Multi-Stage Multiple-Hypothesis Tracking

STEFANO CORALUPPI
CRAIG CARTHEL

While multiple hypothesis tracking (MHT) is widely acknowledged as an effective methodology for multi-target surveillance, there is a challenge to manage effectively a potentially large number of track hypotheses. Advanced single-stage *track-while-fuse* does not always offer the best processing scheme. We study two instances where multi-stage MHT processing is beneficial—dense target scenarios and complementary-sensor surveillance—and propose two processing schemes for these challenges: *track-break-fuse* and *track-before-fuse*, respectively. We provide simulation results demonstrating the advantages of these schemes over *track-while-fuse*. More generally, we argue that multi-stage MHT offers a powerful and flexible paradigm to circumvent limitations in conventional MHT processing.

Manuscript received October 26, 2009; revised December 21, 2010 and April 8, 2011; released for publication April 21, 2011.

Refereeing of this contribution was handled by Dale Blair.

Authors' address: Compunetix Inc., 2420 Mosside Boulevard, Monroeville PA 15146, Email: {stefano.coraluppi,craig.carthel}@compunetix.com.

1. INTRODUCTION

A broad overview of approaches to data fusion is provided in [1]. The most powerful current approach to real-time, scan-based data fusion is multi-hypothesis tracking (MHT), which was first introduced in the late 1970s [11] and made feasible in the mid-1980s with the track-oriented approach [9]. A number of enhancements to the basic approach have appeared over the years [1].

If contact measurement information is available at the tracker output, one can think of a multi-target tracker as a *filter* of sorts that discards spurious contacts and associates the remaining ones through track labeling. As such, tracking is a modular operator which, when applied to contact-level data, takes as input singleton (i.e. single-measurement) tracks. More generally, a mix of track-level and contact-level feeds may be provided to the tracker. Upstream track labels are preserved in downstream processing, except in cases where discrepancies are detected in downstream tracking. This tracker modularity allows for arbitrarily complex multi-stage data fusion architectures. This philosophy, combined with the necessary software modularity, is the basis for the multi-stage MHT approach that we consider in this paper. We find that in some applications multi-stage MHT processing outperforms single-stage MHT processing.

In this paper, we introduce two multi-stage MHT architectures and compare these to single-stage, *track-while-fuse* processing. The first multi-stage architecture, *track-break-fuse*, is computationally efficient without sacrificing the tracking performance of *track-while-fuse*. The second architecture, *track-before-fuse*, provides further computational efficiency at the cost of some tracking performance. The *track-while-fuse* approach is intractable when the application requires deep hypothesis trees; conversely, both of the multi-stage MHT approaches that we introduce here identify a small set of relevant association hypotheses, enabling deep hypothesis trees.

The paper is organized as follows. In Section 2, we provide a short introduction to standard (*track-while-fuse*) track-oriented MHT, following closely on the formalism introduced in [9]. The multi-stage MHT architectures of interest, *track-break-fuse* and *track-before-fuse*, are introduced in Section 3. In Section 4 we study *track-break-fuse* for a challenging, slowly-crossing targets problem. In Section 5 we study *track-before-fuse* for multi-sensor surveillance with complementary, multi-scale sensors. Concluding remarks are in Section 6.

Early results on the multi-stage processing introduced here are in [6] (*track-break-fuse*) and [3] (*track-before-fuse*). A related MHT approach to *track-before-fuse* is discussed in [4], which introduces group-tracking logic to enable deep hypothesis trees. Additionally, within the MHT framework, some techniques to hypothesis management do exist, including K-best assignment or hypothesis-clustering approaches [7, 10]. However,

the difficulty is that one must maintain *relevant* track hypotheses for significant time duration, and often the top-scoring global hypotheses do not have sufficient diversity to insure that this is achieved. Hypothesis clustering may only partially ameliorate the situation. Possibly, one might adopt a probabilistic data association framework to address the problem, leveraging the approach introduced in [8].

2. TRACK-ORIENTED MULTIPLE HYPOTHESIS TRACKING

A key challenge in multi-sensor multi-target tracking is *measurement origin uncertainty*. That is, unlike a classical nonlinear filtering problem, we do not know how many objects are in the surveillance region, and which measurements are to be associated. New objects may be born in any given scan, and existing objects may die.

