REPORT DOCUMENTATION PAGE

1

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
	New Reprint	-

4. TITLE AND SUBTITLE

The ML-PMHT MultistaticTracker for SharplyManeuvering Targets

5a. CONTRACT NUMBER
W911NF-10-1-0369

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
611102

6. AUTHORS

STEVEN SCHOENECKER, Naval Undersea Warfare Center, PETER WILLETT, Fellow, IEEE, YAAKOV BAR-SHALOM, Fellow, IEEE, University of Connecticut

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES

University of Connecticut - Storrs
438 Whitney Road Ext., Unit 1133

Storrs, CT 06269 -1133

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)

U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
57823-CS.65

12. DISTRIBUTION AVAILIBILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT

The maximum likelihood probabilistic multi-hypothesis tracker (ML-PMHT) is applied to a benchmark multistatic active sonar scenario with multiple targets, multiple sources, and multiple receivers. We first compare the performance of the tracker on this scenario when it is applied in Cartesian measurement space, a typical implementation for many

15. SUBJECT TERMS

Maneuvering Target tracking

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Yaakov Bar-Shalom
UU	UU	UU	UU		19b. TELEPHONE NUMBER 860-486-4823

Report Title

The ML-PMHT MultistaticTracker for SharplyManeuvering Targets

ABSTRACT

The maximum likelihood probabilistic multi-hypothesis tracker (ML-PMHT) is applied to a benchmark multistatic active sonar scenario with multiple targets, multiple sources, and multiple receivers. We first compare the performance of the tracker on this scenario when it is applied in Cartesian measurement space, a typical implementation for many trackers, against its performance in delay-bearing measurement space, where the measurement uncertainty is more accurately represented. ML-PMHT is a batch tracker, and the motion of a target being tracked must be given a parameterization that describes the motion of the target throughout the batch. In the scenario in which we apply the tracker, the majority of target returns have low amplitudes (i.e., the targets are low-observable), which makes the choice of a batch tracker very appropriate. In prior work, ML-PMHT was implemented with a straight-line parameterization to describe target motion. However, in order to track maneuvering targets, the tracker was implemented in a sliding-batch fashion under the assumption that a maneuvering track could be approximated as a series of short straight lines. Here, we augment the straight-line parameterization by a maneuver–a single course change within the batch–that allows ML-PMHT to follow even sharply maneuvering targets, and we apply it in both Cartesian and delay-bearing measurement space. We also implement this maneuvering-model parameterization with both a fixed batch-length implementation as well as a variable batch-length implementation. Finally, we develop an expression for the Cram´er-Rao lower bound (CRLB) for the maneuvering-model parameterization and show that the ML-PMHT tracker with the maneuvering-model parameterization is an efficient estimator.

REPORT DOCUMENTATION PAGE (SF298)
(Continuation Sheet)

Continuation for Block 13

ARO Report Number 57823.65-CS
The ML-PMHT MultistaticTracker for SharplyMa...

Block 13: Supplementary Note

Approved for public release; distribution is unlimited.

The ML-PMHT Multistatic Tracker for Sharply Maneuvering Targets

STEVEN SCHOENECKER
Naval Undersea Warfare Center

PETER WILLETT, Fellow, IEEE
YAAKOV BAR-SHALOM, Fellow, IEEE
University of Connecticut

The maximum likelihood probabilistic multi-hypothesis tracker (ML-PMHT) is applied to a benchmark multistatic active sonar scenario with multiple targets, multiple sources, and multiple receivers. We first compare the performance of the tracker on this scenario when it is applied in Cartesian measurement space, a typical implementation for many trackers, against its performance in delay-bearing measurement space, where the measurement uncertainty is more accurately represented. ML-PMHT is a batch tracker, and the motion of a target being tracked must be given a parameterization that describes the motion of the target throughout the batch. In the scenario in which we apply the tracker, the majority of target returns have low amplitudes (i.e., the targets are low-observable), which makes the choice of a batch tracker very appropriate. In prior work, ML-PMHT was implemented with a straight-line parameterization to describe target motion. However, in order to track maneuvering targets, the tracker was implemented in a sliding-batch fashion under the assumption that a maneuvering track could be approximated as a series of short straight lines. Here, we augment the straight-line parameterization by a maneuver—a single course change within the batch—that allows ML-PMHT to follow even sharply maneuvering targets, and we apply it in both Cartesian and delay-bearing measurement space. We also implement this maneuvering-model parameterization with both a fixed batch-length implementation as well as a variable batch-length implementation. Finally, we develop an expression for the Cramér-Rao lower bound (CRLB) for the maneuvering-model parameterization and show that the ML-PMHT tracker with the maneuvering-model parameterization is an efficient estimator.

Manuscript received July 2, 2012; revised September 21, 2012; released for publication December 19, 2012.

IEEE Log No. T-AES/49/4/944697.

Refereeing of this contribution was handled by L. Kaplan.

This work was supported by ONR Grants N00014-10-10412 and N00014-10-1-0029, and ARO Grant W911NF-06-1-0467.

Authors' addresses: S. Schoenecker, Naval Undersea Warfare Center, Sensors and Sonar Department, 1176 Howell Street, Newport, RI 02841-1708, E-mail: (steven.schoenecker@navy.mil); P. Willett and Y. Bar-Shalom, Department of Electrical and Computer Engineering, University of Connecticut, U-2157, Storrs, CT 06269.

I. INTRODUCTION

The maximum likelihood probabilistic multi-hypothesis (ML-PMHT) tracker is an algorithm that has performed well in a multistatic active sonar framework [13, 14, 21, 22]. The ML-PMHT tracker is closely related to a similar and better-known algorithm, the maximum likelihood probabilistic data association (ML-PDA) tracker. ML-PDA was originally developed in [9] and was subsequently expanded in [10], [4], [21], and [3]. As the "maximum-likelihood" in the name suggests, both ML-PMHT and ML-PDA are non-Bayesian estimators—they assume the target's motion is deterministic and can be parameterized by some unknown constant vector to be estimated. This, along with some assumptions about clutter and the environment lead to the development of a log-likelihood ratio (LLR). These assumptions were first developed for ML-PDA [2, 10] and are as follows:

1) A single target is present in each frame with known detection probability P_d. Detections are independent across frames.
2) The number of measurements per frame from the target is zero or one.
3) The kinematics of the target are deterministic. The motion is usually parameterized as a straight line, although any other parameterizations (e.g. an exoatmospheric ballistic trajectory or a maneuvering model) can be used.
4) The number of false detections (clutter) is Poisson distributed with known (fixed) spatial density and, consequently, their locations are uniformly distributed in the search volume.
5) Amplitudes of target and false detections are Rayleigh distributed. The parameter of each Rayleigh distribution is known (although the signal-to-noise ratio (SNR) may be tracked [4] in the case that it is not known; then P_d becomes time varying, which is easy to accommodate).
6) Target measurements are corrupted by additive zero-mean Gaussian noise with known variance.
7) Measurements at different times, conditioned on the parameterized state, are independent.

ML-PMHT is derived by combining these assumptions with the work of [1], [17], [18], and [19]. The one significant difference in assumptions between ML-PDA and ML-PMHT involves the target assignment model; ML-PMHT, instead of allowing only zero or one measurement per frame from the target, allows any number of target-originated measurements in a single frame. This target model may be unappealing to some, but work in [16] showed that in the case of a single target, the performances of ML-PDA and ML-PMHT are virtually identical, and when the actual target measurement generation

model matches the ML-PDA model (i.e., at most one measurement in a scan/frame is generated by the target), the ML-PMHT LLR converges to the ML-PDA LLR. (For convenience, these and all other acronyms used in this paper are spelled out in Table I.)

From the above assumptions, the ML-PMHT LLR (for a single target) is expressed as [21]

$$\Lambda(\mathbf{x},Z)=\sum_{i=1}^{N_w}\sum_{j=1}^{m_i}\ln\{\pi_0+\pi_1 V p[\mathbf{z}_j(i)\mid\mathbf{x}]\rho_j(i)\}. \quad (1)$$

In this equation, N_w is the number of scans/frames in the batch, m_i is the number of measurements in the ith scan, π_0 is the probability that any given measurement is from clutter, π_1 is the probability that any given measurement is from the target, and V is the search volume. Finally, $\rho_j(i)$ is the amplitude likelihood ratio for the ith measurement in the jth scan, $\mathbf{x}$ is the target parameter vector, and the probability density function (pdf) of the measurement, $p[\mathbf{z}_j(i)\mid\mathbf{x}]$, is a target-centered Gaussian. Values for π_0 and π_1 are obtained from the probability of target detection (P_d), the search volume, the spatial clutter density, and the number of targets [16].

The ML-PMHT is a batch algorithm, and we apply it to a scenario where all the targets are low-observable—the majority of target returns have an SNR of less than 9 dB. This makes the choice of a batch algorithm very appropriate, since recursive (nonbatch) trackers have difficulty operating in this regime [2].

There are two advantages to the ML-PMHT LLR formulation in comparison with the ML-PDA LLR formulation. First, the ML-PMHT LLR has a natural extension to multiple targets (ML-PDA does not), which leads to good performance when tracking multiple targets [16]. Secondly, in [10] and [4] an expression for the Cramér-Rao lower bound (CRLB) for ML-PDA was developed; due to the nature of the ML-PDA LLR, calculating this CRLB involved an integral of high dimensionality that needed to be done off-line with Monte-Carlo integration. In contrast, the ML-PMHT CRLB, developed in [16], can be evaluated in real-time with simple numerical integration. Results on the CRLB for the maneuvering-model ML-PMHT are discussed below.

This paper addresses three main topics. In Section II we compare the performance of ML-PMHT for a multistatic system implemented in Cartesian measurement space to that of ML-PMHT implemented in delay-bearing measurement space (this is an extension of work done in [15]). Many other (Kalman-based) trackers operate in Cartesian space in order to achieve a linear state transition. (Often in this space the measurement relationship is still nonlinear and must be linearized.) Such trackers can also operate in delay-bearing space via methods such as the extended Kalman filter (EKF) or the unscented Kalman filter (UKF)—an example of this is found in [7]—but this involves approximations to both the state transition and measurement equations. In contrast, it is possible to implement ML-PMHT (or ML-PDA) without any linearizing approximations in either measurement space. In a multistatic active framework, the measurement covariance is more accurately represented in delay-bearing space, so it should be advantageous to operate in this space. We verify this, and quantify the difference, by comparing the performance of the two implementations via Monte-Carlo simulation.

