

$$\mathsf{P}_X = [\mathbf{P}_X^0, \mathbf{P}_X^1, \ldots \mathbf{P}_X^M] \tag{23}$$

and similarly define P_Y, then

$$\begin{aligned} \mathsf{P} \equiv \{P_{ij}(\mathbf{X})\} &= \left\{ \sum_{m=0}^{M} \pi^m P_{ij}^m(\mathbf{x}^m) \right\} \\ &= \mathsf{P}_X \Lambda \mathsf{P}_Y^{\mathrm{T}} \end{aligned} \tag{24}$$

where Λ is a diagonal matrix of the π^m. Note that (24) is simply a matrix-vector version of (7). They are equivalent, but (24) represents an explicit factorisation of (7) exploiting the separable point spread function.

The normalised measurements are

$$\bar{\mathsf{Z}} = \mathsf{Z}_t ./ \mathsf{P} \tag{25}$$

where the Matlab-style ./ notation denotes element-wise division.

The unscaled mixing proportion estimate is given by

$$p^m = \pi^m \sum_{i,j} \bar{z}_{ij} P_X^m(i) P_Y^m(j) = \pi^m \{\mathbf{P}_X^m\}^{\mathrm{T}} \bar{\mathsf{Z}} \mathbf{P}_Y^m. \tag{26}$$

Again, (26) is equivalent to (11) but explicitly factorises the calculation of p^m due to the separable point spread function.

The cell-level centroids are given by

$$\begin{aligned} \tilde{\mathbf{z}}_{ij}^m &= \frac{1}{P_X^m(i) P_Y^m(j)} \int_{B_X^i} \int_{B_Y^j} \begin{bmatrix} u \\ v \end{bmatrix} g_X(u \mid \mathbf{x}^m) g_Y(v \mid \mathbf{x}^m) dv\, du, \\ &= \frac{1}{P_X^m(i) P_Y^m(j)} \begin{bmatrix} \int_{B_X^i} \int_{B_Y^j} u g_X(u \mid \mathbf{x}^m) g_Y(v \mid \mathbf{x}_t^m) dv\, du \\ \int_{B_X^i} \int_{B_Y^j} v g_X(u \mid \mathbf{x}^m) g_Y(v \mid \mathbf{x}_t^m) dv\, du \end{bmatrix} \\ &= \frac{1}{P_X^m(i) P_Y^m(j)} \begin{bmatrix} P_Y^m(j) \int_{B_X^i} u g_X(u \mid \mathbf{x}^m) du \\ P_X^m(i) \int_{B_Y^j} v g_Y(v \mid \mathbf{x}^m) dv \end{bmatrix} \\ &= \begin{bmatrix} 1/P_X^m(i) \int_{B_X^i} u g_X(u \mid \mathbf{x}^m) du \\ 1/P_Y^m(j) \int_{B_Y^j} v g_Y(v \mid \mathbf{x}^m) dv \end{bmatrix} \equiv \begin{bmatrix} \tilde{z}_X^m(i)/P_X^m(i) \\ \tilde{z}_Y^m(j)/P_Y^m(j) \end{bmatrix} \end{aligned} \tag{27}$$

i.e., as intuitively expected, the centroids are independent in the sensing dimensions. Note that the term $\tilde{z}$ is different to (14) in that it does not include the normalising term (abusing notation, $\tilde{z}_{14}^m = \tilde{z}_{27}^m / P^m$). This has been done to achieve a more efficient vector formulation in the following step.

Thus the synthetic measurements are

$$\begin{aligned}\tilde{\mathbf{z}}^m &= \frac{\pi^m}{p^m}\sum_i\sum_j \bar{z}_{ij}P_X^m(i)P_Y^m(j)\tilde{\mathbf{z}}_{ij}^m,\\ &= \frac{\pi^m}{p^m}\begin{bmatrix}\sum_i\sum_j \bar{z}_{ij}P_X^m(i)P_Y^m(j)\tilde{z}_X^m(i)/P_X^m(i)\\ \sum_i\sum_j \bar{z}_{ij}P_X^m(i)P_Y^m(j)\tilde{z}_Y^m(j)/P_Y^m(j)\end{bmatrix}\\ &= \frac{\pi^m}{p^m}\begin{bmatrix}\sum_i \tilde{z}_X^m(i)\left\{\sum_j \bar{z}_{ij}P_Y^m(j)\right\}\\ \sum_j \tilde{z}_Y^m(j)\left\{\sum_i \bar{z}_{ij}P_X^m(i)\right\}\end{bmatrix}\\ &= \frac{\pi^m}{p^m}\begin{bmatrix}\{\tilde{\mathbf{z}}_X^m\}^{\mathrm{T}}\bar{\mathbf{Z}}\mathbf{P}_Y^m\\ \{\mathbf{P}_X^m\}^{\mathrm{T}}\bar{\mathbf{Z}}\tilde{\mathbf{z}}_Y^m\end{bmatrix}\end{aligned} \tag{28}$$

where

$$\tilde{\mathbf{z}}_X^m = [\tilde{z}_X^m(1),\ldots\tilde{z}_X^m(N_X)]^{\mathrm{T}} \tag{29}$$

and similarly for $\tilde{\mathbf{z}}_Y^m$.

The synthetic measurement and process covariances remain the same, and the state estimates may now be determined using a Kalman Filter.

The results above can be extended to higher dimensions, for example if the sensor forms range, Doppler, azimuth and elevation bins, then the sensor image would be four dimensional. In such a case, compact expressions such as (28) rely on stacking the dimensions to take advantage of matrix algebra.

The matrix-vector formulation above does not explicitly reduce the number of operations required. However, if the algorithm is implemented in a higher level computer language, it is likely that expressing the mathematics this way makes the algorithm amenable to the language's intrinsic pipelining and parallelisation capabilities. It is the author's experience that within the Matlab environment the speed difference is significant.

3.5. Numerical Illustration

Consider a simplistic example to numerically verify the scaling of the algorithm as a function of image size and the number of targets. At this stage we have not introduced track management, so the algorithm is initialised with the true state of each target and recursively updated using the various efficiency measures described above. Fig. 1 shows an example scenario with 24 targets. All of the targets move with a speed of 1 pixel per frame and are positioned away from the image boundaries and each other. For small images with many targets, there will be inevitable overlap.

Monte Carlo trials of the scenario in Fig. 1 were performed as a function of the number of targets and the image size. The target spacing was proportional to the image size. For each combination of target count and image size, 100 Monte Carlo trials were performed (a total of $9 \times 16 \times 100 = 14400$ trials). The CPU time spent by H-PMHT is shown in Fig. 2 with 1-sigma error-bars. Fig. 2(a) shows the increase in cost as a function of the number of targets with a separate curve

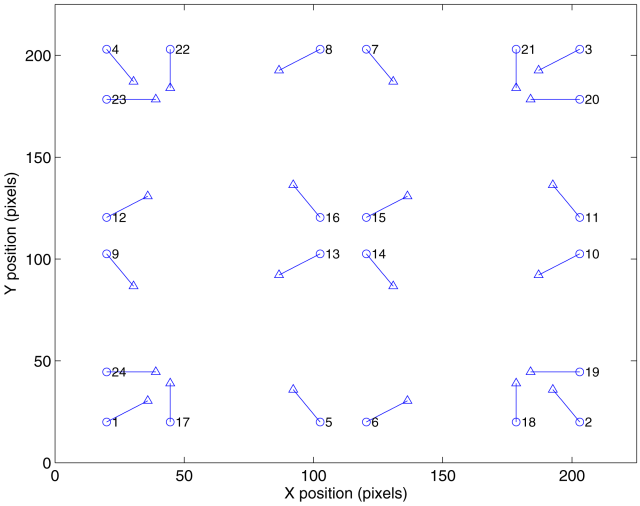


Fig. 1. Scaling scenario.

for each image size. There is no statistical difference between any of these curves. The growth appears to be somewhat less than linear in the number of targets. This is because of the pipelining effect mentioned in the previous section: the code itself has a simple loop over the targets, but the cost of executing this loop is less than linear because of loop overheads and the Matlab interpreter's ability to optimise the loop. The independence of the CPU cost from total image size is emphasised in Fig. 2(b) where the cost for a fixed number of targets is shown as a function of image size. Different curves show an increasing target density, but each curve remains flat.

Next the effect of partial EM iterations was investigated using a fixed number of 4 targets and an image size of 50,000 pixels. The target geometry was the same as in the scaling simulations. However, track management was used to automatically acquire the targets instead of initialising with the truth, as above. The track manager is described in Section 4. The maximum number of EM iterations for a single time update was limited to 100, but this limit was rarely met. The number of iterations for each update of each track was counted for 100 monte carlo trials. Under full EM iterations, all tracks are iterated until all tracks have converged, so the number of iterations was the same for all tracks at a particular scan. Under partial EM iterations, iterations were only performed on tracks yet to converge, so the number of iterations can be different for each track.

The track manager used candidate and established tracks as part of the initiation scheme, as described in Section 4. Separate counts were recorded for candidate and established tracks. The mean number of EM iterations for established tracks was 2.7 under full EM iterations and 2.1 under partial EM iterations. The benefit is minor because the uncertainty in the target state is small and only a couple of EM iterations are required. In effect, the prior is very good because these tracks correspond to real targets that have been under track

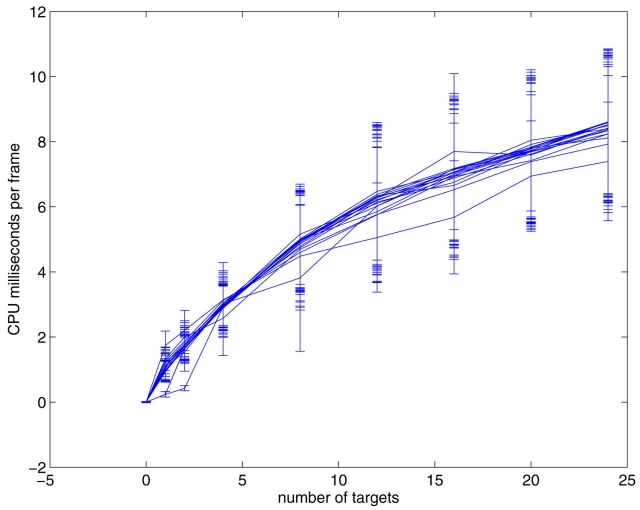


Fig. 2. CPU requirement scaling. (a) Growth with number of targets. (b) Growth with image size.

for some time. In contrast, the mean number of EM iterations for candidate tracks was 55.8 under full EM and 12.0 under partial EM. Histograms of the number of iterations per track under full and partial EM are shown in Fig. 3.