We assume that for each sensor scan, contact-level (or detection-level) data are available, in the sense that signal processing techniques are applied to raw sensor data yielding contacts for which the detection and localization statistics are known. We are interested in a scan-based (or real-time) approach that, perhaps with some delay, yields an estimate of the number of objects and corresponding object state estimates at any time.

Several approaches to contact-level scan-based tracking exist. In this section, we employ a hybrid-state formalism to describe the track-oriented multiple-hypothesis tracking approach. Our approach follows closely the one introduced in [9]. We assume Poisson distributed births at each scan with mean λ_b, Poisson distributed false returns with mean λ_{fa}, object detection probability p_d, object death or termination probability p_χ at each scan. (We neglect the *time-dependent* nature of birth and death probabilities as would ensue from an underlying continuous-time formulation, and we neglect as well inter-scan birth and death events.)

We have a sequence of sets of contacts $Z^k = (Z_1, \ldots, Z_k)$, and we wish to estimate the state history X^k for all objects present in the surveillance region. X^k is compact notation that represents the state trajectories of targets that exist over the time sequence $(t_1, \ldots, t_k)$. Note that each target may exist for a subset of these times, with a single birth and a single death occurrence i.e. targets do not reappear. We introduce the auxiliary discrete state history q^k that represents a full interpretation of all contact data: which contacts are false, how the object-originated ones are to be associated, and when objects are born and die. There are two fundamental assumptions of note. The first is that there are no target births in the absence of a corresponding detection, i.e. we do not reason over new, undetected objects. The second is that there is *at most* one contact per object per scan.

We are interested in the probability distribution $p(X^k \mid Z^k)$ for object state histories given data. This quantity can be obtained by conditioning over all possible auxiliary states histories q^k.

$$p(X^k \mid Z^k) = \sum_{q^k} p(X^k, q^k \mid Z^k) = \sum_{q^k} p(X^k \mid Z^k, q^k) p(q^k \mid Z^k). \quad (1)$$

A pure MMSE approach would yield the following:

$$\hat{X}_{\text{MMSE}}(Z^k) = E[X^k \mid Z^k] = \sum_{q^k} E[X^k \mid Z^k, q^k] p(q^k \mid Z^k). \quad (2)$$

The track-oriented MHT approach is a mixed MMSE/MAP one, whereby we identify the MAP estimate for the auxiliary state history q^k, and identify the corresponding MMSE estimate for the object state history X^k conditioned on the estimate for q^k.

$$\hat{X}(Z^k) = \hat{X}_{\text{MMSE}}(Z^k, \hat{q}^k) \quad (3)$$

$$\hat{q}^k = \hat{q}_{\text{MAP}}(Z^k) = \arg\max_{q^k} p(q^k \mid Z^k). \quad (4)$$

Each feasible q^k corresponds to a global hypothesis. (The set of global hypotheses is generally constrained via measurement gating and hypothesis generation logic.) We are interested in a *recursive* and *computationally efficient* expression for $p(q^k \mid Z^k)$ that lends itself to functional optimization without the need for explicit enumeration of global hypotheses. We do so through repeated use of Bayes' rule. Note that, for notational simplicity, we use $p(\cdot)$ for both probability density and probability mass functions. Also, as discussed in Section 2, the normalizing constant c_k does not impact MAP estimation.

$$p(q^k \mid Z^k) = \frac{p(Z_k \mid Z^{k-1}, q^k) p(q^k \mid Z^{k-1})}{c_k} = \frac{p(Z_k \mid Z^{k-1}, q^k) p(q_k \mid Z^{k-1}, q^{k-1}) p(q^{k-1} \mid Z^{k-1})}{c_k} \quad (5)$$

$$c_k = p(Z_k \mid Z^{k-1}) = \sum_{q^k} p(Z_k \mid Z^{k-1}, q^k) p(q^k \mid Z^{k-1}). \quad (6)$$

Recall that we assume that in each scan the number of target births is Poisson distributed with mean λ_b, the number of false returns is Poisson distributed with mean λ_{fa}, targets die with probability p_χ, and targets are detected with probability p_d. The recursive expression (5) involves two factors that we consider in turn, with the discrete state probability one first. It will be useful to introduce the aggregate variable ψ_k (consistent with the approach in [9]) that accounts for the number of detections d for the τ existing tracks, the number of track deaths χ, the number of new tracks b, and the

number of false returns $r-d-b$, where r is the number of contacts in the current scan.