TABLE I
List of Acronyms

ML-PMHT	Maximum Likelihood Probabilistic Multi-Hypothesis Tracker
ML-PDA	Maximum Likelihood Probabilistic Data Association
SNR	Signal-to-Noise Ratio
CRLB	Cramér-Rao Lower Bound
FIM	Fisher Information Matrix
LLR	Log-Likelihood Ratio
MMFL	Maneuvering Model Fixed Length
MMVL	Maneuvering Model Variable Length
NEES	Normalized Estimation Error Squared

In Section III, we develop and test a maneuvering-model parameterization for ML-PMHT. As stated above, ML-PMHT must make an assumption about a target's motion; it parameterizes this motion over a batch of measurements with some vector (which includes the maneuver parameterization). In previous work, the target motion was always parameterized as a straight line, but obviously, not all targets move in straight lines. Because of this, the ML-PMHT tracker was implemented with a sliding batch/window, with the assumption that a maneuvering target trajectory could be reasonably approximated with a series of line segments. This allowed ML-PMHT to track moderately maneuvering targets, but not sharply maneuvering targets. To overcome this, we add a maneuver time t_m and a maneuver angle θ_m to the parameter set describing the target, which allows for a maneuver within the batch.

Finally, in Section IV, we extend the work in [16] that developed an expression for the Fisher information matrix (FIM) for the ML-PMHT tracker. This work covered straight-line target parameterizations; we expand this to cover the maneuvering-model parameterization developed in Section III. With the FIM, we can obtain the CRLB for ML-PMHT, and we show that, as in the case of straight-line parameterization, ML-PMHT with the maneuvering-model parameterization is an efficient estimator. As a result, the CRLB can be used in an

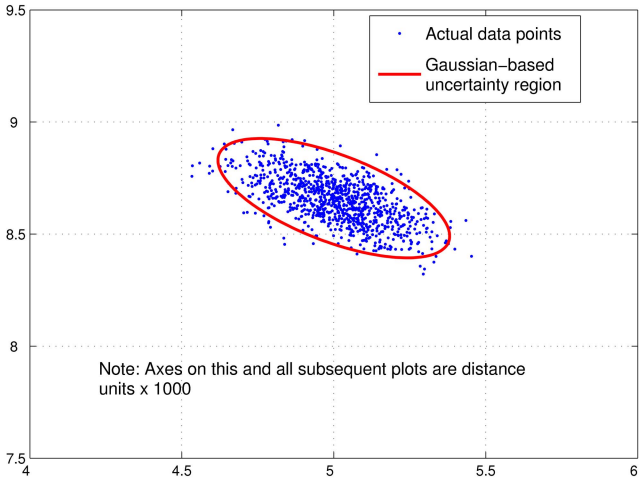


Fig. 1. Simulated points and equivalent Gaussian uncertainty region for $\sigma_t = 0.1$ s and $\sigma_\theta = 1°$.

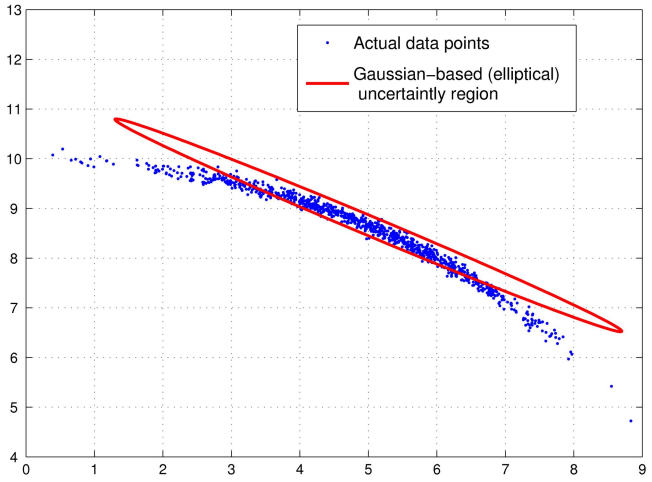


Fig. 2. Simulated points and equivalent Gaussian uncertainty region for $\sigma_t = 0.1$ s and $\sigma_\theta = 10°$.

ML-PMHT tracking implementation to provide an estimate for the covariance of any output tracks.

II. CARTESIAN MEASUREMENT SPACE VERSUS DELAY-BEARING MEASUREMENT SPACE

In previous work (see [8], [12], [13], and [14]), the ML-PMHT algorithm was implemented in Cartesian measurement space. Measurements of time-delay and azimuth were converted to (x, y) Cartesian coordinates using the bistatic equations of [6]. The Gaussian-distributed measurement errors in time-delay and azimuth space were converted to (approximate) Gaussian distributions in Cartesian space following the work of [5]. This approach worked reasonably well as long as the actual measurement errors were relatively small. However, as the errors (especially the azimuthal error) grow, the accuracy of the Gaussian assumption for the converted measurements starts to break down. This becomes a problem not only for ML-PMHT, but for all trackers operating in Cartesian measurement space. Fortunately, ML-PMHT can operate without any linearizing approximations in the delay-bearing measurement space. In contrast, many Kalman-filter based trackers stay in Cartesian measurement space in order to maintain at least a linear state transition.

To get a feel for the errors caused by converting measurements and their associated covariances to Cartesian space, consider the following simple example of a target at a range of 10,000 distance units (all distances in this work have arbitrary units) with a true bearing of 030° relative to north (assume that both the source and the receiver are at the origin in this case). Measurement errors are simulated for $\sigma_t = 0.1$ s and $\sigma_\theta = 1°$. These simulated points are plotted (after conversion into Cartesian coordinates) in Fig. 1. The ellipse in this figure is the Gaussian-based 95% probability region. In this situation, this approximation fits the data very well. Now consider the same geometry and time error, but with an azimuthal error

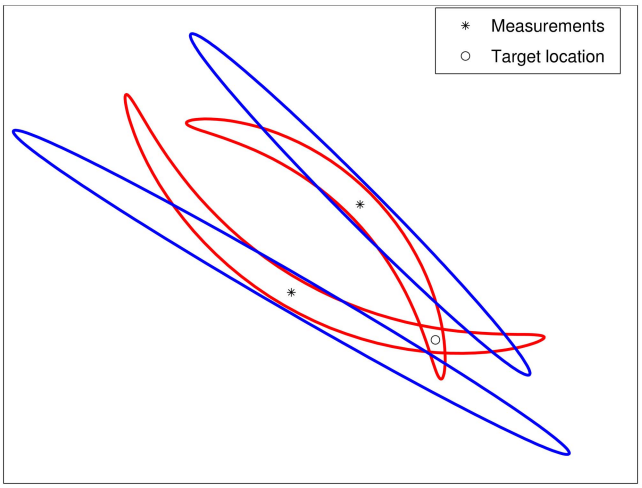


Fig. 3. Gaussian-approximated versus actual (crescent) uncertainty regions. In Gaussian approximation, target-originated measurements will be ignored.

of $\sigma_\theta = 10°$. Again, simulated measurements with the equivalent Gaussian-based ellipse are shown in Fig. 2. It is apparent in this case that the Gaussian is no longer a good approximation to the actual distribution of the measurement data: the Gaussian-based ellipse cannot cover the "crescent-shaped" uncertainty region.[1]

Another way to think about this is in terms of data fusion. The ML-PMHT likelihood ratio (1) implicitly fuses measurements that are close together. If the uncertainty regions of a given set of measurements overlap, then these measurements are effectively fused together and used in the determination of the target solution. Figure 3 illustrates this. Two target-originated measurements (from two different source-receiver pairs) are plotted in relation to the

[1]This is the so-called "contact lens problem" discussed in [20] where recursive estimators were considered. Here we use a batch estimator for low-SNR targets.

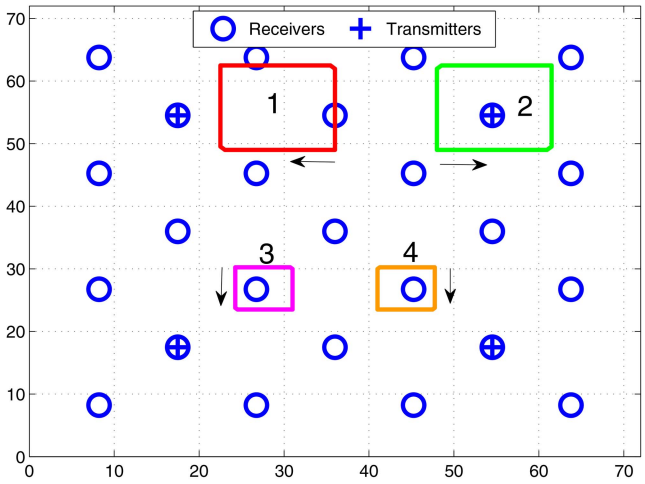


Fig. 4. Multistatic scenario used for Monte-Carlo runs.

target. Their approximate Gaussian-based ellipses do not overlap each other (or the target location), so it is unlikely that these measurements will be associated together—the chances are good that both will be ignored in the solution determination. In contrast, the correct "crescent" uncertainty regions do overlap, so with this representation, cross-sensor data association and fusion are possible and the measurements will (correctly) be used in the determination of the target solution $\mathbf{x}$.

Consider now a target moving in a straight line and a tracker operating in delay-bearing measurement space. Here, the measurement equation is highly nonlinear. The target motion (assumed to be deterministic over the relatively short time interval covered by a batch of measurements) is parameterized by the vector

$$\mathbf{x} = (x_0 \quad \dot{x} \quad y_0 \quad \dot{y})^T. \tag{2}$$

The Cartesian measurement matrix $\mathbf{H}$ is given by

$$\mathbf{H} = \begin{bmatrix} 1 & t & 0 & 0 \\ 0 & 0 & 1 & t \end{bmatrix} \tag{3}$$

so the predicted Cartesian measurements at time t for this state vector are given by

$$\begin{pmatrix} x \\ y \end{pmatrix} = \mathbf{H}\mathbf{x}. \tag{4}$$

We can now use these predicted Cartesian measurements to get the predicted time-delay

$$\tau = \frac{r_{TS} + r_{TR}}{c} \tag{5}$$

where r_{TS} is the distance from the predicted measurement to the source, r_{TR} is the distance from the predicted measurement to the receiver, and c is the speed of sound. The azimuth is given by

$$\theta = \tan^{-1}\left(\frac{y - y_R}{x - x_R}\right). \tag{6}$$

Here, x_R and y_R are the (known) Cartesian positions of the receiver. This state-to-measurement conversion is highly nonlinear, making it difficult to use Kalman-based estimation techniques. In contrast, the ML-PMHT algorithm can operate without any linearization in Cartesian space or in delay-bearing measurement space. For a given set of measurements, it finds the parameterization of the target motion producing predicted measurements that maximize the ML-PMHT LLR (1). In delay-bearing space, these predicted measurements are produced by taking an initial target state (2) and propagating the target forward to the time of the measurement according to (4). Then, given the target's assumed (x, y) position, the predicted delay time and azimuth measurements τ and θ are calculated via (5) and (6).