In this example, the tracker only formed established tracks on the 4 targets whereas there were an average of 74.8 candidate tracks. This means that the computational performance is limited by the candidate tracks, and the partial EM iterations have delivered roughly a factor of 5 improvement. The number of candidate tracks depends on the size of the surveillance region and, in practice, the maximum number of iterations would be limited to many fewer than 100. This means that the benefit of partial EM iterations will be application dependent and likely much less than the factor of 5 here. Nevertheless, it offers a substantial improvement.

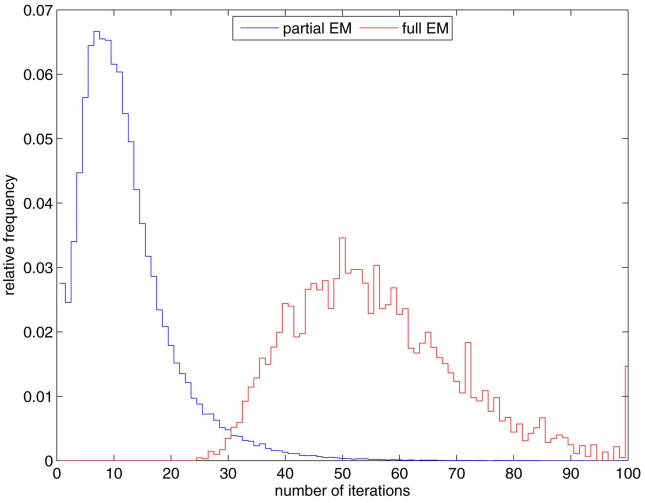


Fig. 3. Benefit of partial EM iterations.

4. TRACK MANAGEMENT

The core H-PMHT algorithm updates existing tracks, but does not provide a means for initiating new tracks or terminating old tracks. These functions are essential for a operationally relevant tracker, especially since the benefit of H-PMHT is espoused as high detection sensitivity. This section describes how track initiation and termination can be incorporated into the algorithm.

To begin with, observe that the H-PMHT as a black-box looks very similar to a Joint Probabilistic Data Association tracker: there is a list of tracks, grouped into clusters, which interact with each other, and the algorithm modifies these tracks based on observed data. Given this structural familiarity, it should come as no surprise that well known track maintenance methods can be directly used with H-PMHT.

The tracker uses two lists of tracks: established tracks, which the algorithm has high confidence in; and candidate tracks (also referred to as tentative tracks), which correspond to potential targets, but do not have a high confidence. Candidate tracks are promoted to established based on their confidence, and track termination is also based on confidence. Established tracks have priority access to the sensor data, followed by candidate tracks, and finally a process which forms new candidate tracks, as shown in Fig. 4. In the figure, the two track-update blocks are separate instances of the H-PMHT algorithm. For the purposes of this paper they are identical, although for some applications it may be appropriate to vary their parameters. The measurements and residual measurements are sensor images. The hierarchial data access prevents the tracker from forming duplicate tracks on targets that already have an existing track in either the established or candidate list.

Recall that these two track lists are a representation of a mixture model. In this light, one can view the established track list as those components of the mixture that are persistent and relatively strong, whereas the

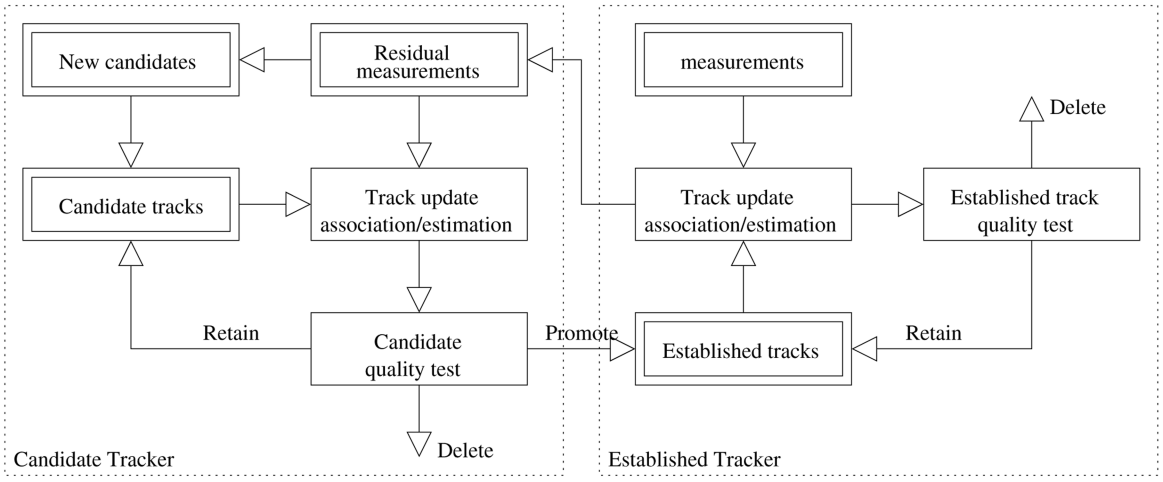


Fig. 4. Track management flow diagram.

candidate track list is a mixture representation of the remaining image structure that is inconsistent with the clutter model.

In order to realise the management structure in Fig. 4 the following elements are required: a measure of track quality or confidence; a set of rules for track decisions; a method for vetting the sensor image to enable hierarchial data access; and a method for forming new candidate tracks based on data. Each of these is now described.

The quality of each track is quantified by estimating the track's SNR. In this context, define the SNR as the ratio of the peak level of the target spread function to the local noise floor. It will be expressed in decibels, namely

$$s_t^m \equiv 10\log_{10}\frac{\pi_t^m \max_l\{P_l^m\}}{\pi_t^0}. \tag{30}$$

For a rectangular image, a Gaussian point spread function and uniform clutter, the SNR is simplified to

$$s_t^m = 10\log_{10}\frac{\pi_t^m N_X N_Y}{\pi_t^0 |2\pi\mathsf{R}|}. \tag{31}$$

The SNR equations above use the true mixing proportions, π_t^m, which must be replaced by their estimates, since they are unknown. This paper is focused on a sequential implementation of H-PMHT, which means that these estimates are based on a single frame of data and are therefore expected to fluctuate. An alternative is to introduce dynamics within the π_t^m priors through the Hysteresis approach [7]. This method has not yet been applied to H-PMHT. However, the algorithm is very similar to the point-measurement PMHT used in [7] and the application of the Hysteresis model would be relatively straightforward.

The next element of the track manager is the set of decision rules. Since the track confidence is driven by SNR estimates based on a single frame of data, the track quality statistics are expected to fluctuate and it is appropriate to use M out of N style decisions, much the same as might be used in a conventional point-measurement tracker. The particular threshold and suitable values for M and N are application dependent. The values used in specific examples in this paper are provided with the examples.

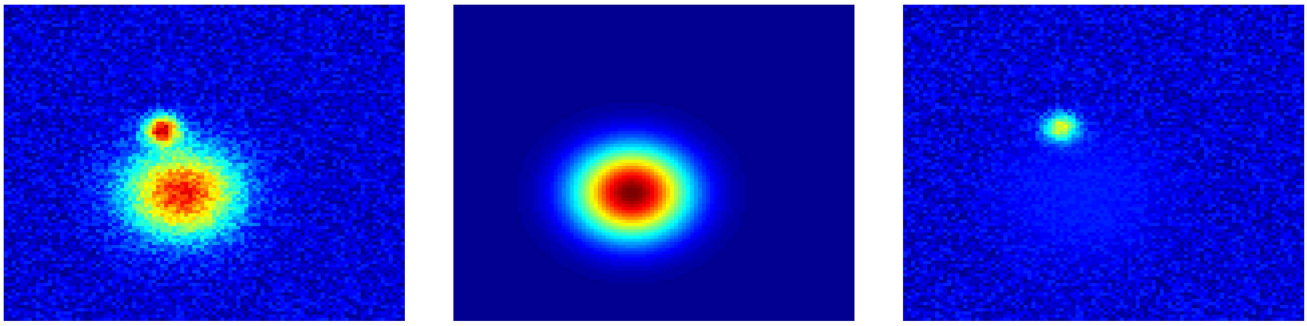
The data vetting function is used to give the different status tracks hierarchial access to the sensor image. In a point-measurement tracker, this is as simple as removing measurements that have been associated with established tracks from the measurement list before data association with the candidate tracks. The H-PMHT requires a vetting method that modifies the sensor image and removes target energy associated with existing tracks. For this, the track manager uses the *whitening* method proposed in [24].

The H-PMHT estimates the parameters of a mixture distribution. On convergence, the total cell probabilities, P, represent the fitted distribution. Near targets, the P_l values will be relatively high and for pixels far from targets, the P_l value will be purely the background noise component, $\pi^0 P_l^0$. The ratio of the total cell probability to the background component, $P_l/(\pi_t^0 P_l^0)$ is a mask which is unity far from targets and has a higher value close to targets. Dividing the sensor image pixel-wise by this mask suppresses the targets

$$z'_{tl} = \frac{\pi_t^0 P_l^0}{P_l} z_{tl}. \tag{32}$$

This whitened image is provided as input to subsequent stages. Namely, the established track mixture whitens the image before candidate tracks are updated and then the candidate track mixture whitens the image before the manager attempts to start new candidate tracks.

Fig. 5 shows a simplified illustration of the process. In the example, there are two targets: one broad target at $(45,60)$ and a more compact one at $(40,40)$. The background noise is uniform and Gaussian. Fig. 5(a) shows the sensor image. Suppose that the tracker has an established track on the broad target. Fig. 5(b) shows the mixture model corresponding to the established track



(a) (b) (c)

Fig. 5. Sensor image vetting. (a) Original sensor image. (b) Mixture pdf of existing tracks. (c) Whitened sensor image.

list. Fig. 5(c) shows the whitened image achieved by pixel-wise dividing the image in (a) by the image in (b). The second target is preserved but the first target, for which a track already exists, is suppressed.

The final element of the track manager is the process to create new candidate tracks. The author has tested two alternatives for this process. The first alternative is to use incoherent integration along with the process model to accumulate data over multiple frames. When a pixel exceeds a prescribed threshold, a new candidate track is created in that location with zero velocity. This is similar to one-point initialisation [3] but the detector input is an accumulated image instead of a single frame. The accumulated image is recursively defined as

$$\zeta_t = (1-\alpha)\mathsf{Z}_t + \alpha\{\zeta_{t-1} \star \mathsf{T}\} \tag{33}$$

where $0 < \alpha < 1$ is a forgetting factor, $\star$ is a two dimensional convolution, and T is a transition kernal which gives the prior probability that a target in pixel (i, j) at time $t-1$ will be in pixel (i', j') at time t: it is specified by the prior target velocity distribution. The author found this method to work well for slow moving targets.

The application in this paper uses the second method for forming candidates: two point differencing [3]. Here a single frame detector is used to locate potential targets and then a new candidate track is formed on pairs of detections from consecutive frames when they are sufficiently close. This was found to work better in the application because the target moves through tens of pixels between each frame and so the transition kernal T above was too broad: the accumulated image was dominated by the current measurement Z_t and too many candidate tracks were initiated on clutter.