$$p(q_k \mid Z^{k-1}, q^{k-1}) = p(\psi_k \mid Z^{k-1}, q^{k-1}) p(q_k \mid Z^{k-1}, q^{k-1}, \psi_k) \tag{7}$$

$$p(\psi_k \mid Z^{k-1}, q^{k-1}) = \left\{ \binom{\tau}{\chi} p_\chi^\chi (1-p_\chi)^{\tau-\chi} \right\} \cdot \left\{ \binom{\tau-\chi}{d} p_d^d (1-p_d)^{\tau-\chi-d} \right\} \cdot \left\{ \frac{\exp(-\lambda_b) p_d^b \lambda_b^b}{b!} \right\} \cdot \left\{ \frac{\exp(-\lambda_{fa}) \lambda_{fa}^{r-d-b}}{(r-d-b)!} \right\} \tag{8}$$

$$p(q_k \mid Z^{k-1}, q^{k-1}, \psi_k) = \frac{1}{\binom{\tau}{\chi} \binom{\tau-\chi}{d} \left(\frac{r!}{(r-d)!} \right) \binom{r-d}{b}}. \tag{9}$$

Substituting (8–9) into (7) and simplifying yields the following.

$$p(q_k \mid Z^{k-1}, q^{k-1}) = \left\{ \frac{\exp(-\lambda_b - \lambda_{fa}) \lambda_{fa}^r}{r!} \right\} p_\chi^\chi ((1-p_\chi)(1-p_d))^{\tau-\chi-d} \left(\frac{(1-p_\chi) p_d}{\lambda_{fa}} \right)^d \left(\frac{p_d \lambda_b}{\lambda_{fa}} \right)^b. \tag{10}$$

The first factor in (5) is given below, where $Z_k = \{z_j, 1 \le j \le r\}$, $|J_d| + |J_b| + |J_{fa}| = r$, and the factors on the R.H.S. are derived from filter innovations, filter initiations, and the false contact distribution (generally uniform over measurement space). For example, in the linear Gaussian case, $f_d(z_j \mid Z^{k-1}, q^k)$ is a Gaussian residual, i.e. it is the probability of observing z_j given a sequence of preceding measurements. If there is no prior information on the target, $f_b(z_j \mid Z^{k-1}, q^k)$ is generally the value of the uniform density function over measurement space. Similarly, $f_{fa}(z_j \mid Z^{k-1}, q^k)$ is as well usually taken to be the value of the uniform density function over measurement space, under the assumption of uniformly distributed false returns. Note that the expressions given here are general and allow for quite general target and sensor models.

$$p(Z_k \mid Z^{k-1}, q^k) = \prod_{j \in J_d} f_d(z_j \mid Z^{k-1}, q^k) \cdot \prod_{j \in J_b} f_b(z_j \mid Z^{k-1}, q^k) \cdot \prod_{j \in J_{fa}} f_{fa}(z_j \mid Z^{k-1}, q^k). \tag{11}$$

Substituting (10–11) into (5) and simplifying results in (12–13). This expression is the key enabler of track-oriented MHT. In particular, it provides a recursive expression for $p(q^k \mid Z^k)$ that consists of a number of factors that relate to its constituent local track hypotheses.

An implicit reduction in the set of hypotheses in (12–13) is that target births are assumed to occur only in the presence of a detection (i.e. there is no reasoning over un-detected births). Correspondingly, the factor p_d reduces the effective birth rate to $p_d \lambda_b$ (though surprisingly the factor is absent in [9]). Further, in the first scan of data, it would be appropriate to replace $p_d \lambda_b$ by $p_d \lambda_b / p_\chi$ to account properly for the steady-state expected number of targets. (More generally, target birth and death parameters should reflect sensor scan rates, as the underlying target process is defined in continuous time.) Further reduction in the set of hypotheses is generally achieved via *measurement gating* procedures [1]. Finally, for a given track hypothesis, one usually applies rule-based spawning of a missed detection or termination hypothesis, but not both (e.g. only spawn a missed detection hypothesis until a sufficiently-long sequence of missed detection is reached).