A. Data Set Description

A data scenario was created to compare the performance of the Cartesian versus delay-bearing implementation of ML-PMHT. This data scenario was set up to match the first scenario in the Metron 2009 dataset [11], which was a simulated benchmark multistatic data set put out for use by the Multistatic Tracking Working Group (MSTWG). The scenario (shown in Fig. 4) features four targets, with 25 receivers and four transmitters, set out over an approximate 60,000-by-60,000 distance unit square grid. Each target does a single revolution in a rectangular pattern. The simulation features two types of pings—a CW ping with a delay-time error of $\sigma_t = 0.1$ s, and an FM ping with $\sigma_t = 0.01$ s. On average, there are approximately 35 clutter returns per scan, and the majority of the target returns have an SNR between 4.95 dB (the detector threshold) and 9 dB. The average target probability of detection in a given scan is $P_d = 0.11$. These produce typical values of $\pi_0 \approx 0.99$ and $\pi_1 \approx 0.01$ in the LLR (1). Under such conditions the EKF or other recursive algorithms may encounter difficulties. Finally, what makes this dataset interesting from the point of view of delay-bearing processing is the azimuthal uncertainty of the receivers—this uncertainty is set at $\sigma_\theta = 8°$, which will produce an uncertainty region similar to that shown in Fig. 2.

B. Monte-Carlo Results

For the scenario shown in Fig. 4, 200 runs were performed with Cartesian processing and delay-bearing processing. For each run, the following metrics were evaluated: target in-track percentage, root mean-square error (RMSE), track fragmentation, number of duplicate tracks, number of false tracks, and mean false track length. Briefly, the metrics are defined as follows. In-track percentage was calculated by taking (for each target) the ratio of target truth points that had a track associated with them to the

TABLE II
Metrics from Straight-Line and Maneuver-Model Monte-Carlo Testing from Metron Scenario 1

	Straight Line				Maneuver Model					
	Cartesian		Delay-Bearing		Cartesian		DB MMFL		DB MMVL	
	Mean	Conf. Int.	Mean	Conf. Int.	Mean	Conf. Int.	Mean	Conf. Int.	Mean	Conf. Int.
Target In-Track Percentage										
Tgt 1	73.3	[70.6, 76.0]	72.5	[69.2, 75.9]	76.0	[73.0, 79.0]	99.1	[98.3, 100]	100.0	[100, 100]
Tgt 2	58.1	[54.6, 61.8]	67.1	[64.5, 69.8]	68.1	[64.5, 71.7]	95.1	[93.0, 97.1]	99.96	[99.91, 100]
Tgt 3	78.6	[75.6, 81.7]	99.7	[99.6, 99.9]	82.7	[80.3, 85.2]	99.9	[99.8, 100]	99.86	[99.75, 100]
Tgt 4	81.3	[78.6, 84.0]	99.5	[99.2, 99.9]	82.7	[79.7, 85.6]	99.9	[99.8, 100]	99.95	[99.88, 100]
RMSE										
Tgt 1	965	[919, 1010]	771	[711, 833]	872	[826, 917]	388	[347, 429]	193	[187, 199]
Tgt 2	1216	[1154, 1278]	1127	[1090, 1165]	1000	[947, 1053]	617	[562, 673]	226	[218, 233]
Tgt 3	951	[909, 992]	671	[645, 697]	854	[810, 899]	317	[301, 334]	207	[200, 213]
Tgt 4	914	[871, 957]	786	[755, 818]	862	[816, 909]	404	[380, 429]	219	[211, 226]
Track Fragmentation										
Tgt 1	1.70	[1.58, 1.83]	0.63	[0.55, 0.70]	1.44	[1.30, 1.58]	0.03	[0.01, 0.05]	0	[0, 0]
Tgt 2	2.01	[1.88, 2.13]	0.48	[0.39, 0.57]	1.66	[1.53, 1.80]	0.12	[0.08, 0.17]	0.01	[0.00, 0.02]
Tgt 3	1.68	[1.55, 1.80]	0.07	[0.03, 0.12]	1.37	[1.24, 1.50]	0	[0, 0]	0	[0, 0]
Tgt 4	1.58	[1.45, 1.72]	0.05	[0.02, 0.08]	1.42	[1.29, 1.54]	0.03	[0.00, 0.05]	0	[0, 0]
Number Duplicate Tracks										
Tgt 1	0.68	[0.58, 0.79]	0.87	[0.75, 0.99]	0.14	[0.09, 0.20]	0.27	[0.20, 0.34]	0	[0, 0]
Tgt 2	0.62	[0.50, 0.75]	0.98	[0.88, 1.09]	0.15	[0.09, 0.21]	0.45	[0.38, 0.52]	0	[0, 0]
Tgt 3	0.67	[0.56, 0.74]	1.09	[0.97, 1.20]	0.09	[0.05, 0.13]	0.09	[0.05, 0.13]	0.01	[0, 0.01]
Tgt 4	0.65	[0.54, 0.76]	1.51	[1.40, 1.62]	0.20	[0.13, 0.25]	0.25	[0.19, 0.31]	0.01	[0, 0.01]
Number False Tracks										
	3.84	[3.57, 4.11]	3.50	[3.33, 3.66]	1.92	[1.72, 2.11]	0.28	[0.19, 0.31]	0	[0, 0]
Mean False Track Length										
	4.53	[4.33, 4.73]	7.02	[6.81, 7.23]	4.67	[4.24, 5.11]	2.03	[1.41, 2.66]	0	[0, 0]

total number of target truth points. Target duplicate tracks were calculated by counting (again for each target) the number of tracks that were assigned to a target that had at least some overlap with another track assigned to the same target (i.e., at least one target truth point had more than one track assigned to it). Track fragmentation was determined by counting the number of breaks in track for a target. Finally, the overall number of false tracks was the number of tracks not associated with a target, and mean false track length was the average number of updates for which a false track was active. (These metrics are described in more detail in [13].) All results are presented in Table II.

Overall, the delay-bearing implementation clearly outperformed the Cartesian implementation. In particular, two metrics stand out. First, the delay-bearing implementation was nearly 20 points better in some cases in terms of in-track percentage. The delay-bearing implementation also performed much better—in all cases by a factor of almost two—in terms of track fragmentation. This is a measure of the algorithm's ability to maintain a continuous (unbroken) track. The number of duplicate tracks was slightly higher for the delay-bearing implementation, but this turned out to be somewhat of a unique case that was due to how the duplicate track metric is calculated.[2] Overall, the more accurate measurement covariance for the delay-bearing implementation allows ML-PMHT to better initiate and maintain track on a target.

An example result for the scenario using the ML-PMHT Cartesian implementation is shown in

[2]As is stated above, a duplicate track is counted if there is any overlap (even just one point) between two tracks on the same target. In the Cartesian case, there were large gaps between track segments. In the delay-bearing case, the track segments were on average much closer to each other (as is seen by the higher P_d values). In many cases a track would go all the way to the end of a leg. It would be unable to "turn the corner" with the target, and a new track would be started at the corner on the new leg. There would be one point of overlap for the two tracks at the corner, generating a duplicate track. In contrast, for the Cartesian implementation, usually the first track would not make it all the way to the corner, or a second track would not start up right away on the second leg. Thus, the poorer P_d results for the Cartesian implementation are "suppressing" the duplicate track results.

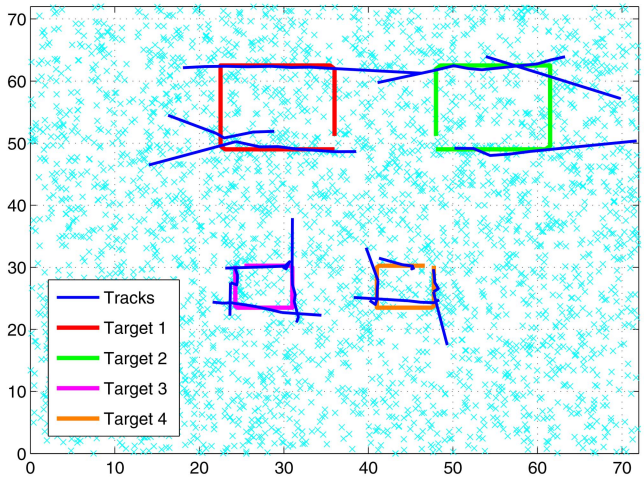


Fig. 5. Example plot with Cartesian straight-line parameterization. Measurements from first batch shown.

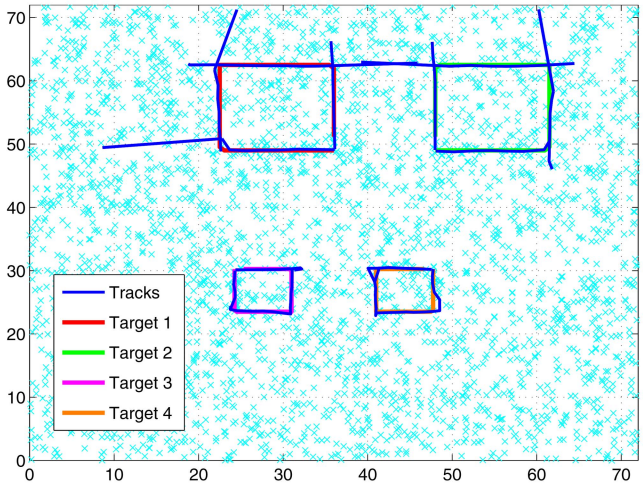


Fig. 6. Example plot with delay-bearing straight-line parameterization. Measurements from first batch shown.

Fig. 5. A result from the same scenario using the ML-PMHT delay-bearing implementation is shown in Fig. 6. There is a noticeable difference between these results—the delay-bearing processing shows cleaner, more accurate tracks, with higher in-track percentage, which is consistent with the results seen in Table II.