5. STRUCTURED TARGETS

The original presentation of the H-PMHT algorithm [22] showed that for linear Gaussian mixture components, i.e., $G(\tau \mid \mathbf{x}) = \mathcal{N}(\tau; \mathsf{H}\mathbf{x}, \mathsf{R})$, the EM auxiliary function is equivalent to the log-likelihood of a point-measurement filtering problem. This is achieved by factorising (12) and completing the square. Essentially (12) is a sum of quadratics in the target state which may be collected into a single quadratic, which is equivalent to the log of a normal distribution.

The same method can be used for a Gaussian measurement function where the mean is a non-linear function of the target state, i.e., $G(\tau \mid \mathbf{x}) = \mathcal{N}(\tau; \mathbf{h}(\mathbf{x}), \mathsf{R})$, with $\mathbf{h}(\mathbf{x})$ any function of the target state. In this case completing the square leads to an equivalent point-measurement problem where the measurements are non-linear and the noise is Gaussian [6]. The original non-linear relationship between the mean of the target point-spread-function and the target state is preserved as the non-linear point-measurement function, i.e.,

$$\sum_{l \in \mathcal{S}} \pi_t^{m(i)} \bar{z}_{tl} \int_{B_l} G^m(\tau \mid x_t^{m(i)}) \log\{G^m(\tau \mid x_t^m)\} d\tau \rightarrow \log\{\mathcal{N}(\tilde{\mathbf{z}}; \mathbf{h}(\mathbf{x}), \tilde{\mathsf{R}})\}. \tag{34}$$

Depending on the non-linearity, this might be solved by analytic linearisation (an extended Kalman Filter) or Monte Carlo methods.

For a target signature that is non-Gaussian, it becomes more difficult. For an arbitrary target, the integral in (34) is intractable. One solution is to use a numerical approximation: in [6] a grid-based approximation was used to evaluate the integral in (34) which was then used as an equivalent likelihood function to drive a particle filter. This method is capable of tracking any target structure, including a non-parametric shape estimate derived from data. However, it is numerically intensive. This section describes an alternative method that can used for non-Gaussian targets based on approximating the target signature as a Gaussian with spatially varying parameters.

5.1. Cell-Varying Point Spread Function

Unfortunately, the application that follows has a more complicated measurement function than above. In the application, the sensor builds up an image by sweeping an antenna across azimuth and measuring a range profile at each position. This creates an array of data in range bins and azimuths, but the data from different azimuths is collected sequentially, not simultaneously. This means that the airborne sensor platform

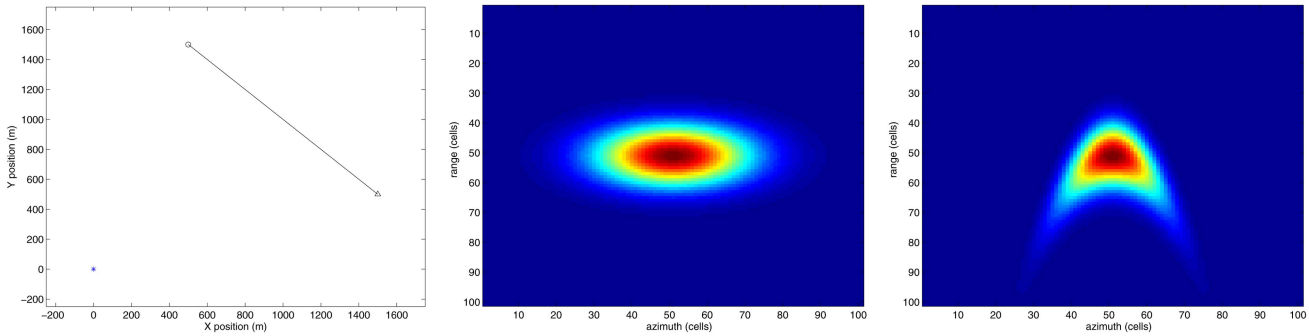


Fig. 6. Point spread function distortion due to sensor motion. (a) Target-sensor geometry. (b) Fixed sensor. (c) Moving sensor.

moves between azimuth collects, and the relative target position changes. The sensor response is a non-linear function of the relative target position. One way to model this is through a cell-varying target response: $G(\tau \mid x) \rightarrow G_l(\tau \mid x)$.

It can be shown (see Appendix A.1 for details) that the point-measurement result (34) can be further extended to the cell-varying psf situation. Provided that the target response can be expressed as a Gaussian function with a cell-dependent non-linear mean, $G_l(\tau \mid \mathbf{x}) = \mathcal{N}(\tau; \mathbf{h}_l(\mathbf{x}), \mathsf{R})$, then there exists an equivalent point-measurement problem with

$$\sum_{l \in \mathcal{S}} \pi_t^{m(i)} \bar{z}_{tl} \int_{B_l} G_l^m(\tau \mid x_t^{m(i)}) \log\{G_l^m(\tau \mid x_t^m)\} d\tau \rightarrow \log\{\mathcal{N}(\tilde{\mathbf{z}}_t^m; \tilde{\mathbf{h}}_t^m(\mathbf{x}_t^m), \tilde{\mathsf{R}}_t^m)\}. \quad (35)$$

Refer to $\tilde{\mathsf{R}}_t^m$, $\tilde{\mathbf{h}}_t^m(\mathbf{x})$ and $\tilde{\mathbf{z}}_t^m$ as the synthetic-measurement-covariance, synthetic-mean and synthetic-measurement respectively. For clarity, suppress the time and target indices.

The synthetic measurement covariance is the same as for the linear case, given in (15), namely

$$\tilde{\mathsf{R}} = \frac{1}{p_t^{m(i)}} \mathsf{R} \equiv (\pi_t^{m(i)} w)^{-1} \mathsf{R}$$

where $w = \sum_{l \in \mathcal{S}} w_l$, and $w_l = \bar{z}_{tl} P_l^m(x_t^{m(i)})$.

The synthetic mean is defined as

$$\tilde{\mathbf{h}}^{\mathrm{T}} \mathsf{R}^{-1} \tilde{\mathbf{h}} = \frac{1}{w} \sum_{l \in \mathcal{S}} w_l \mathbf{h}_l^{\mathrm{T}} \mathsf{R}^{-1} \mathbf{h}_l. \quad (36)$$

If the psf is separable, as in the previous section, then R will be diagonal, and (36) simplifies to

$$\tilde{\mathbf{h}}(j) = \sqrt{\frac{1}{w} \sum_{l \in \mathcal{S}} w_l \mathbf{h}_l(j)^2}. \quad (37)$$

That is, $\tilde{\mathbf{h}}$ is a weighted RMS average of the $\mathbf{h}_l$ values.

The synthetic measurement is defined as

$$\tilde{\mathbf{h}}^{\mathrm{T}} \mathsf{R}^{-1} \tilde{\mathbf{z}} = \frac{1}{w} \sum_{l \in \mathcal{S}} w_l \mathbf{h}_l^{\mathrm{T}} \mathsf{R}^{-1} \tilde{\mathbf{z}}_l \quad (38)$$

where $\tilde{\mathbf{z}}_l$ is the cell-level centroid for the target at cell l and is given by (14). Again, if the psf is separable, then R will be diagonal, and (38) simplifies to

$$\tilde{\mathbf{z}}(j) = \frac{1}{w \tilde{\mathbf{h}}(j)} \sum_{l \in \mathcal{S}} w_l \mathbf{h}_l(j) \tilde{\mathbf{z}}_l(j). \quad (39)$$

5.2. Application Example

For the application in this paper, the sensor forms range and azimuth cells and it moves as it collects the data. We now demonstrate how the cell-dependent psf model just described can be used to account for this motion.

Let (u_{tl}^s, v_{tl}^s) denote the Cartesian position of the sensor when cell l was observed. These locations change from one scan to the next and within the scan (hence the indices). The target state is in a Cartesian frame $\mathbf{x}_t^m = [u_t^m, \dot{u}_t^m, v_t^m, \dot{v}_t^m]^{\mathrm{T}}$ but the sensor response is polar, based on the range and bearing to the target when the particular cell was formed.

Note that both the target and the sensor move as a function of time but only the sensor motion has been modeled at a cell-to-cell level. The target has been implicitly assumed to be at the same position for every cell l. This is not correct because it's a moving target, but the target speed in the application is small and the effect of target motion will be assumed to be minor.

Because the sensor moves between collecting cells, the range and bearing equation is cell-dependent and is given by

$$\mathbf{h}_l(\mathbf{x}_t^m) = \left\{ \begin{array}{c} \sqrt{(u_t^m - u_{tl}^s)^2 + (v_t^m - v_{tl}^s)^2} \\ \arctan\left(\dfrac{v_t^m - v_{tl}^s}{u_t^m - u_{tl}^s} \right) \end{array} \right\}. \quad (40)$$

The effect of this sensor movement is therefore to distort the psf. Fig. 6 illustrates the difference between the psf for a fixed sensor and for a moving one. Fig. 6(a) shows the target-sensor geometry, (b) shows the psf for a fixed sensor at (1000, 1000) and (c) shows the moving sensor psf. This is an exaggerated example, but illustrates the potential effect of sensor motion.

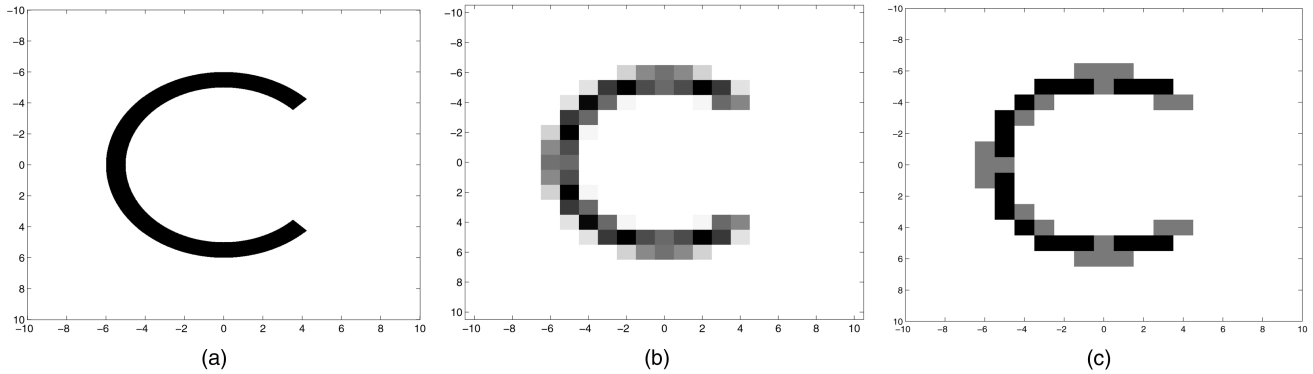


Fig. 7. Non-Gaussian point spread function. (a) Continuous-domain target psf. (b) Target response per cell. (c) Cell-varying spread function.