One cannot consider too large a set of scans before pruning or merging local (or track) hypotheses in some fashion. A popular mechanism to control these hypotheses is *n-scan pruning*. This amounts to solving (4), generally by a relaxation approach to an integer programming problem [4, 6, 11], followed by pruning of all local hypotheses that differ from $\hat{q}^k$ in the first scan. This methodology is applied after each new scan

$$p(q^k \mid Z^k) = p_\chi^\chi ((1-p_\chi)(1-p_d))^{\tau-\chi-d} \prod_{j \in J_d} \left[\frac{(1-p_\chi) p_d f_d(z_j \mid Z^{k-1}, q^k)}{\lambda_{fa} f_{fa}(z_j \mid Z^{k-1}, q^k)} \right] \prod_{j \in J_b} \left[\frac{p_d \lambda_b f_b(z_j \mid Z^{k-1}, q^k)}{\lambda_{fa} f_{fa}(z_j \mid Z^{k-1}, q^k)} \right] \frac{p(q^{k-1} \mid Z^{k-1})}{\bar{c}_k} \tag{12}$$

$$\bar{c}_k = \frac{c_k}{\left\{ \frac{\exp(-\lambda_b - \lambda_{fa})}{r!} \lambda_{fa}^r \right\} \prod_{j \in J_d \cup J_b \cup J_{fa}} f_{fa}(z_j \mid Z^{k-1}, q^k)}. \tag{13}$$

of data are received, resulting in a fixed-delay solution to the tracking problem.

Often, *n-scan* pruning is referred to as a *maximum likelihood* (ML) approach to hypothesis management. ML estimation is closely related to *maximum a posteriori* (MAP) estimation. In particular, we have:

$$\hat{X}_{\mathrm{MAP}}(y) = \arg\max f(y \mid X) f(X) \tag{14}$$

$$\hat{X}_{\mathrm{ML}}(y) = \arg\max f(y \mid X). \tag{15}$$

Note that ML estimation is a *non-Bayesian* approach as it does not rely on a prior distribution on X. ML estimation can be interpreted as MAP estimation with a uniform prior. In the track-oriented MHT setting, n-scan pruning relies on a single parent global hypothesis, thus the ML and MAP interpretations are both valid.

Once hypotheses are resolved, in principle one has a state of object histories given by $\hat{X}(Z^k)$. In practice, it is common to apply track confirmation and termination logic to all object histories [1]. A justification for this is that it provides a mechanism to remove spurious tracks induced by the sub-optimality inherent in practical MHT implementations that include limited hypothesis generation and hypothesis pruning or merging.

Given the need for post-association track confirmation and termination logic, a reasonable simplification that is pursued in [5] is to employ equality constraints in the data-association process, which amounts to accounting for all contact data in the resolved tracks. Spurious tracks are subsequently removed in the track-extraction stage.

3. MULTI-STAGE MHT

Multi-stage fusion as performed here has two defining characteristics that differ from many legacy systems that exist today [1]. The first is that each tracker module retains measurement-level information at the output. That is, each module performs the following: it removes large numbers of measurement data, and associates the remaining measurements to form tracks over time. If the tracker is working well and the data are of reasonable quality, false measurements will largely be removed and target-originated measurements will mostly be maintained and associated into tracks that persist over time with limited fragmentation. Since measurement data are available at the tracker output, optimal track fusion and state estimation is achievable in downstage tracker modules; the cost to achieve this performance benefit is a slightly larger bandwidth requirement between processing stages. The second defining characteristic is that track fusion is achieved in a real-time, scan-based manner. Often, track fusion is performed in a post-processing batch mode that is not readily amenable to real-time surveillance application [1].

The theoretical optimality of unified, batch and centralized approaches to fusion and tracking (*track-while-fuse*) is at odds with a number of practical considerations. Principally, in many surveillance settings optimal processing algorithms are either not known, or are computationally infeasible. Thus, improved performance often can be achieved with multi-stage processing that involves simpler and less computationally intensive algorithms than with centralized processing.

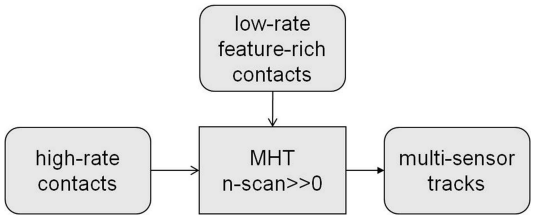

Fig. 1. *Track-while-fuse:* single-stage processing.