III. MANEUVERING-MODEL PARAMETERIZATION

We now introduce a maneuvering-model parameterization that allows ML-PMHT to track sharply maneuvering targets. Up to this point, the ML-PMHT tracker was implemented in a sliding-batch manner, and within a single batch, the target motion was assumed to be a straight line that could be completely described by the parameter vector in (2). As long as the target was not maneuvering severely, this sliding-batch straight-line parameterization was sufficient to track through maneuvers, as is illustrated with a simple example

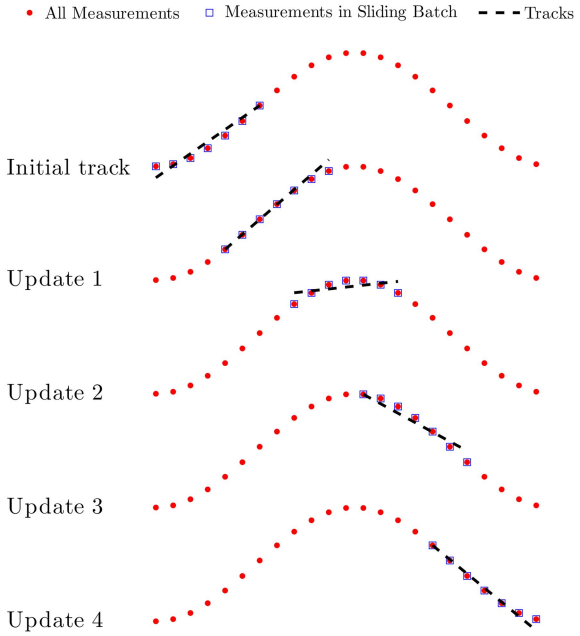


Fig. 7. Sliding window implementation for ML-PMHT in case of moderately-maneuvering target. For measurements in each sliding batch (denoted by blue squares), ML-PMHT straight-line solutions are shown.

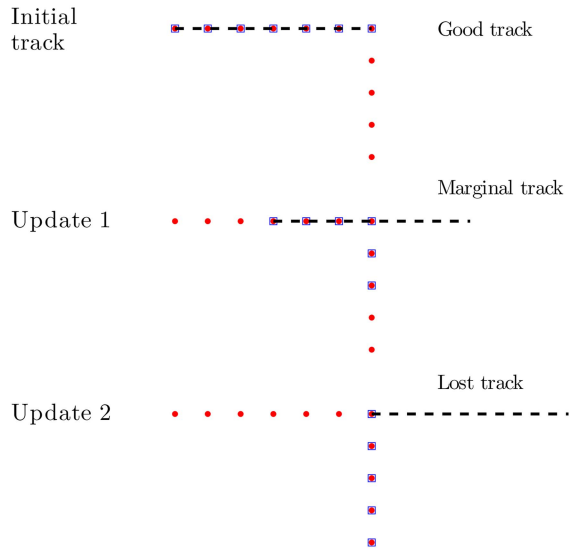


Fig. 8. Sliding window straight-line parameterization for ML-PMHT cannot follow sharply maneuvering target. Track is lost on target.

in Fig. 7.[3] However, when the target maneuvers more severely, the straight-line parameterization causes ML-PMHT to track off the target, which is shown with another simple example in Fig. 8. This

[3]This figure is merely illustrative for how the sliding batch works. In the actual tracker implementation, each batch is 1800 s (10 time updates) long. At the end of each tracker update, the batch is slid forward by 360 s. For a given existing track, as its batch is slid forward, the solution from the previous update is projected forward and used as an initialization point for the optimization to find the solution for the current batch, which obviates the need to do track-to-track association between subsequent batches.

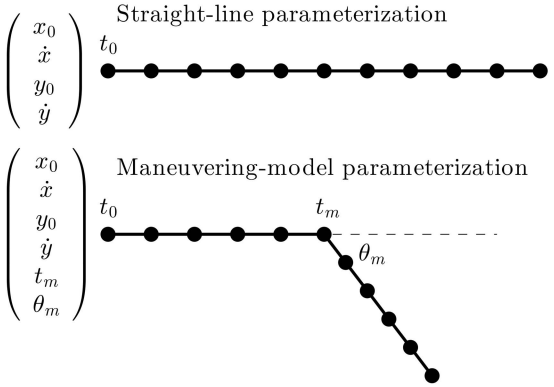


Fig. 9. Straight-line parameterization versus maneuvering-model parameterization.

can also be seen in actual tracking examples in the previous section, in Figs. 5 and 6. In these examples, for some of the targets ML-PMHT has a good track initially; then the target maneuvers suddenly by 90°. The tracker is not able to follow the target; the best straight-line solution it can come up with misses the maneuver, and the track ends up being dead-reckoned off on the original path until it is dropped. (New tracks are initialized after the maneuvers, but this is still at best a break in the track.) In order to enable ML-PMHT to track targets performing these more severe maneuvers, we now add two components to the parameter vector, a maneuver time t_m and a maneuver angle (course change) θ_m. Now, the target motion in a batch can be represented as two line segments—this new parameterization is shown in comparison to the previous straight-line parameterization in Fig. 9.

With this maneuvering model, obtaining the predicted location of a measurement at any time during the batch is only slightly more difficult. If the time of the predicted measurement is prior to the (assumed) maneuver time, then nothing changes—the position of this predicted measurement is that given by (4) (or (5) and (6) if we are operating in delay-bearing mode). If the predicted measurement is after the maneuver time, its location is calculated by first defining a new 4-component state $\mathbf{x}_m$ directly after the maneuver has occurred

$$\mathbf{x}_m = \mathbf{H}_m \mathbf{x} \tag{7}$$

where $\mathbf{H}_m$ is given by

$$\mathbf{H}_m = \begin{bmatrix} 1 & t_m & 0 & 0 \\ 0 & R_{00} & 0 & R_{01} \\ 0 & 0 & 1 & t_m \\ 0 & R_{10} & 0 & R_{11} \end{bmatrix}. \tag{8}$$

Here, we have introduced the notation of R_{ij} being the (i,j)th component of a standard 2-by-2 rotation matrix with θ_m as the rotation angle. Now, to get the projected Cartesian postmeasurement point, we simply project forward from the maneuver point with the (rotated) velocity vector

$$\begin{pmatrix} x_{\text{pm}} \\ y_{\text{pm}} \end{pmatrix} = \begin{pmatrix} 1 & t-t_m & 0 & 0 \\ 0 & 0 & 1 & t-t_m \end{pmatrix} \mathbf{x}_m. \tag{9}$$

We combine (7)–(9) to obtain a final expression for the projection point at any time after the maneuver:

$$\begin{pmatrix} x_{\text{pm}} \\ y_{\text{pm}} \end{pmatrix} = \bar{\mathbf{H}} \mathbf{x}_0 \tag{10}$$

where

$$\begin{aligned} \bar{\mathbf{H}} &= \mathbf{H}, \qquad t \le t_m \\ \bar{\mathbf{H}} &= \begin{bmatrix} 1 & S_{00} & 0 & T_{01} \\ 0 & T_{10} & 1 & S_{11} \end{bmatrix}, \qquad t > t_m. \end{aligned} \tag{11}$$

Here, we have introduced the notation of $S_{ij} = t_m + (t-t_m)R_{ij}$ and $T_{ij} = (t-t_m)R_{ij}$. Again, if we are operating in delay-bearing space, we simply take the results of (11) and apply them to (5) and (6).

A. Implementing the Maneuvering Model

The overall logic for implementing the maneuvering-model parameterization is shown in Fig. 10. The idea is to cycle through each existing track and check the track's maneuver status. If a maneuver is in progress, we optimize over the 4-dimensional parameter vector (2), holding the maneuver time and the maneuver angle constant. If no maneuver is in progress, we check for a new maneuver by optimizing over the parameters in (2) as well as the maneuver time and maneuver angle. If we find a maneuver time and angle that produce a better solution (in terms of a larger LLR value) than the straight-line parameterization (subject to some checks described below), we declare a new maneuver.

When checking for a new maneuver (by optimizing over the six parameters), we implement some constraints to limit the declaration of "spurious" maneuvers. First, we constrain the maneuver time t_m to be close to the middle of the batch. If the entire batch length is T seconds long, the only allowable maneuver times fall between $5T/12$ and $2T/3$ seconds. (A series of different window sizes was tried during development; this time window was found empirically to work the best.) Additionally, only maneuvers greater than a certain angle are allowed. Consider Fig. 11—this shows an example of a "maneuver" that is just caused by measurement noise; we want to avoid declaring this a maneuver. Let d be the distance a target moves in a batch of length T, and Δh be the "cross-range" component of a "typical" measurement covariance. For this typical measurement covariance, we select a geometry where the range from target to receiver (r_{TR}) is small, giving a high probability of target detection. If σ_θ is the receiver's angular measurement uncertainty, then to a good approximation