Substituting the cell-dependent $\mathbf{h}_l(\mathbf{x}_t^m)$ definition into the synthetic measurement leads to the equivalent measurement function

$$\tilde{\mathbf{h}}(\mathbf{x}) = \begin{Bmatrix} \left(\frac{1}{w}\sum_l w_l[(u_t^m - u_{tl}^s)^2 + (v_t^m - v_{tl}^s)^2]\right)^{1/2} \\ \left(\frac{1}{w}\sum_l w_l\left[\arctan\left(\frac{v_t^m - v_{tl}^s}{u_t^m - u_{tl}^s}\right)\right]^2\right)^{1/2} \end{Bmatrix} \tag{41}$$

which is the weighted root-mean-square range and bearing.

The estimation problem has been transformed from an image-measurement with target distortion due to motion, into a point-measurement with Gaussian noise. The implementation demonstrated in Section 6 used an extended Kalman Filter (EKF) to deal with the nonlinearity [3]. Although the weighting values, w_l, are data dependent, the synthetic measurement function may be constructed analytically, so it is possible to determine a Jacobian.

The sensor positions in (41) are known constants and the derivatives required for the Jacobian can be shown to be

$$\frac{\partial\tilde{\mathbf{h}}(\mathbf{x})[1]}{\partial u_t^m} = \frac{1}{\tilde{\mathbf{h}}(\mathbf{x})[1]}\left(u_t^m - \frac{1}{w}\sum_l w_l u_{tl}^s\right)$$

$$\frac{\partial\tilde{\mathbf{h}}(\mathbf{x})[1]}{\partial v_t^m} = \frac{1}{\tilde{\mathbf{h}}(\mathbf{x})[1]}\left(v_t^m - \frac{1}{w}\sum_l w_l v_{tl}^s\right)$$

$$\frac{\partial\tilde{\mathbf{h}}(\mathbf{x})[2]}{\partial u_t^m} = -\frac{1}{\tilde{\mathbf{h}}(\mathbf{x})[2]}\frac{1}{w}\sum_l w_l\frac{v_t^m - v_{tl}^s}{(u_t^m - u_{tl}^s)^2 + (v_t^m - v_{tl}^s)^2}$$

$$\frac{\partial\tilde{\mathbf{h}}(\mathbf{x})[2]}{\partial v_t^m} = \frac{1}{\tilde{\mathbf{h}}(\mathbf{x})[2]}\frac{1}{w}\sum_l w_l\frac{u_t^m - u_{tl}^s}{(u_t^m - u_{tl}^s)^2 + (v_t^m - v_{tl}^s)^2}$$

which are unsurprisingly very similar to the terms found in a standard range-bearing point-measurement problem [3].

5.3. Numerical Illustration

The result above may seem obscure and perhaps only useful in special situations. However, it can actually be used as a very general tool for approximating non-Gaussian targets. In [6] the particle filter was used to track a non-Gaussian target with H-PMHT. The target in that example was chosen look look like the letter C. It was shown that a non-Gaussian H-PMHT was far superior to the Gaussian approximation on several examples, including crossing targets and track initiation. We now demonstrate that this same letter C target can be tracked well using a cell-varying Gaussian point spread function. The purpose here is to qualitatively indicate the utility of the approach, not quantitatively assess it.

The letter-C target signature is given in radial coordinates by

$$h(r,\theta) = \begin{cases} A & \text{if} \quad 5 \geq r \geq 6 \quad \text{and} \quad |\theta| > \dfrac{\pi}{4} \\ 0 & \text{otherwise} \end{cases} \tag{42}$$

where A is a normalising constant. This response is shown in Fig. 7(a). The contribution of the target to each pixel is the integral of $h(r,\theta)$ over that pixel. An example of this is shown in Fig. 7(b).

Approximate the target signature by

$$h_l(x,y) = \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_l \\ y_l \end{bmatrix} \tag{43}$$

where x_l and y_l define a cell varying mean.

Denoting the coordinates of cell l as (i_l, j_l), and their radial equivalent coordinates as (r_l, θ_l), then an appropriate cell varying mean is

$$\begin{bmatrix} x_l \\ y_l \end{bmatrix} = \begin{cases} \begin{bmatrix} i_l \\ j_l \end{bmatrix} & \text{if} \quad 5 \geq r_l \geq 6 \\ & \text{and} \quad |\theta_l| > \dfrac{\pi}{4} \\ \mathsf{H}\mathbf{x}_t^m + 5.5\begin{bmatrix} \cos(\theta_l) \\ \sin(\theta_l) \end{bmatrix} & \text{if} \quad 0.9\times 5 \geq r_l \geq 1.1\times 6 \\ & \text{and} \quad |\theta_l| > 0.9\times\dfrac{\pi}{4} \\ \begin{bmatrix} -\infty \\ -\infty \end{bmatrix} & \text{otherwise} \end{cases} \tag{44}$$

with H the common cartesian position-only measurement matrix.

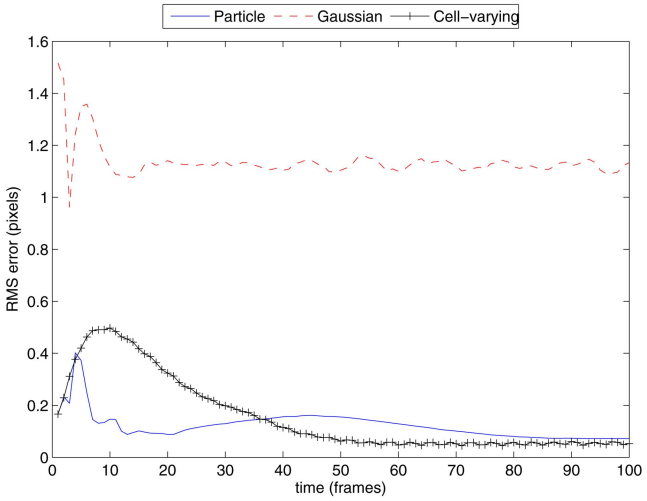


Fig. 8. Non-Gaussian target estimation accuracy.

Fig. 9. Small speedboat target.

The second tier allows for a graduated step down from the C rather than a sudden drop. This gives the algorithm an opportunity to correct for initialisation errors. A similar method was used in [6]. The resulting approximate target response is shown in Fig. 7(c).

The cell-varying approximation to the C psf was implemented and compared with the particle filter algorithm of [6]. Several scenarios were considered in [6], but for brevity only one of them is repeated here. This test scenario contains a single target under constant velocity motion, which the tracker must detect and track. The performance measure is the 2-D RMS estimation error averaged over 100 Monte Carlo trials. For this scenario there was a significant degradation in performance when the C psf was estimated with a single Gaussian compared with the particle filter H-PMHT implementation. Both of these RMS curves are shown in Fig. 8 along with the performance attained with the cell-varying approximation.

It is clear that the cell-varying approximation gives similar performance to the particle filter in this case. Although neither of the algorithm implementations has been optimised, it is informative to look at the computation cost. For 500 particles, the H-PMHT particle implementation took 430 seconds per monte carlo trial, compared with 3.4 seconds for the cell varying Gaussian and 1.7 seconds for the fixed Gaussian. Obviously this analytic approximation method has been able to achieve similar performance as numerical approximation at a fraction of the cost.

The results here show only a simple scenario where it was straightforward to devise a mapping of the non-Gaussian target psf to a cell-varying Gaussian one. In a more realistic situation it may be a more difficult task. Nevertheless, the example illustrates the power of this method: for only a minor increase in computation cost it allows the use of an arbitrary psf, not merely a Gaussian one.

6. DETECTING A SMALL BOAT

An implementation of H-PMHT for detecting a boat in experimental data is now described. The data used was collected by DRDC in Halifax harbour, Canada, using the DRDC Ottawa X-band Wideband Experimental Airborne Radar. The sensor observed a small speedboat, as shown in Fig. 9. A detailed description of the experiment and the data characteristics can be found in [13], [11], [12].

Some salient features of the experimental data will now be discussed and a conventional sequential detect-then-track processing method described. The conventional algorithm is then compared with the H-PMHT output.

6.1. Data Characteristics

Due to the high sensor bandwidth, the range resolution of the sensor was quite fine compared with the physical extent of the target, and the resulting range profile was spread over many range bins. The sensor swept through azimuth and collected range profiles. Again, the azimuth resolution was relatively fine compared with the beam width of the sensor, so the target response was spread over many azimuth bins.

The volume of data from the sensor is relatively large: each scan consists of around 13000 range bins and over 400 azimuth bins. This makes it challenging to process the data in a timely manner and provides an excellent measure of the scalability of the H-PMHT approach to realistic surveillance volumes.

Fig. 10 shows a segment of one scan. Two strong returns are evident near the centre of the image. The rightmost return is from a buoy which happened to be near the target, and the leftmost one is the speedboat. Both buoy and boat have a response that is spread over tens of range bins and tens of azimuths. Numerous other vertical streaks are evident. These are due to sea clutter and are unfortunately target-like. Dividing the environment into background noise and interference spikes, it is clear that the signal to noise ratio is high.

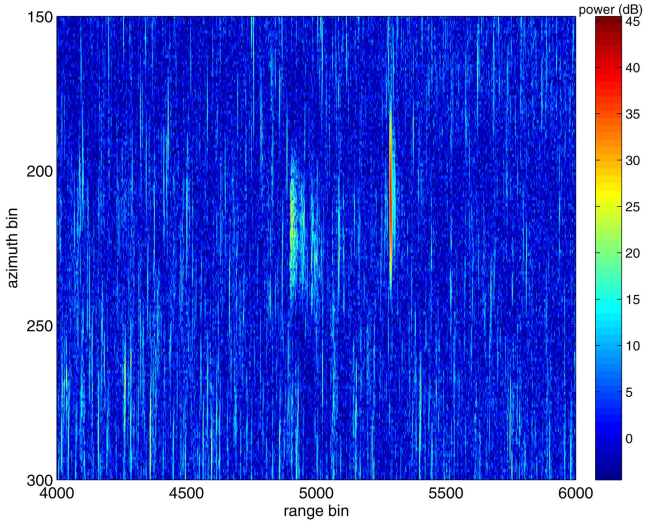


Fig. 10. Segment of a single scan showing target and clutter.

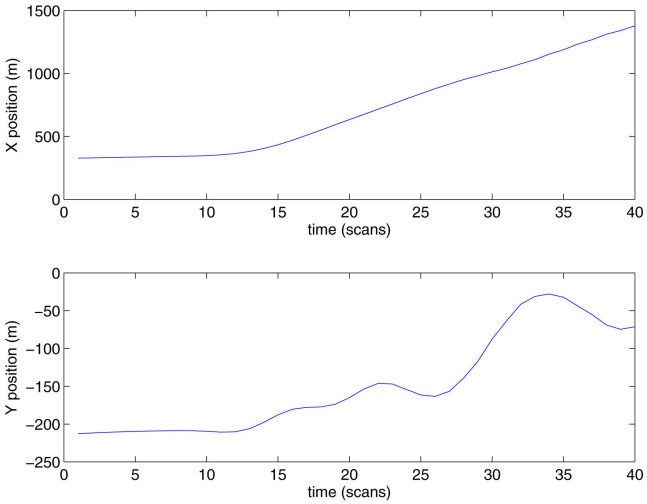


Fig. 11. Target trajectory (local cartesian frame).

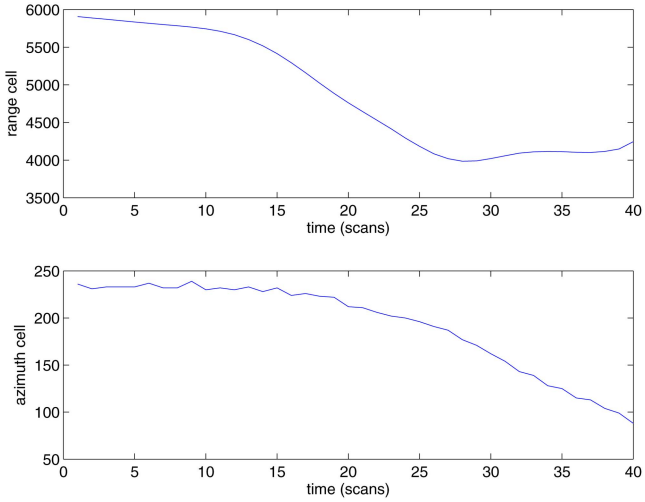


Fig. 12. Target trajectory (sensor frame).

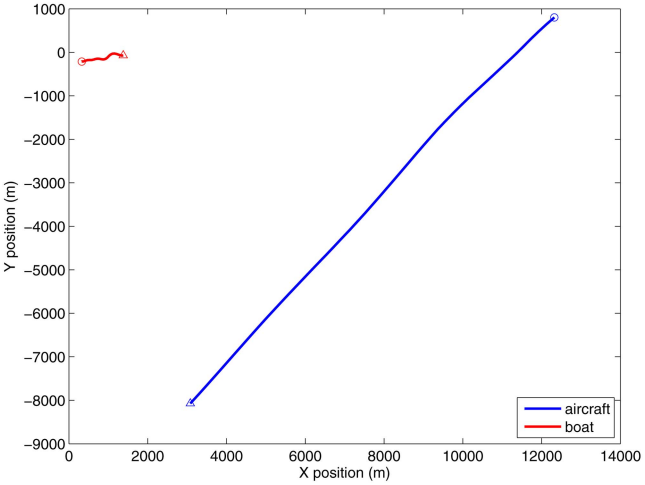


Fig. 13. Target-sensor geometry.

However, the signal to interference ratio is not. In fact the brightest spots on each image are usually sea clutter spikes.

McDonald has characterised the clutter distribution [13], [11], [12]. It was found to be well described by a KA distribution, which has significantly higher tails than the Rayleigh distribution typically assumed by radar applications of TkBD. This is of no direct consequence for H-PMHT since it does not use the data likelihood ratio, but it may be expected to lead to higher false track rates.

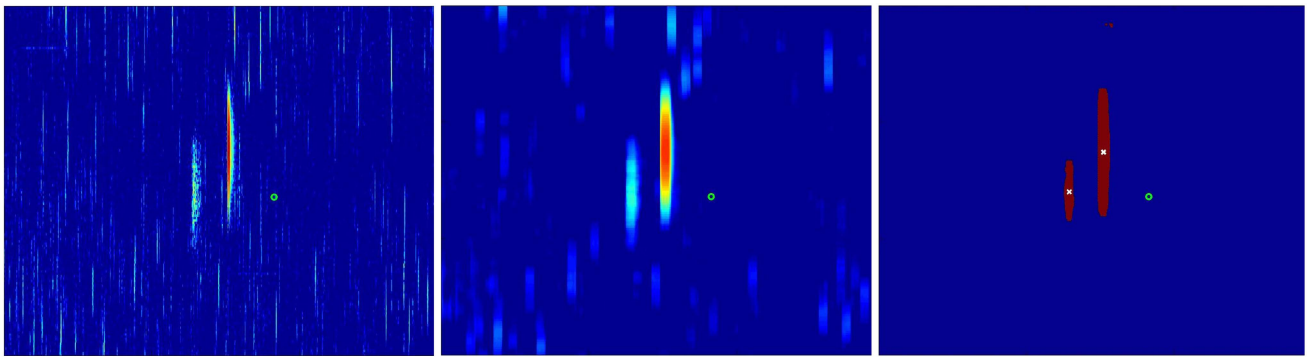
Fig. 10 also highlights that the target response in range is not simply a broad point spread function, but rather shows a much more complicated structure due to scattering off various parts of the boat. The target has the potential to be perceived by the sensor as several targets.

The speedboat was fitted with a GPS logger. However, registration of the GPS data with the radar data has proven difficult [13]. Thus the GPS data is useful as an indication of where the target is, but it is not useful from the perspective of determining target localisation error.

The data of interest is a sequence of 40 frames collected while the speedboat was manoeuvring. The boat was initially almost stationary for around 10 frames and then it accelerated, following a snaking trajectory for the remaining frames. The signal to noise ratio during the first part of the trajectory was relatively poor, but increased when the speedboat moves more quickly.

Fig. 11 shows the speedboat trajectory in local cartesian co-ordinates and Fig. 12 shows the trajectory in the measurement frame (range and azimuth cells). These figures highlight the volatility of the target motion and also demonstrate that the target was moving very rapidly through the sensor field of view. Between scans 10 and 25, the target moves through roughly 1500 range cells.

Fig. 13 shows the overall scenario geometry in local cartesian co-ordinates. Circles mark the starting position of the aircraft and the boat and triangles mark their final position. The aircraft maintained an approximately constant altitude of 300m and moved in an approximately straight path.



(a) (b) (c)

Fig. 14. Target trajectory (sensor frame). (a) Input sensor image. (b) Smoothed sensor image. (c) Segmentation mask.

6.2. Conventional Detection and Tracking

The conventional approach is to apply a single frame detector followed by a point-measurement tracking algorithm. In this case, the size of the target response is relatively large compared with the sensor resolution, so it is appropriate to use an image segmentation based detector. The general strategy is to find the centroids of extended objects and track them using a Probabilistic Data Association Filter (PDAF), similar to [15].

A significant effort was spent in tuning the preprocessing before applying the PDAF. First, it was observed that the target signal was spread over a greater number of image cells than the impulsive noise returns. This motivated the use of spatial averaging, which was implemented by convolving the sensor image with a 10 beam by 5 range uniform rectangular kernel. A threshold was then applied to the smoothed image and adjacent pixels that exceed the threshold using a two-pass labeling algorithm [9] (this can be efficiently implemented in Matlab via the *bwlabel* function). The centroid of each connected set of pixels was determined and this centroid was mapped to a cartesian reference space. Measurements were only retained if the number of foreground pixels (i.e., threshold crossings) associated with a particular object was within an upper and lower tolerance band. This single-frame detection scheme was very CPU-intensive to perform, but yielded a high probability of detection for very few false alarms. From this point onwards, it was straightforward to form tracks, so only a very simple point-tracker was considered.

Fig. 14 illustrates the preprocessing used to feed the conventional tracker. Fig. 14(a) shows the raw radar image with a green circle at the GPS location, which is close to the target but not co-located with it. Fig. 14(b) shows the result of the smoothing filter and (c) shows the output of the segmentation algorithm. The larger target near the middle of the image is the buoy and the smaller one to the left of the buoy is the boat.

Each measurement was transformed into a local cartesian reference frame on the ground using the known sensor location. The tracker assumed a cartesian measurement model with a fixed sensor variance, R, as given in Table I. The X-direction in the local frame was approximately aligned with the radial between the target and sensor (although this varied with time) and so the measurement noise was smaller in the X-direction than the Y-direction.

The PDAF used was a simplified version of the algorithm described in [5] and differed from a text-book algorithm (e.g. [1]) only in that it used target *visibility* to determine the merit of tracks for track maintenance decisions. Uniform clutter was assumed, the target state was Cartesian position and velocity in two dimensions and an almost-constant-velocity model was used. The process noise for this model, Q, is given in Table I.

Thus the measurement association weights were given by

$$\beta_i = \frac{b_i}{\sum_{i=-1}^{n_t} b_i} \tag{45}$$

where n_t is the number of measurements in the scan and

$$b_i = \begin{cases} \dfrac{1 - Pv_{t|t-1}}{Pv_{t|t-1}} & i = -1 \\ 1 - Pd & i = 0 \\ \dfrac{PdV}{|2\pi\mathsf{S}|}\exp\{-\frac{1}{2}d_i^2\} & 0 < i \le n_t \end{cases} \tag{46}$$

with Pd the probability of detection, V the surveillance volume, S the innovation covariance, $d_i^2 = (\mathsf{H}\mathbf{x} - \mathbf{z}_i)^{\mathrm{T}}\mathsf{S}^{-1}(\mathsf{H}\mathbf{x} - \mathbf{z}_i)$ the statistical distance between the track and measurement i, and $Pv_{t|t-1}$ the predicted target visibility, given by

$$Pv_{t|t-1} = (1 - P_{\text{death}})Pv_{t-1|t-1} + P_{\text{rebirth}}(1 - Pv_{t-1|t-1}) \tag{47}$$

where the probability of target death and rebirth are tuning parameters given in Table I. The updated target visibility is given by $Pv_{t|t} = 1 - \beta_{-1}$ and is used as the basis of track management decisions. Similar to the H-PMHT track manager, candidate tracks are formed using 2-point differencing.

Table I lists the parameters of the conventional detection and tracking approach.