$$\Delta h \approx r_{TR}\sigma_\theta. \tag{12}$$

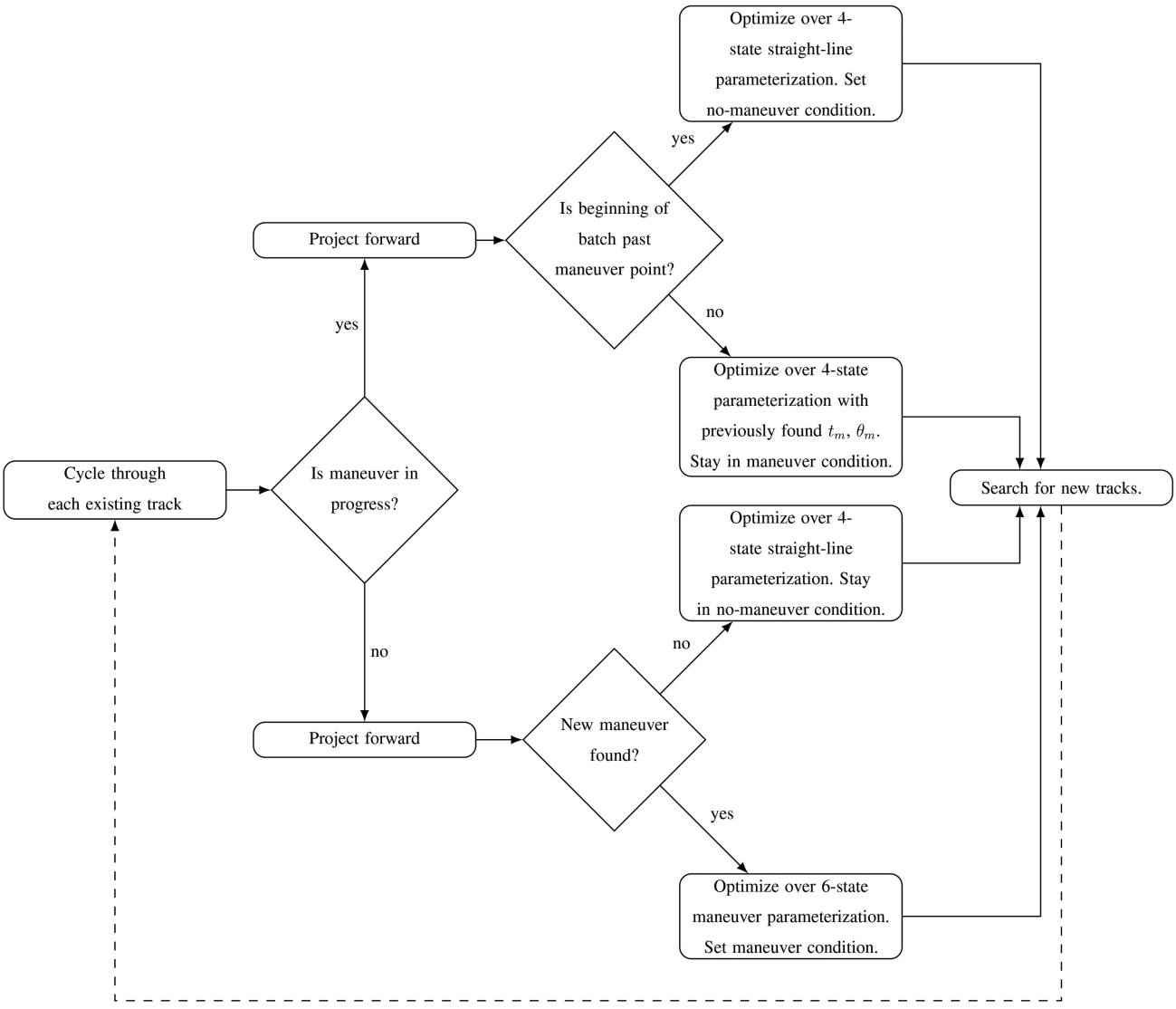


Fig. 10. Maneuvering-model logic diagram.

From the geometry of Fig. 11, it easy to see that

$$\theta_{\min} = 2\tan^{-1}\left(\frac{2\Delta h}{d}\right). \tag{13}$$

For example, for a value of $d = 2000$ units (a typical value for a slow speed target during a batch of data) and $r_{TS} = 2000$ units with $\sigma_\theta = 8°$, we end up with a value of $\theta_{\min} = 30°$. (Previous work [16] has shown that the original straight-line parameterization can handle gradually maneuvering targets with a maneuver angle less than this value.)

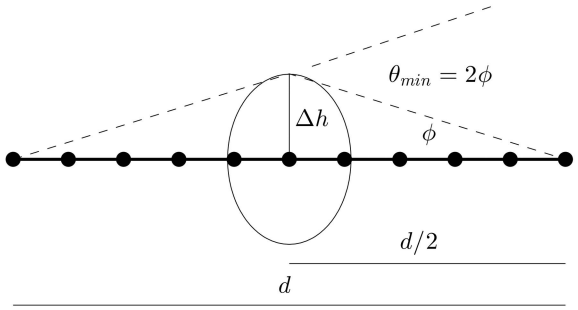


Fig. 11. Geometry of minimum maneuver allowed.

The maneuvering-model parameterization was implemented in two different ways. First, it was done with a fixed-length batch size (in this work the length was ten sampling times), regardless of whether the target was in straight-line mode or in maneuvering mode. The maneuvering-model parameterization was also implemented with a variable-length batch, following the work of [4]. In this implementation, when searching for an initial maneuver, the back end of the batch was fixed, and the front end of the batch was allowed to expand out to a maximum of twice the initial batch size. For each different batch size, a search was performed for the maneuver parameters t_m and θ_m—this is shown in Fig. 12. The maneuver parameters that produced the largest ML-PMHT LLR (along with their corresponding batch size) were selected as a potential maneuver. If the LLR produced

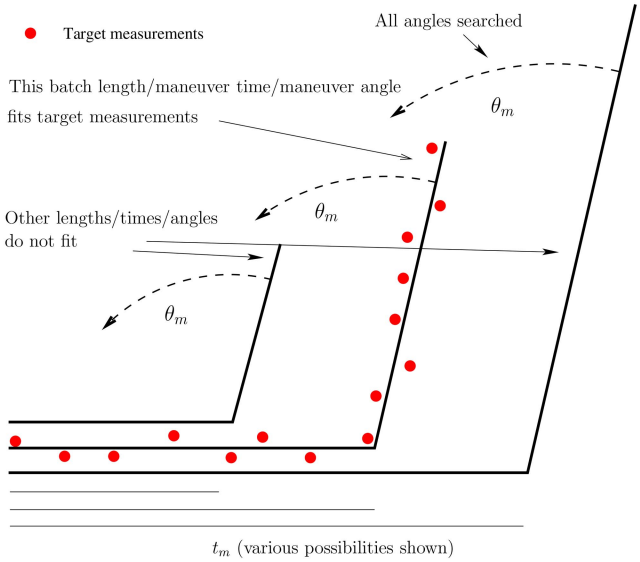


Fig. 12. Illustration of maneuvering-model parameterization with variable-batch length.

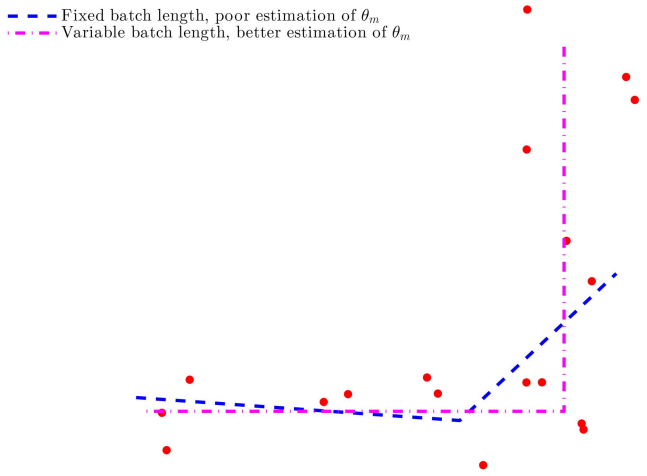


Fig. 13. Comparison of fixed-length batch (MMFL) versus variable-length batch for maneuvering-model parameter determination (MMVL). MMFL implementation calls maneuver early and inaccurately, whereas MMVL implementation accurately finds maneuver.

by this maneuver was larger than the LLR from the simple straight-line parameterization, a maneuver was declared. At this point the maneuver parameters and the front end of the batch were fixed, while the back of the batch was advanced at each time update. When the back of the batch passed the maneuver point, the maneuver was declared over, and the process was started again.

The idea behind this variable-batch length implementation was to avoid finding maneuvers "too early." With the fixed-length batch size implementation, if a maneuver was declared with the minimum allowed amount of measurements after the maneuver time t_m, this could cause the maneuver angle to be inaccurately determined. Since the maneuver angle is fixed once a maneuver is declared, this would often lead to ML-PMHT losing track on subsequent updates—an example of this is shown in Fig. 13.

For Cartesian processing, only the fixed-batch length method (from now on denoted MMFL) was used. For delay-bearing processing, both the fixed-batch length and variable-batch length (from now on denoted MMVL) were used.

Overall, there is a tradeoff when using the maneuver-model parameterization (both MMFL and MMVL) between finding sharp maneuvers and tracking straight-line targets. The maneuver-model parameterization can be thought of as opening the bandwidth of the tracker, which makes it possible to find sharp maneuvers within a batch, but it also allows for the possibility of overparameterizing the target motion by fitting a maneuver to the measurement noise. This was seen in the preliminary work done with the maneuver-model parameterization where there was no constraint on the maneuver angles that could be found. This resulted in the tracker almost constantly fitting small maneuvers to straight-line motion—it was overparameterizing the motion. This was fixed by recognizing that the straight-line parameterization already had acceptable performance for moderately-maneuvering targets and thus it was possible to limit "found" maneuvers to those with significant maneuver angles.

We also note (as is shown in Fig. 10) that the maneuver model logic is only applied to existing tracks. Continuing with the "bandwidth" notion, a wider passband can permit more clutter to affect the solution, which in turn can adversely affect ML-PMHT's ability to find dim targets. As a result, the search for new tracks (which is done at every update) is only done with the straight-line parameterization.

Finally, for simplicity and consistency, we intentionally make it difficult to declare a maneuver, and once a maneuver is declared, the maneuver parameters (time and angle) are held constant until the target has passed through the maneuver point. By doing this we are again limiting the degree to which the bandwidth of the tracker is opened—this will limit the fitting of maneuvers to noise. (For the same reason we chose to only look for one maneuver at a time.) Additionally (and not insignificantly) there is also the consideration of computation time. The process of searching for the maneuver parameters increases the required computation time. If the straight-line parameterization processing time is taken as the baseline, the MMFL approach requires an increase of approximately twice the baseline computation time, and the MMVL approach requires an increase between four and five times the baseline. This increase in processing time is almost entirely due to searching

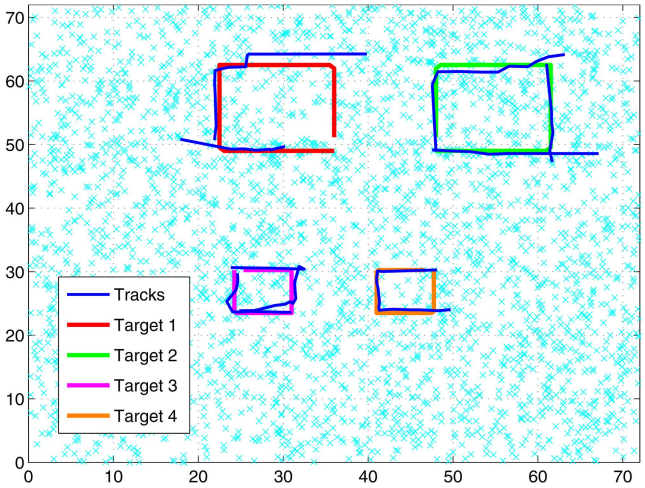


Fig. 14. Example of Cartesian maneuvering-model processing. Measurements from first batch shown.