TABLE I
Conventional Detection and Tracking Parameters

Detector Parameters	
minimum segment size	75 pixels
maximum segment size	none
segmentation threshold	$2 \times$ image mean
Tracker Parameters	
track state vector	$[x, \dot{x}, y, \dot{y}]^{\mathrm{T}}$ x and y in metres $\dot{x}$ and $\dot{y}$ in metres per frame
measurement vector	$[x, y]^{\mathrm{T}}$ in metres
R	$\begin{bmatrix} 400 & 0 \\ 0 & 900 \end{bmatrix}$
Q	$100 \begin{bmatrix} \mathrm{Q2} & 0 \\ 0 & \mathrm{Q2} \end{bmatrix}$
Q2	$\begin{bmatrix} \frac{1}{3}\Delta_t^3 & \frac{1}{2}\Delta_t^2 \\ \frac{1}{2}\Delta_t^2 & \Delta_t \end{bmatrix}$
P_{death}	0.012
P_{rebirth}	0
initial visibility	$Pv_0 = 0.5$
promotion threshold	$Pv_{t\|t} > 0.6$ for any $t > 4$ frames

TABLE II
H-PMHT Parameters

Detector Parameters	(used to form candidate tracks)
minimum segment size	75 pixels
maximum segment size	none
segmentation threshold	$3 \times$ image mean
H-PMHT(rb) Parameters	
track state vector	$[r, \dot{r}, b, \dot{b}]^{\mathrm{T}}$ r and b in cells $\dot{r}$ and $\dot{b}$ in cells per frame
Q	$\begin{bmatrix} 100\mathrm{Q2} & 0 \\ 0 & 10^4\mathrm{Q2} \end{bmatrix}$
H-PMHT(xy) Parameters	
track state vector	$[x, \dot{x}, y, \dot{y}]^{\mathrm{T}}$ x and y in metres $\dot{x}$ and $\dot{y}$ in metres per frame
Q	$1.2 \times 10^5 \begin{bmatrix} \mathrm{Q2} & 0 \\ 0 & \mathrm{Q2} \end{bmatrix}$
Common Parameters	
measurement vector	$[r, b]^{\mathrm{T}}$ in cells
R	$\begin{bmatrix} 400 & 0 \\ 0 & 900 \end{bmatrix}$
confirmation threshold	4 frames greater than 20 dB

6.3. Specific H-PMHT models and Parameters

The H-PMHT was implemented as a time-recursive filter for this analysis, that is, there was no batch; each scan was processed sequentially and only one scan was available to the algorithm at a time. The state estimates were not smoothed.

Two processing strategies for H-PMHT were considered. In the first, the movement of the sensor was ignored, and the target state was modeled in the measurement frame. That is, the state was pixels and pixels per frame. This makes life much easier for implementation, and the separable point spread function expressions may be used. This version is referred to as H-PMHT(rb) since the target state is in range-beam space.

The second strategy was to model the target state in Cartesian coordinates on the ground and use the cell-dependent non-linear method to relate the target state to the sensor image. This approach has much higher implementation complexity but should more accurately model the true system. This version is referred to as H-PMHT(xy).

The H-PMHT(rb) software was a Matlab implementation of the equations derived in Section 3.4. There are two integrals that need to be implemented and these are vector versions of the per-cell contribution of each target in (6)

$$P_l^m(\mathbf{x}_t^m) = \int_{B_l} G^m(\tau \mid \mathbf{x}_t^m) d\tau$$

and the cell-level centroid in (14)

$$\tilde{z}_{tl}^m = \frac{1}{P_l^m(x_t^m)} \int_{B_l} \tau G^m(\tau \mid x_t^m) d\tau.$$

Appendix A.2 provides some discussion on implementing these integrals. Besides those two expressions, the remaining software is a direct encoding of Section 2.4.

The background was assumed to be uniform and the target was assumed to be a simple Gaussian with a diagonal covariance matrix as given in Table II. The target was more spread in azimuth than range, so the azimuth variance was higher.

The target model was an almost constant velocity model in the plane. That is, the target motion was approximated with a constant rate of movement through range and azimuth bins. The target state was in units of bins for position and bins-per-scan for velocity. This was transformed into ground units as a post-processing stage for comparison with the other trackers and the ground truth. The process noise variance used is given in Table II, also using bins and scans as units.

The efficiency measures described in Section 3 were employed, including gating and a per-target convergence test. A maximum of 10 EM iterations was performed per frame. The single frame detector used for the PDA was also used here to seed candidate tracks which were initialised using two-point differences.

Candidates were promoted if they had an estimated SNR of greater than 20 dB for four frames and terminated if they had an estimated SNR of less than -5 dB for two consecutive frames.

The H-PMHT(xy) software was a Matlab implementation of the cell-varying psf equations in Section 5.2. This algorithm used the same measurement covariance

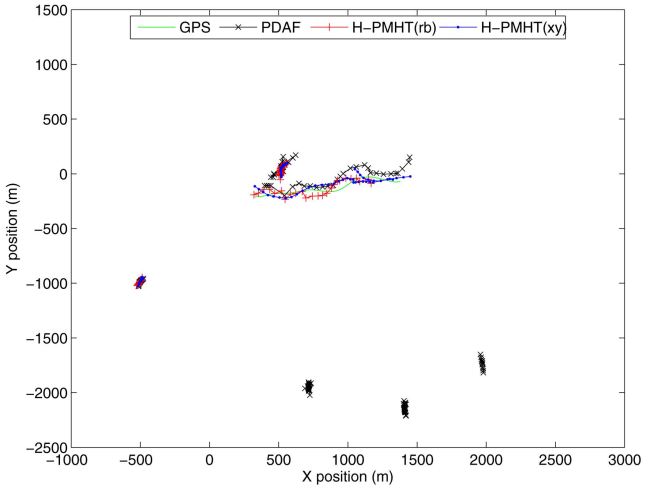


Fig. 15. Small speedboat target.

matrix as the measurement-space implementation and also used an almost constant velocity target model. However, the target state was in different units, namely metres, so a different process noise covariance matrix was required, as given in Table II. The same track management rules were used as for the measurement-space implementation.

For the H-PMHT(xy), the position part of the state vector was in metres, but the measurement frame was in range and azimuth cells. The estimator was therefore implemented as an EKF as described in Section 5.2. The measurement model included the scaling and offset from range in metres to range in cells and similarly for azimuth. The per-cell contribution of the target P_l^m was calculated as described in Appendix A.2, but the cell-centroids were replaced by the cell-dependent terms in Section 5.2.

6.4. Results

The tracking outputs of the PDAF based approach and the two H-PMHT approaches are now presented. Each are compared with GPS data collected from the target of interest.

Fig. 15 shows all of the output tracks from each of the three trackers overlaid with the GPS measurements. The target tracks are roughly centred in the plot and move from left to right. As mentioned earlier, there was a buoy that was coincidentally in the region, near to the target. The buoy location is slightly north of the target starting position. There are no other known targets in the area, and the other tracks are assumed to be false.

All three trackers were able to detect the speedboat and the buoy. The PDAF tracker shows a number of false tracks to the south of the area, which is closer in range to the sensor. At close range, the clutter spikes are more prominent and so the tracker forms false tracks.

By manual inspection, the tracks that have followed the target were determined and are shown on Fig. 16. It is clear that none of the trackers were able to detect the target during the initial period where it was stationary.

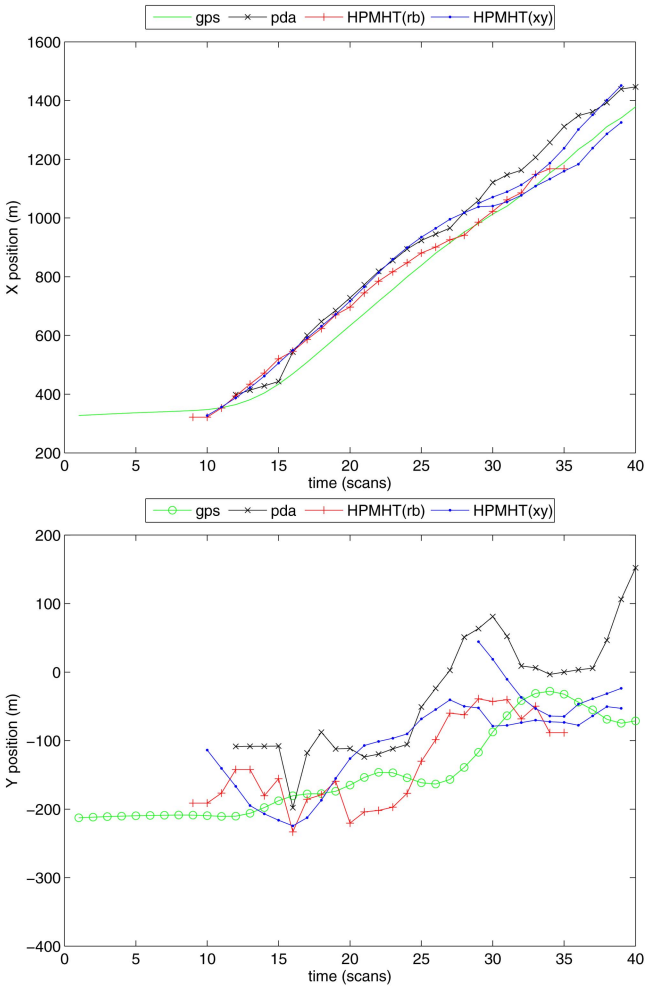


Fig. 16. Small speedboat target.

During this time it is not possible to detect the target by eye in the data either: the received SNR is very low. Once it starts moving and the SNR improves, the PDAF tracker took a little longer to establish track.

At approximately scan 32, the target performed a manoeuvre that caused the PDAF tracker to diverge. Unfortunately, soon after this the target left the sensor coverage area in scan 39 and there is insufficient data to determine whether the PDAF would have recovered from the error. The measurement-space H-PMHT(rb) tracker lost the target at the same time. The ground-space H-PMHT(xy) followed the target until the second to last scan. However during the last ten scans it produced a duplicate track on the target. The original track started out following the centre of mass of the target response, but later it drifted slightly to focus on the short-range component (recall that the target response is a complicated superposition of multiple partially resolved scatterers). This caused the tracker to form a second track on the longer-range component of the target response.

Fig. 17 shows tracks that were manually determined to be following the buoy. The buoy has an extremely high SNR and is easily tracked by all of the algorithms.

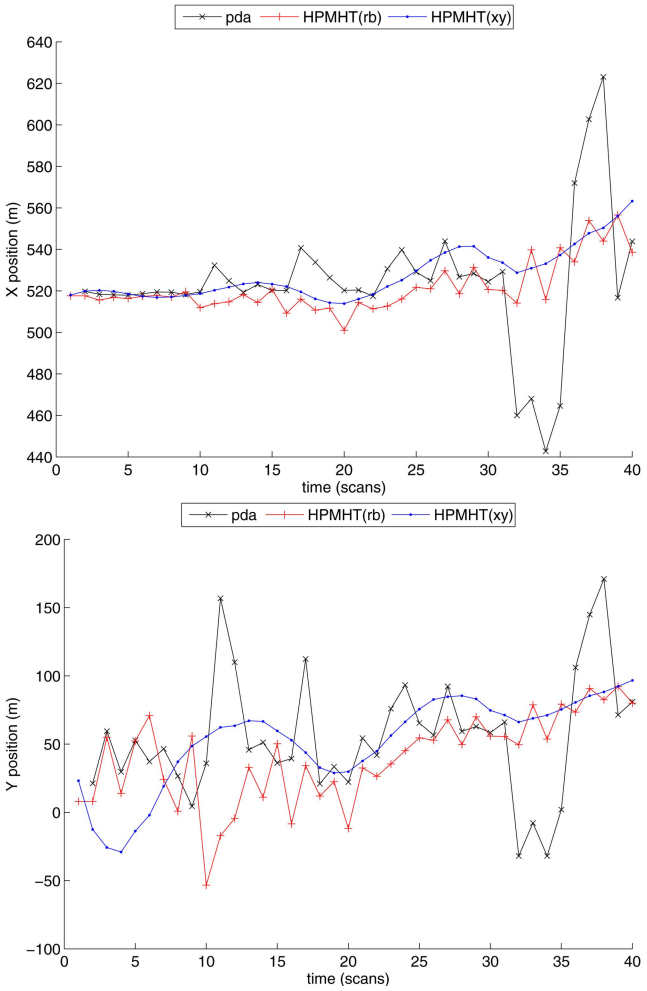


Fig. 17. Strong buoy target of opportunity.

At times, the peak level of the buoy is 40 dB higher than the boat. However, the H-PMHT(xy) track is considerably smoother than the others, especially the PDAF which shows several large excursions. This is partly because the PDAF was tuned with a relatively high process noise to cope with the high manoeuvrability of the speedboat and partly because the detection process reports the centroid of a group of connected pixels, which can be highly variable.

The implementation of the algorithms was not optimised, so the CPU time spent by each is not a reliable measure of performance. Nevertheless, figures are given as a qualitative comparison. The H-PMHT(rb) was the fastest algorithm, requiring 280 CPU seconds to run under Matlab on a quad core PC. For the PDAF approach, the single-frame detector was orders of magnitude more expensive than the PDAF tracker. This is because of the very large number of pixels in the sensor image and the double-pass clustering required to extract detections. In fact, the single-frame detector took around twice the computation effort of the measurement-space H-PMHT. Substantial experimentation showed that simpler detectors for the PDAF lead to an abundance of false tracks. The PDAF total cost was 478 CPU seconds.

The ground-space H-PMHT(xy) was the slowest at 575 CPU seconds. Since the sensor image is so large, the available memory limited the number of intermediate variables that could be stored. In particular, for candidate tracks, it was not feasible to store the P_l^m values, which meant that they had to be calculated twice for each EM iteration. These values alone can amount to around half a gigabyte of storage for a modest number of tracks. Thus the trade-off between memory and computation resulted in a longer execution time for the ground-space H-PMHT. For the sensor-space H-PMHT, these quantities can be stored as their marginal vectors, which are a fraction of a percent of the size. An optimised implementation of the ground-space algorithm could alleviate much of this overhead.

7. SUMMARY

This article has reviewed the Histogram PMHT algorithm from an implementation perspective. It addresses some of the practical issues arising from real applications: efficient implementation, track management, and non-linearity. The purpose has been to demonstrate that the encouraging simulation results obtained in recent comparisons on thumbnail images with benevolent targets do translate to more realistic conditions.

The H-PMHT algorithm was applied to an experimental data set collected by DRDC containing a manoeuvring speedboat amongst highly challenging sea clutter and close to a strong secondary target: a buoy. The buoy return was at times 40 dB higher than the speedboat, which had a peak signal-to-interference ratio of less than 0 dB. Nevertheless, the H-PMHT algorithm was able to detect the boat and follow its manoeuvres while forming almost no false tracks. The H-PMHT demonstrated a good ability to track multiple closely spaced targets with a high dynamic range.

The experimental data was also processed via a more conventional method, using image segmentation and a PDAF. The conventional output took marginally longer to establish track on the target and produced more false tracks. The conventional detector was very computationally demanding so the overall cost of the H-PMHT was less than the conventional approach.

The good performance of H-PMHT illustrates that the algorithm is not sensitive to the assumed target model, since the speedboat target response was a complex interaction of multiple scatterers rather than a simple Gaussian. It was also not sensitive to target-like clutter responses in the experimental data. Although the H-PMHT does not explicitly model the clutter amplitude likelihood, intuition suggests that the maths must make some implicit modeling assumption, which would doubtlessly be broken by this data. Initialisation was also poor since the target moved very quickly through the image at an average of one hundred range bins per scan.

In addition to the experimental case study, several important results were introduced. An efficient matrix representation of the algorithm was derived for two-dimensional sensor images and when combined with appropriate gating was demonstrated to give an algorithm independent of sensor image size for a fixed resolution. Track management was discussed in detail, a function that recent papers describe as lacking from H-PMHT. Finally a novel measurement model was introduced as a method for dealing with non-Gaussian targets by transforming them into a Gaussian target with a spatially varying mean. This model was demonstrated on a simple simulation and used to compensate for ownship motion in the experimental data.

This article has demonstrated that H-PMHT is an effective algorithm for automatically extracting multiple targets against a very challenging background and can process very high data volumes at a reasonable rate.

8. ACKNOWLEDGMENT

The author would like to thank Defence Research and Development Canada for providing the test data set and in particular Dr Michael McDonald for his assistance in interpreting and processing the data.

APENDIX

A.1. Cell-varying Point Spread Function Derivation

Consider the measurement component of the target auxiliary function (12)

$$\sum_{t=1}^{T}\sum_{l\in\mathcal{S}}\pi_t^{m(i)}\bar{z}_{tl}\int_{B_l} G_l^m(\tau\mid\mathbf{x}_t^{m(i)})\log\{G_l^m(\tau\mid\mathbf{x}_t^m)\}d\tau. \tag{48}$$

For clarity, consider one term from the temporal summation and suppress the target and time indices, m and t. Abbreviate $G_l^{(i)}\equiv G_l^m(\tau\mid\mathbf{x}_t^{m(i)})$. Using the cell-dependent Gaussian model, one time-slice of (48) becomes

$$\begin{aligned}
&\pi^{(i)}\sum_{l\in\mathcal{S}}\bar{z}_l\int_{B_l} G_l^{(i)}\log\{\mathcal{N}(\tau;\mathbf{h}_l,\mathsf{R})\}d\tau\\
&\quad=-\frac{1}{2}\pi^{(i)}\sum_{l\in\mathcal{S}}\bar{z}_l\int_{B_l} G_l^{(i)}(\tau-\mathbf{h}_l)^{\mathrm{T}}\mathsf{R}^{-1}(\tau-\mathbf{h}_l)d\tau+C\\
&\quad=-\frac{1}{2}\pi^{(i)}\sum_{l\in\mathcal{S}}\bar{z}_l\left\{\int_{B_l} G_l^{(i)}\mathbf{h}_l^{\mathrm{T}}\mathsf{R}^{-1}\mathbf{h}_l d\tau\right.\\
&\qquad\qquad\left.-2\int_{B_l} G_l^{(i)}\mathbf{h}_l^{\mathrm{T}}\mathsf{R}^{-1}\tau d\tau\right\}+C\\
&\quad=-\frac{1}{2}\pi^{(i)}\sum_{l\in\mathcal{S}}\bar{z}_l\left\{\mathbf{h}_l^{\mathrm{T}}\mathsf{R}^{-1}\mathbf{h}_l\int_{B_l} G_l^{(i)}d\tau\right.\\
&\qquad\qquad\left.-2\mathbf{h}_l^{\mathrm{T}}\mathsf{R}^{-1}\int_{B_l} G_l^{(i)}\tau d\tau\right\}+C\\
&\quad=-\frac{1}{2}\pi^{(i)}\left\{\sum_{l\in\mathcal{S}}\bar{z}_l P_l^m(\mathbf{x}^{(i)})\mathbf{h}_l(\mathbf{x})^{\mathrm{T}}\mathsf{R}^{-1}\mathbf{h}_l(\mathbf{x})\right.\\
&\qquad\qquad\left.-2\sum_{l\in\mathcal{S}}\bar{z}_l P_l^m(\mathbf{x}^{(i)})\mathbf{h}_l(\mathbf{x})^{\mathrm{T}}\mathsf{R}^{-1}\tilde{\mathbf{z}}_l\right\}+C
\end{aligned} \tag{49}$$

where C soaks up the terms that are constant with respect to $\mathbf{x}$, and

$$\tilde{\mathbf{z}}_l=\frac{1}{P_l^m(\mathbf{x}^{(i)})}\int_{B_l}\mathcal{N}(\tau;\mathbf{h}_l(\mathbf{x}^{(i)}),\mathsf{R})\tau d\tau \tag{50}$$

that is, the cell-level centroid defined in (14).

Let

$$w=\sum_{l\in\mathcal{S}}w_l=\sum_{l\in\mathcal{S}}\bar{z}_l P_l^m(\mathbf{x}^{(i)}). \tag{51}$$

The expression in (49) can be represented as a single quadratic by completing the square, i.e.,

$$\begin{aligned}
&-\frac{1}{2}\pi^{(i)}\left\{\sum_{l\in\mathcal{S}}w_l\mathbf{h}_l(\mathbf{x})^{\mathrm{T}}\mathsf{R}^{-1}\mathbf{h}_l(\mathbf{x})-2\sum_{l\in\mathcal{S}}w_l\mathbf{h}_l(\mathbf{x})^{\mathrm{T}}\mathsf{R}^{-1}\tilde{\mathbf{z}}_l\right\}+C\\
&\quad=-\frac{1}{2}(\tilde{\mathbf{z}}-\tilde{\mathbf{h}}(\mathbf{x}))^{\mathrm{T}}\tilde{\mathsf{R}}^{-1}(\tilde{\mathbf{z}}-\tilde{\mathbf{h}}(\mathbf{x}))+C'
\end{aligned} \tag{52}$$

where C' is another constant and the terms $\tilde{\mathbf{z}}$, $\tilde{\mathbf{h}}(\mathbf{x})$, and $\tilde{\mathsf{R}}$ are to be determined. These terms allow the measurement component of the auxiliary function to be expressed as an equivalent point-measurement.

Choose

$$\tilde{\mathsf{R}}^{-1}=\pi^{(i)}\sum_{l\in S}w_l\mathsf{R}^{-1}=\pi^{(i)}w\mathsf{R}^{-1}. \tag{53}$$

Equating the quadratic terms gives

$$\begin{aligned}
-\frac{1}{2}\tilde{\mathbf{h}}(\mathbf{x})^{\mathrm{T}}\tilde{\mathsf{R}}^{-1}\tilde{\mathbf{h}}(\mathbf{x})&=-\frac{1}{2}\pi^{(i)}\sum_{l\in S}w_l\mathbf{h}_l(\mathbf{x})^{\mathrm{T}}\mathsf{R}^{-1}\mathbf{h}_l(\mathbf{x})\\
\pi^{(i)}w\tilde{\mathbf{h}}(\mathbf{x})^{\mathrm{T}}\mathsf{R}^{-1}\tilde{\mathbf{h}}(\mathbf{x})&=\pi^{(i)}\sum_{l\in S}w_l\mathbf{h}_l(\mathbf{x})^{\mathrm{T}}\mathsf{R}^{-1}\mathbf{h}_l(\mathbf{x})\\
\tilde{\mathbf{h}}(\mathbf{x})^{\mathrm{T}}\mathsf{R}^{-1}\tilde{\mathbf{h}}(\mathbf{x})&=\frac{1}{w}\sum_{l\in S}w_l\mathbf{h}_l(\mathbf{x})^{\mathrm{T}}\mathsf{R}^{-1}\mathbf{h}_l(\mathbf{x})
\end{aligned} \tag{54}$$

which can be simplified to

$$\tilde{\mathbf{h}}(\mathbf{x})[j]=\sqrt{\frac{1}{w}\sum_{l\in S}w_l\mathbf{h}_l(\mathbf{x})[j]^2} \tag{55}$$

if R is diagonal. Here, $\mathbf{h}_l(\mathbf{x})[j]$ is the jth element of vector $\mathbf{h}_l(\mathbf{x})$.