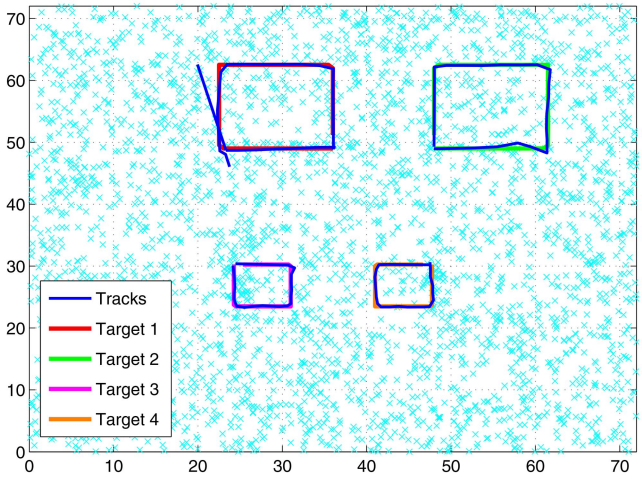


Fig. 15. Example of delay-bearing MMFL processing. Measurements from first batch shown.

for maneuver parameters when a maneuver has not been found. Not fixing the maneuver parameters (i.e., searching for the maneuver parameters at every update) would greatly increase the required processing time.

The approach we have taken is a suitable compromise between opening up the bandwidth of the tracker to find maneuvers while maintaining its original straight-line performance.

B. Maneuvering Model Results

The introduction of the maneuvering-model processing greatly improved the performance of the ML-PMHT tracker. For the Cartesian implementation (using the measurements converted into Cartesian coordinates), when going from the straight-line parameterization to the maneuvering-model parameterization, the average percentage time-in-track went up slightly, between 3% and 10%, for all four targets. For the delay-bearing implementation (i.e., using the measurements in the sensor coordinates), there were more impressive performance gains; for targets 3 and 4 with MMFL, there was an increase in percentage time-in-track of nearly 30 points. For this maneuvering-model implementation, all four targets were being tracked almost 100% of the time. The MMVL parameterization did even better, tracking all targets in the scenario 100% of the time, with higher accuracy than the MMFL parameterization; the RMSE values for MMVL are approximately half that of MMFL.

The maneuvering-model processing also produced a slight decrease in the number of false tracks, which is somewhat counterintuitive, since the bandwidth of the tracker is being widened. This decrease in false tracks is due to the fact that with straight-line processing, sometimes a track initialized on a true target near a maneuver ended up wandering off immediately and being counted as a false track. This "wandering" was less likely with the maneuvering-model processing.

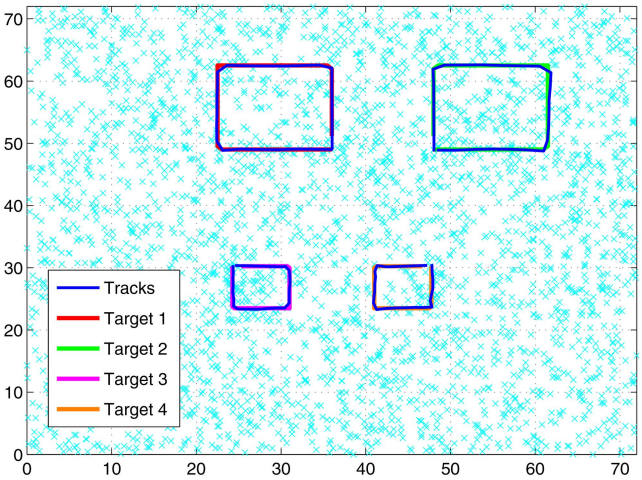


Fig. 16. Example of delay-bearing MMVL processing. Measurements from first batch shown.

Three example plots are shown (see Figs. 14–16) to demonstrate the progression from the Cartesian implementation of the maneuver model parameterization to the delay-bearing MMFL to the delay-bearing MMVL. There is clearly an increase in track quality from plot to plot. It is also instructive to consider the improvement in performance between the original, straight-line-only Cartesian implementation to the best delay-bearing maneuvering-model version, MMVL. The time-in-track percentages for targets 1, 2, 3, and 4 were 73%, 58%, 79%, and 81% for the former, while the value was practically 100% for all four targets for the latter. RMSE decreased from approximately 1000 distance units to 200 distance units, and all other metrics showed great improvement as well. Overall, the maneuvering-model

parameterization in conjunction with delay-bearing processing adds significant performance improvement to the ML-PMHT tracker.

IV. MANEUVER MODEL CRAMÉR-RAO LOWER BOUND

In this section, we develop the FIM for the maneuvering-model parameterization and then check the ML-PMHT tracker for efficiency. In [16] the FIM was developed for the ML-PMHT tracker with a straight-line parameterization; we expand on this work.

We note that the bound we are computing here is actually the model CRLB—that is, the expression calculated is the lower bound given that the ML-PMHT target measurement assignment model is true.

The expression for the FIM for a batch of data (for either straight-line or maneuvering-model parameterizations) is given by

$$\mathbf{J} = \sum_{i=1}^{N_w} \mathbf{J}_i \tag{14}$$

and the FIM for a single scan is given by

$$\mathbf{J}_i = \mathbf{D}_\phi^T \sum_{j=1}^{m_i} \mathbf{G}_j^T \int_{\tau_a}^{\infty} \int_V \frac{\frac{[\pi_1 p_1^\tau(a_j)]^2}{|2\pi\mathbf{R}_j|} e^{-\xi_j^T \xi_j} \xi_j \xi_j^T}{\frac{\pi_0 p_0^\tau(a_j)}{V} + \frac{\pi_1 p_1^\tau(a_j)}{\sqrt{|2\pi\mathbf{R}_j|}} e^{-(1/2)\xi_j^T \xi_j}} d\xi_j da_j \times \frac{\mathbf{G}_j}{|\mathbf{G}_j|} \mathbf{D}_\phi. \tag{15}$$

(The full derivation of this expression is in [16].) Here, $\mathbf{R}_j$ is the measurement covariance for the jth measurement in the ith scan (the scan number is implied), and $\mathbf{G}_j$ is the Cholesky decomposition of $\mathbf{R}_j^{-1}$. Finally, τ_a is the amplitude threshold, and ξ_j is either a 2-dimensional or 3-dimensional variable of integration, depending on whether Doppler information is being processed. If the measurement covariance $\mathbf{R}$ is constant within a scan (which it is above in the delay-bearing implementation), (15) can be simplified (note the j-subscripts are dropped), and the FIM for a scan reduces to a single 2-dimensional or 3-dimensional integration:

$$\mathbf{J}_i = m_i \mathbf{D}_\phi^T \mathbf{G}^T \mathbf{K}_i \mathbf{G} \mathbf{D}_\phi \tag{16}$$

where

$$\mathbf{K}_i = \frac{1}{|\mathbf{G}|} \int_{\tau_a}^{\infty} \int_V \frac{\frac{[\pi_1 p_1^\tau(a)]^2}{|2\pi\mathbf{R}|} e^{-\xi^T \xi} \xi \xi^T}{\frac{\pi_0 p_0^\tau(a)}{V} + \frac{\pi_1 p_1^\tau(a)}{\sqrt{|2\pi\mathbf{R}|}} e^{-(1/2)\xi^T \xi}} d\xi da \tag{17}$$

The Jacobian is defined as

$$\mathbf{D}_\phi = \begin{pmatrix} \frac{\partial\phi_1}{\partial x_0} & \frac{\partial\phi_1}{\partial \dot{x}} & \frac{\partial\phi_1}{\partial y_0} & \frac{\partial\phi_1}{\partial \dot{y}} & \frac{\partial\phi_1}{\partial t_m} & \frac{\partial\phi_1}{\partial \theta_m} \\ \frac{\partial\phi_2}{\partial x_0} & \frac{\partial\phi_2}{\partial \dot{x}} & \frac{\partial\phi_2}{\partial y_0} & \frac{\partial\phi_2}{\partial \dot{y}} & \frac{\partial\phi_2}{\partial t_m} & \frac{\partial\phi_2}{\partial \theta_m} \\ \frac{\partial\phi_3}{\partial x_0} & \frac{\partial\phi_3}{\partial \dot{x}} & \frac{\partial\phi_3}{\partial y_0} & \frac{\partial\phi_3}{\partial \dot{y}} & \frac{\partial\phi_3}{\partial t_m} & \frac{\partial\phi_3}{\partial \theta_m} \end{pmatrix}. \tag{18}$$

The entries of this Jacobian matrix (which differ between the straight-line parameterization and the maneuvering-model parameterization) are now developed. In Cartesian measurement space, the predicted measurement vector ϕ is given by

$$\phi^C = (x \quad y \quad \tilde{r})^T. \tag{19}$$

First, components x, y, $\dot{x}$, and $\dot{y}$ (premaneuver or postmaneuver) are calculated according to (7) and (10). The bistatic range-rate equation is given by [6]. With these expressions, the matrix entries of (18) can be calculated. In Cartesian processing these expressions are relatively straightforward, and are presented in the Appendix.

In delay-bearing measurement space, the predicted measurement vector is given by

$$\phi^{DB} = (t \quad \theta \quad \tilde{r})^T. \tag{20}$$

The expressions for the matrix entries of (18) in this case are slightly more complicated; however, they can be written as functions of the Cartesian Jacobian entries. They are also presented in the Appendix.

With these expressions we can check to see if ML-PMHT with the maneuvering-model parameterization is an efficient estimator. In [16], it was shown that the estimation error for ML-PMHT with straight-line parameterization (for a straight-line target) meets the CRLB. Now, the maneuvering-model parameterization for ML-PMHT is tested to see if it meets the maneuvering-model CRLB. To this end, 500 runs were performed on a single target making an 85° turn (close to one of the turns shown above in Fig. 4). For each run, when a maneuver was declared, the corresponding 6-parameter FIM was calculated, and the inverse of this was taken as the CRLB. With this the normalized estimation error squared (NEES) value was calculated

$$\text{NEES} = (\hat{\mathbf{x}} - \mathbf{x}_{\text{truth}})^T \mathbf{C}^{-1} (\hat{\mathbf{x}} - \mathbf{x}_{\text{truth}}) \tag{21}$$

where $\mathbf{C}$ is the CRLB matrix, and $\mathbf{C}^{-1} = \mathbf{J}$. Each NEES value is a realization of a chi-squared random variable with 6 degrees of freedom, so the average of all 500 NEES values should be 6. The 95-percent probability interval is [5.49, 6.51]; the actual average was 6.35, which falls within the interval. A subset of 20 runs is plotted in Fig. 17—here we are showing solutions overlaid on the position components of the CRLB for the initial state, the maneuver state, and

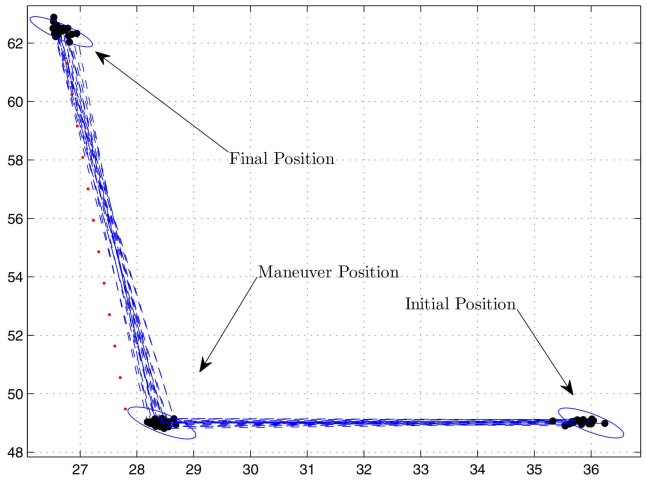


Fig. 17. CRLB contours for initial position, maneuver position, and final position.

the final state. These results show that ML-PMHT with the maneuvering-model parameterization is an efficient estimator.