Similarly, equating the linear terms

$$\begin{aligned}
\tilde{\mathbf{h}}(\mathbf{x})^{\mathrm{T}}\tilde{\mathsf{R}}^{-1}\tilde{\mathbf{z}}&=\pi^{(i)}\sum_{l\in S}w_l\mathbf{h}_l(\mathbf{x})^{\mathrm{T}}\mathsf{R}^{-1}\tilde{z}_l\\
\tilde{\mathbf{h}}(\mathbf{x})^{\mathrm{T}}\mathsf{R}^{-1}\tilde{\mathbf{z}}&=\frac{1}{w}\sum_{l\in S}w_l\mathbf{h}_l(\mathbf{x})^{\mathrm{T}}\mathsf{R}^{-1}\tilde{z}_l
\end{aligned} \tag{56}$$

which simplifies to

$$\tilde{\mathbf{z}}[j]=\frac{1}{w\tilde{\mathbf{h}}(\mathbf{x})[j]}\sum_{l\in S}w_l\mathbf{h}_l(\mathbf{x})^{\mathrm{T}}[j]\tilde{z}_l[j] \tag{57}$$

if R is diagonal.

A.2. H-PMHT integral simplifications

Implementation of H-PMHT requires the encoding of two integrals, namely the per-cell contribution of each target in (6)

$$P_l^m(\mathbf{x}_t^m) = \int_{B_l} G^m(\tau \mid \mathbf{x}_t^m) d\tau$$

and the cell-level centroid in (14)

$$\tilde{z}_{tl}^m = \frac{1}{P_l^m(\mathbf{x}_t^m)} \int_{B_l} \tau G^m(\tau \mid \mathbf{x}_t^m) d\tau.$$

Here analytic expressions are developed for each of these suitable for software implementation. It will be assumed that the sensor cells, B_l, conform to a regular grid and that the target psf is Gaussian, i.e.,

$$G^m(\tau \mid \mathbf{x}_t^m) = \frac{1}{|2\pi \mathsf{R}|} \exp\left\{ -\frac{1}{2} (\mathsf{H}\mathbf{x}_t^m - \tau)^{\mathrm{T}} \mathsf{R}^{-1} (\mathsf{H}\mathbf{x}_t^m - \tau) \right\}.$$

The integral in (6) is simply the area under the pdf for cell l and can be evaluated using erf functions since $G^m(\cdot)$ is Gaussian. However, these are relatively costly to evaluate. For efficiency, the implementation in this paper makes the approximation that $G^m(\cdot)$ is constant over B_l and so the approximate per-cell contribution is

$$P_l^m(\mathbf{x}_t^m) \approx \frac{\Delta_\tau}{|2\pi \mathsf{R}|} \exp\left\{ -\frac{1}{2} (\mathsf{H}\mathbf{x}_t^m - \bar{\tau}_l)^{\mathrm{T}} \mathsf{R}^{-1} (\mathsf{H}\mathbf{x}_t^m - \bar{\tau}_l) \right\}$$

with $\bar{\tau}_l$ the centre of cell l and Δ_τ the cell area, which is the same for all cells because of the regular grid. This approximation is reasonable when R is large compared to the cell size and the psf is thus slowly varying, as is the case for the application considered in this paper.

The cell-level centroid is the mean of the target psf over the cell. An expression is now developed for this for a one-dimensional psf. The two-dimensional version is built from two one-dimensional terms as described in Section 3. For simplicity of notation, assume that the target state is simply position, so H is unity. The extension to constant velocity is simply a matter of book-keeping. Since the psf is one-dimensional, R is a scalar.

Since $G^m(\cdot)$ is Gaussian it follows that

$$G^m(\tau \mid \mathbf{x}_t^m)' = \frac{dG^m(\tau \mid \mathbf{x}_t^m)}{d\tau} = G^m(\tau \mid \mathbf{x}_t^m) \left(\frac{\mathbf{x}_t^m}{\mathsf{R}} - \frac{\tau}{\mathsf{R}} \right)$$

which implies

$$\tau G^m(\tau \mid \mathbf{x}_t^m) = \mathbf{x}_t^m G^m(\tau \mid \mathbf{x}_t^m) - \mathsf{R} G^m(\tau \mid \mathbf{x}_t^m)'.$$

Substituting this back into the centroid definition gives

$$\begin{aligned} \tilde{z}_{tl}^m &= \frac{\mathbf{x}_t^m}{P_l^m} \int_{B^l} G^m(\tau \mid \mathbf{x}_t^m) d\tau - \frac{\mathsf{R}}{P_l^m} \int_{B^l} G^m(\tau \mid \mathbf{x}_t^m)' d\tau, \\ &= \mathbf{x}_t^m - \frac{\mathsf{R}}{P_l^m} \left[G^m\left(\bar{\tau}_l + \frac{\Delta_\tau}{2} \,\middle|\, \mathbf{x}_t^m \right) - G^m\left(\bar{\tau}_l - \frac{\Delta_\tau}{2} \,\middle|\, \mathbf{x}_t^m \right) \right] \end{aligned}$$

which is easily implemented in software.

REFERENCES

[1] Y. Bar-Shalom and X. R. Li
Multitarget-multisensor tracking: principles and techniques.
Storrs, CT: YBS, 1995.

[2] Y. Barniv
Dynamic programming algorithm for detecting dim moving targets.
In Y. Bar-Shalom, Ed., *Multitarget-Multisensor Tracking: Advanced Applications*, Artech House, 1990, ch. 4.

[3] S. S. Blackman and R. Popoli
Design and Analysis of Modern Tracking Systems.
Norwood, MA: Artec House, 1999.

[4] Y. Boers and J. N. Driessen
Particle filter based detection for tracking.
In *Proceedings of the American Control Conference*, Arlington, VA, June 2001, 4393–4397.

[5] S. B. Colegrove and S. J. Davey
PDAF with multiple clutter regions and target models.
IEEE Transactions on Aerospace and Electronic Systems, **39**, 1 (Jan. 2003), 110–124.

[6] S. J. Davey
Histogram PMHT with particles.
In *Proceedings of the 14th International Conference on Information Fusion*, July 2011.

[7] S. J. Davey and D. A. Gray
Integrated track maintenance for the PMHT via the hysteresis mdoel.
IEEE Transactions on Aerospace and Electronic Systems, **43**, 1 (Jan. 2007), 93–111.

[8] S. J. Davey, M. G. Rutten, and B. Cheung
A comparison of detection performance for several track-before-detect algorithms.
EURASIP Journal on Advances in Signal Processing, 2008.

[9] R. M. Haralick and L. G. Shapiro
Computer and Robot Vision.
Addison-Wesley, 1992, vol. 1.

[10] T. E. Luginbuhl, Y. Sun, and P. Willett
A track management system for the PMHT algorithm.
In *Proceedings of the 4th International Conference on Information Fusion*, Montreal, Canada, Aug. 2001.

[11] M. McDonald and B. Balaji
Continuous-discrete filtering for dim manoeuvring maritime targets.
July 2007.

[12] M. McDonald and B. Balaji
Impact of measurement model mismatch on nonlinear track-before-detect performance.
In *Radar Conference, 2008* (RADAR '08, IEEE), May 2008, 1–6.

[13] M. McDonald and B. Balaji
Track-before-detect using swerling 0, 1, and 3 target models for small manoeuvring maritime targets.
Journal on Advances in Signal Processing (EURASIP), 2008.

[14] R. M. Burczewski and N. C. Mohanty
Detection of moving optical objects.
In *International Telemetering Conference*, Los Angeles, CA, 1978, 325–330.

[15] E. Oron, A. Kumar, and Y. Bar-Shalom
Precision tracking with segmentation for imaging sensors.
IEEE Transactions on Aerospace and Electronic Systems, **29**, 3 (July 1993), 977–987.

[16] A. G. Pakfiliz and M. Efe
Multi-target tracking in clutter with histogram probabilistic multi-hypothesis tracker.
In *IEEE Conference on Systems Engineering*, 2005, 137–142.

[17] S. C. Pohlig
An algorithm for detection of moving optical targets.
IEEE Transactions on Aerospace and Electronic Systems, **25**, 1 (Jan. 1989), 56–63.

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

[19] D. J. Salmond and H. Birch
A particle filter for track-before-detect.
In *Proceedings of the American Control Conference*, Arlington, VA, June 2001, 3755–3760.

[20] M. C. Smith and E. M. Winter
On the detection of target trajectories in a mutli target environment.
In *IEEE Conference on Decision and Control*, San Diego, CA, 1978, 1189–1194.

[21] L. D. Stone, C. A. Barlow, and T. L. Corwin
Bayesian Multiple Target Tracking.
Artech House, 1999.

[22] R. L. Streit
Tracking on intensity-modulated data streams.
NUWC, RI: Technical report 11221, May 2000.

[23] R. L. Streit
Tracking targets with specified spectra using the H-PMHT algorithm.
NUWC, RI: Technical report 11291, June 2001.

[24] R. L. Streit, M. L. Graham, and M. J. Walsh
Multitarget tracking of distributed targets using histogram-PMHT.
Digital Signal Processing, **12**, 2 (July 2002).

[25] B-N. Vo, B-T. Vo, N-T. Pham, and D. Suter
Joint detection and estimation of multiple objects from image observations.
IEEE Transactions on Signal Processing, **58**, 10 (Oct. 2010), 5129–5141.

[26] M. Wieneke and S. J. Davey
Histogram PMHT with target extent estimates based on random matrices.
In *Proceedings of the 14th International Conference on Information Fusion*, July 2011.

Samuel Davey received the Bachelor of Engineering, Master of Mathematical Science, and Ph.D. degrees from the University of Adelaide, Australia, in 1996, 1999, and 2003, respectively.

Since 1995 he has worked for the Defence Science and Technology Organisation, Australia, in the areas of target tracking, tracker performance assessment, and multi-sensor fusion. He is also a Visiting Research Fellow at the University of Adelaide.