V. CONCLUSION

The ML-PMHT tracker was first applied to a benchmark multistatic sonar simulation implemented in both Cartesian measurement space and delay-bearing measurement space. As expected, the tracker performed far better in delay-bearing space, since in this space the measurement Gaussian distribution correctly represents the uncertainty. This shows that ML-PMHT (and any other trackers that can operate in delay-bearing space) have a significant advantage when operating on multistatic sonar data over trackers that work in Cartesian measurement space. In the low-observable target-amplitude regime of the scenario, ML-PMHT also benefited from being a batch tracker.

A maneuvering-model parameterization was developed for ML-PMHT to increase the tracker's ability to follow rapidly-maneuvering targets. This parameterization, especially when implemented in a delay-bearing variable-batch length mode, significantly increased the tracker's performance in almost all metric areas.

Finally, an expression previously developed for the ML-PMHT FIM/CRLB was extended for the maneuvering-model parameterization. Using this, the NEES calculated over Monte-Carlo trials showed that ML-PMHT with the maneuvering-model parameterization is an efficient estimator. This is a valuable result in and of itself, and it also allows the use of the CRLB as a consistent estimate for the state covariance in an ML-PMHT tracking framework.

Overall, each of the three portions of this work extend the capability of the ML-PMHT tracker and make the algorithm a powerful low-observable multistatic sonar tracker.

APPENDIX MANEUVER-MODEL JACOBIAN ENTRIES

Here, we present the components for the entries of the Jacobian given in (18). For Cartesian measurement space, the first entry in measurement space (Cartesian x-position) is

$$\phi_1 = x_0 + T_{00}\dot{x} + S_{01}\dot{y} \tag{22}$$

where $T_{ij} = t_m + (t - t_m)R_{ij}$ and $S_{ij} = (t - t_m)R_{ij}$. (Here, R_{ij} is the (i,j)th entry of the rotation matrix with the maneuver angle as the rotation angle.) The derivatives with respect to the positions and velocities are

$$\frac{\partial \phi_1}{\partial x_0} = 1, \quad \frac{\partial \phi_1}{\partial \dot{x}} = T_{00}, \quad \frac{\partial \phi_1}{\partial y_0} = 0, \quad \frac{\partial \phi_1}{\partial \dot{y}} = S_{01}. \tag{23}$$

The derivatives with respect to the maneuver time and angle are

$$\frac{\partial \phi_1}{\partial t_m} = \Delta\dot{x}, \quad \frac{\partial \phi_1}{\partial \theta_m} = -\Delta t \dot{y}_{\text{pm}} \tag{24}$$

where $\Delta\dot{x} = \dot{x} - \dot{x}_{\text{pm}}$ (i.e., the difference in x-velocity premaneuver and postmaneuver), and $\Delta t = t - t_m$. The next component in the predicted measurement (Cartesian y-position) is

$$\phi_2 = S_{01}\dot{x} + y_0 + T_{11}\dot{y}. \tag{25}$$

For this, the derivatives with respect to position and velocity are

$$\frac{\partial \phi_2}{\partial x_0} = 0, \quad \frac{\partial \phi_2}{\partial \dot{x}} = S_{01}, \quad \frac{\partial \phi_2}{\partial y_0} = 1, \quad \frac{\partial \phi_2}{\partial \dot{y}} = T_{11}. \tag{26}$$

The maneuver time and angle derivatives are

$$\frac{\partial \phi_2}{\partial t_m} = \Delta\dot{y}, \quad \frac{\partial \phi_2}{\partial \theta_m} = \Delta t \dot{x}_{\text{pm}}. \tag{27}$$

Finally, ϕ_3, the bistatic range-rate, is given by [6], and its derivative components all take the form

$$\frac{\partial \phi_3}{\partial a} = \frac{1}{2}\left(\frac{\partial \phi_3^S}{\partial a} + \frac{\partial \phi_3^R}{\partial a}\right). \tag{28}$$

Here, $a \in \{x_o, \dot{x}, y_0, \dot{y}, t_m, \theta_m\}$, and the S and R superscripts signify the source and receiver components of the expression. Letting $\tilde{r}$ represent the "generic" ϕ_3 (i.e., a common expression that can be used for both the source and the receiver components)

$$\frac{\partial \tilde{r}}{\partial x_0} = \frac{y_{\text{pm}}^2 \dot{x}_{\text{pm}} - x_{\text{pm}} y_{\text{pm}} \dot{y}_{\text{pm}}}{r_{\text{pm}}^3}. \tag{29}$$

Here, once again, r is the range from the target to the source or receiver, and the pm subscripts signify postmaneuver. Furthermore, it is understood that the positions and velocities have the source or receiver positions and velocities subtracted out (depending on the component for which the equation is being used).

The remaining derivatives are

$$\frac{\partial \tilde{r}}{\partial \dot{x}} = \frac{R_{00}(x_{\rm pm}^3 + y_{\rm pm}^2 x_{\rm pm}) + R_{10}(x_{\rm pm}^2 y_{\rm pm} + y_{\rm pm}^3)}{r_{\rm pm}^3} + \cdots \frac{T_{00}(y_{\rm pm}^2 \dot{x}_{\rm pm} - x_{\rm pm} y_{\rm pm} \dot{y}_{\rm pm}) + S_{10}(x_{\rm pm}^2 \dot{y}_{\rm pm} - x_{\rm pm} y_{\rm pm} \dot{x}_{\rm pm})}{r_{\rm pm}^3} \tag{30}$$

$$\frac{\partial \tilde{r}}{\partial y_0} = \frac{x_{\rm pm}^2 \dot{y}_{\rm pm} - x_{\rm pm} y_{\rm pm} \dot{x}_{\rm pm}}{r_{\rm pm}^3} \tag{31}$$

$$\frac{\partial \tilde{r}}{\partial \dot{y}} = \frac{R_{01}(x_{\rm pm}^3 + y_{\rm pm}^2 x_{\rm pm}) + R_{11}(x_{\rm pm}^2 y_{\rm pm} + y_{\rm pm}^3)}{r_{\rm pm}^3} + \cdots \frac{T_{11}(x_{\rm pm}^2 \dot{y}_{\rm pm}^2 - x_{\rm pm} y_{\rm pm} \dot{x}_{\rm pm}) + S_{01}(y_{\rm pm}^2 \dot{x}_{\rm pm} - x_{\rm pm} y_{\rm pm} \dot{y}_{\rm pm})}{r_{\rm pm}^3} \tag{32}$$

$$\frac{\partial \tilde{r}}{\partial t_m} = \frac{(x_{\rm pm}\Delta\dot{y} - y_{\rm pm}\Delta\dot{x})(x_{\rm pm}\dot{y}_{\rm pm} - y_{\rm pm}\dot{x}_{\rm pm})}{r_{\rm pm}^3} \tag{33}$$

$$\frac{\partial \tilde{r}}{\partial \theta_m} = \frac{[x_{\rm pm}^2 + y_{\rm pm}^2 - \Delta t(x_{\rm pm}\dot{x}_{\rm pm} + y_{\rm pm}\dot{y}_{\rm pm})][y_{\rm pm}\dot{x}_{\rm pm} - x_{\rm pm}\dot{y}_{\rm pm}]}{r_{\rm pm}^3}. \tag{34}$$

Expressions for the delay-bearing implementation can be written as functions of the above Cartesian expressions. Introducing the superscripts C and DB to differentiate between the Cartesian and delay-bearing versions, we have,

$$\phi_1^{DB} = \tan^{-1}\left(\frac{\Delta\phi_2^C}{\Delta\phi_1^C}\right) \tag{35}$$

where

$$\Delta\phi_1^C = \phi_1^C - x_R, \qquad \Delta\phi_2^C = \phi_2^C - y_R. \tag{36}$$

Here, x_R and y_R are again the Cartesian positions of the receiver. We can write a generic formula for the Jacobian entries in delay-bearing form that is a function of the respective derivatives from the Cartesian form

$$\frac{\partial \phi_1^{DB}}{\partial a} = \frac{\Delta\phi_1^C \dfrac{\partial \phi_2^C}{\partial a} - \Delta\phi_2^C \dfrac{\partial \phi_1^C}{\partial a}}{r_{TR}^2}. \tag{37}$$

The predicted delay-time measurement is

$$\phi_2^{DB} = \frac{r_{TS} + r_{TR}}{c} \tag{38}$$

where c is the speed of sound. Again, we can write a generic formula for the delay-bearing form of the Jacobian entries from the Cartesian terms

$$\frac{\partial \phi_2^{DB}}{\partial a} = \left[\frac{(x - x_S)\dfrac{\partial \phi_1^C}{\partial a} + (y - y_S)\dfrac{\partial \phi_2^C}{\partial a}}{r_{TS}} + \frac{(x - x_R)\dfrac{\partial \phi_1^C}{\partial a} + (y - y_R)\dfrac{\partial \phi_2^C}{\partial a}}{r_{TR}}\right]. \tag{39}$$

The Jacobian entries for ϕ_3 (range-rate) are identical for the Cartesian and delay-bearing cases.

Finally, the above expressions can also be used for any premaneuver measurements with the following adjustments: $T_{i,j} \to t$, $S_{i,j} \to 0$, and $\Delta t \to 0$. All derivatives with respect to maneuver time and maneuver angle are set to zero.

REFERENCES

[1] Avitzour, D.
A maximum likelihood approach to data association.
IEEE Transactions on Aerospace and Electronic Systems, **28**, 2 (1996), 560–566.

[2] Bar-Shalom, Y., Willett, P., and Tian, X.
Tracking and Data Fusion: A Handbook of Algorithms.
Storrs, CT: YBS Publishing, 2011.

[3] Blanding, W., Willett, P., and Coraluppi, S.
Sequential ML for multistatic sonar tracking.
In *Proceedings of Oceans 2007—Europe*, Aberdeen, Scotland, June 2007.

[4] Chummun, M. R., Bar-Shalom, Y., and Kirubarajan, T.
Adaptive early-detection ML-PDA estimator for LO targets with EO sensors.
IEEE Transactions on Aerospace and Electronic Systems, **38**, 2 (2002), 694–707.

[5] Coraluppi, S.
Mulitstatic sonar localization.
IEEE Journal of Oceanic Engineering, **31**, 4 (2006), 964–974.

[6] Cox, H.
Fundamentals of bistatic active sonar.
BBN Systems and Technologies Corporation, Technical Report W1038, 1988.

[7] Daun, M. and Ehlers, F.
Tracking algorithms for multistatic sonar systems.
EURASIP Journal on Advances in Signal Processing, **2010**, 1 (2010), 1–28.

[8] Georgescu, R., Willett, P., and Schoenecker, S.
GM-CPHD and ML-PDA applied to the Metron multi-static sonar dataset.
In *Proceedings of the 13th International Conference on Information Fusion*, Edinburgh, Scotland, 2010.

[9] Jauffret, C. and Bar-Shalom, Y.
Track formation with bearing and frequency measurements in clutter.
In *Proceedings of the 29th Conference on Decision and Control*, Honolulu, HI, Dec. 1990.

[10] Kirubarajan, T. and Bar-Shalom, Y.
Low observable target motion analysis using amplitude information.
IEEE Transactions on Aerospace and Electronic Systems, **32**, 4 (1996), 1367–1384.

[11] Orlov, K.
Description of the Metron simulation data set. Document provided to the *Multistatic Tracker Working Group* (MSTWG), 2009. Unpublished.

[12] Schoenecker, S., Willett, P., and Bar-Shalom, Y.
Maximum likelihood probabilistic data association tracker applied to bistatic sonar data sets.
In *Proceedings of SPIE Conference on Signal and Data Processing of Small Targets*, vol. 7698-19, Orlando, FL, 2010.

[13] Schoenecker, S., Willett, P., and Bar-Shalom, Y.
A comparison of the ML-PDA and the ML-PMHT algorithms.
In *Proceedings of the 14th International Conference on Information Fusion*, Chicago, IL, 2011.

[14] Schoenecker, S., Willett, P., and Bar-Shalom, Y.
Maximum likelihood probabilistic multi-hypothesis tracker applied to multistatic sonar data sets.
In *Proceedings of SPIE Conference on Signal Processing, Sensor Fusion, and Target Recognition*, vol. 8050-9, Orlando, FL, 2011.

[15] Schoenecker, S., Willett, P., and Bar-Shalom, Y.
Maximum likelihood probabilistic data association (ML-PDA) tracker implemented in delay-bearing space applied to multistatic sonar data sets.
In *Proceedings of SPIE Conference on Signal and Data Processing of Small Targets*, vol. 8393-18, Baltimore, MD, 2012.

[16] Schoenecker, S., Willett, P., and Bar-Shalom, Y.
ML-PDA and ML-PMHT: Comparing multistatic sonar trackers for VLO targets using a new multitarget implementation.
To be published in the *Journal of Oceanic Engineering*, 2012.

[17] Streit, R. and Luginbuhl, T.
A probabilistic multi-hypothesis tracking algorithm without enumeration.
In *Proceedings of the 6th Joint Data Fusion Symposium*, Laurel, MD, June 1993.

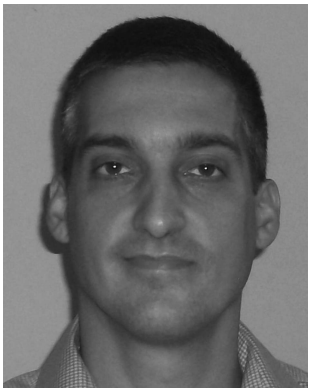
[18] Streit, R. and Luginbuhl, T.
Maximum likelihood method for probabilistic multi-hypothesis tracking.
In *Proceedings of SPIE Conference on Signal and Data Processing of Small Targets*, Orlando, FL, 1994.

[19] Streit, R. and Luginbuhl, T.
Probabilistic multi-hypothesis tracking.
Naval Undersea Warfare Center, Technical Report TR 10428, 1995.

[20] Tian, X. and Bar-Shalom, Y.
Robust tracking for very long range radars, part I: Algorithm comparisons.
In *Proceedings of SPIE Conference on Signal and Data Processing of Small Targets*, San Diego, Aug. 2007.

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

[22] Willett, P., Coraluppi, S., and Blanding, W.
Comparison of soft and hard assignment ML trackers on multistatic data.
In *Proceedings of the IEEE Aerospace Conference*, Big Sky, MT, Mar. 2006.



Steven Schoenecker was born in 1972 in Washington, D.C. He received his A.B. degree in physics from Princeton University in 1995 and his M.S. degree in computer science from Rensselaer at Hartford in 2002. Currently, he is working toward his Ph.D. degree in electrical engineering at the University of Connecticut, working under the direction of Professor P. Willett and Professor Y. Bar-Shalom.

From 1995 to 2002 he was an officer in the United States Navy's submarine force. From 2002 to the present, he has worked at the Naval Undersea Warfare Center in Newport, Rhode Island. His area of interest is statistical signal processing, with current emphasis on multitarget, multisensor tracking.

Peter Willett (S'83—M'86—SM'97—F'03) received his B.A.Sc. (engineering science) from the University of Toronto in 1982, and his Ph.D. degree from Princeton University in 1986.

He has been a faculty member at the University of Connecticut ever since, and since 1998 has been a professor. He was awarded IEEE Fellow status effective 2003. His primary areas of research have been statistical signal processing, detection, machine learning, data fusion, and tracking. He has interests in and has published in the areas of change/abnormality detection, optical pattern recognition, communications and industrial/security condition monitoring.

Dr. Willett was editor-in-chief for *IEEE Transactions on Aerospace and Electronic Systems* from 2006–2011, and is presently the VP of Publications for the IEEE AESS. In the past he has been associate editor for three active journals—*IEEE Transactions on Aerospace and Electronic Systems* (for Data Fusion and Target Tracking) and *IEEE Transactions on Systems, Man, and Cybernetics, parts A and B*. He is also associate editor for the *IEEE AES Magazine*, associate editor for ISIF's electronic *Journal of Advances in Information Fusion*. He was general cochair (with Stefano Coraluppi) for the 2006 ISIF/IEEE Fusion Conference in Florence, Italy, Executive Chair of the 2008 Fusion Conference in Cologne, and Emeritus Chair for the 2011 Fusion Conference in Chicago. He was program cochair (with Eugene Santos) for the 2003 IEEE Conference on Systems, Man & Cybernetics in Washington, D.C., and program cochair (with Pramod Varshney) for the 1999 Fusion Conference in Sunnyvale. He has been a member of the IEEE Signal Processing Society's Sensor-Array & Multichannel (SAM) technical committee since 1997, and both serves on that TC's SAM conferences' program committees and maintains the SAM website.

Yaakov Bar-Shalom (S'63—M'66—SM'80—F'84) was born on May 11, 1941. He received his B.S. and M.S. degrees from the Technion, Israel Institute of Technology, in 1963 and 1967 and his Ph.D. degree from Princeton University, Princeton, NJ, in 1970, all in electrical engineering.

From 1970 to 1976 he was with Systems Control, Inc., Palo Alto, California. Currently he is Board of Trustees Distinguished Professor in the Department of Electrical and Computer Engineering, the Marianne E. Klewin Professor in Engineering, and the director of the ESP (Estimation and Signal Processing) Lab at the University of Connecticut.

His current research interests are in estimation theory, target tracking and data fusion. He has published over 400 papers and book chapters in these areas and in stochastic adaptive control. He coauthored the monograph *Tracking and Data Association* (Academic Press, 1988), the graduate texts *Estimation and Tracking: Principles, Techniques and Software* (Artech House, 1993), *Estimation with Applications to Tracking and Navigation: Algorithms and Software for Information Extraction* (Wiley, 2001), the advanced graduate texts, *Multitarget-Multisensor Tracking: Principles and Techniques* (YBS Publishing, 1995), *Tracking and Data Fusion* (YBS Publishing, 2011), and edited the books *Multitarget-Multisensor Tracking: Applications and Advances* (Artech House, Vol. I, 1990; Vol. II, 1992; Vol. III, 2000).

He has been elected Fellow of IEEE for "contributions to the theory of stochastic systems and of multi-target tracking." He has been consulting to numerous companies and government agencies, and originated the series of Multitarget-Multisensor Tracking short courses offered via UCLA Extension, at Government Laboratories, private companies and overseas.

During 1976 and 1977 he served as Associate Editor of the *IEEE Transactions on Automatic Control* and from 1978 to 1981 as Associate Editor of *Automatica*. He was Program Chairman of the 1982 American Control Conference, General Chairman of the 1985 ACC, and Cochairman of the 1989 IEEE International Conference on Control and Applications. During 1983–1987 he served as Chairman of the Conference Activities Board of the IEEE Control Systems Society and during 1987–1989 was a Member of the Board of Governors of the IEEE CSS. He was a Member of the Board of Directors of the International Society of Information Fusion (1999–2004) and served as General Chairman of FUSION 2000, President of ISIF in 2000 and 2002 and Vice President for publications in 2004–2011.

In 1987 he received the IEEE CSS Distinguished Member Award. Since 1995 he is a Distinguished Lecturer of the IEEE AESS and has given numerous keynote addresses at major national and international conferences. He is corecipient of the M. Barry Carlton Award for the best paper in the *IEEE Transactions on Aerospace and Electronic Systems* in 1995 and 2000 and recipient of the 1998 University of Connecticut AAUP Excellence Award for Research. In 2002 he received the J. Mignona Data Fusion Award from the DoD JDL Data Fusion Group. He is a Member of the Connecticut Academy of Science and Engineering. In 2008 he was awarded the IEEE Dennis J. Picard Medal for Radar Technologies and Applications. He is listed by academic.research.microsoft as first among the researchers in Aerospace Enginering based on the citations of his work.