The multi-stage paradigm is seemingly at odds with fundamental results in the nonlinear filtering and distributed detection literature [13]. However, this is not actually the case. Rather, multi-stage approaches may outperform single-stage ones for two reasons: (1) like all trackers, single-stage tracking approaches are necessarily sub-optimal as they must contend with measurement origin uncertainty; and (2) measurement information is carried to downstream stages of multi-stage processing. As such, multi-stage processing as defined here is not in fact an instance of distributed processing.

A systems representation of *track-while-fuse*, *track-break-fuse*, and *track-before-fuse* is illustrated in Figs. 1–3. Note that use of these architectures need not require the availability of multi-sensor feeds: indeed our application of *track-break-fuse* (Section 4) is in the context of single-sensor surveillance.

In our two approaches to *multi-stage* MHT processing, we first seek to identify relevant, target-originated contact-level data from the high-rate sensor in a first tracker processing stage. We are not particularly concerned that multi-target interactions be handled properly. Indeed, the first-stage tracker need not be an MHT module, though it is important that it perform hard data association and that the module provide contact-level data associated with tracks.

In many multi-sensor settings, one has high-rate sensors (perhaps providing a scan every several seconds) that provide detection information but without much target feature information, if any. An example of this is a surveillance radar. Additionally, one may have a low-rate sensor (perhaps providing a scan every several hours) that provides detection information with significant target features, or attributes, besides kinematic information. An example of this is *synthetic aperture radar* (SAR) imagery that may provide target dimensions or target type. A standard single-stage processing architecture is illustrated in Fig. 1.

In the *track-break-fuse* architecture (Fig. 2), track labels are removed from the single-sensor tracks, and the resulting contact-level data are fed to the second-stage tracker along with the low-rate feature-rich contacts

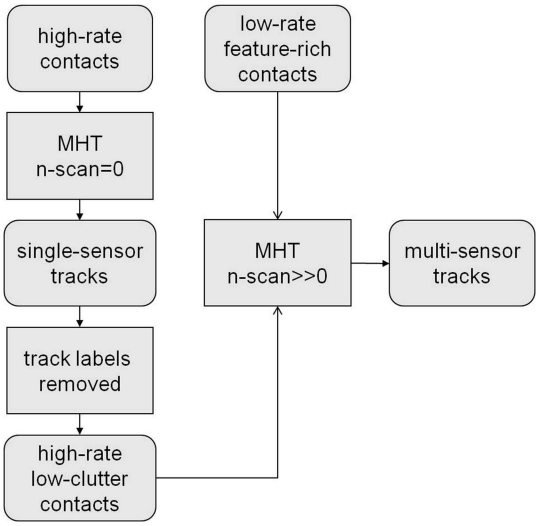

Fig. 2. *Track-break-fuse:* multi-stage processing with removal of track labels after first tracker module.

from the second sensor (if available). With the *track-before-fuse* architecture (Fig. 3), single-sensor tracks are fed to the second-stage MHT module along with the low-rate feature-rich contacts from the second sensor.

An important requirement for the *track-before-fuse* architecture is to have track-breakage logic in the second tracking module. Specifically, in instances in which the first-stage tracking has incorrectly introduced a track swap, subsequent feature information may identify that a tracking error has occurred. We wish to recover *gracefully* from the error, without the time-rollback solution that is operationally infeasible in large-scale surveillance. Error identification is prevalent when the feature-rich sensor has unambiguous target identification information. Correspondingly, when an infeasible update to a fused track is received, the fused track is terminated, and each upstream track of which the fused track is composed is subsequently treated as a new input track and made available for fusion with other fused tracks or for fused track initiation.

4. DENSE TARGET SCENARIOS AND THE TRACK-BREAK-FUSE ARCHITECTURE

A fundamental difficulty in target tracking is multi-target ambiguity, exhibited for example in a slowly-crossing target scenario. We find that MHT processing with large hypothesis tree depths improves tracking performance including a reduction of track swap occurrences. However, this poses a significant processing challenge as deep hypothesis trees are required. Indeed, hypothesis trees must be deep enough and rich enough so that the (local) track hypotheses associated with crossing and non-crossing tracks are maintained until ambiguities are resolved.

We will see that two-stage MHT processing with a *track-break-track* architecture does not impact tracking performance, it provides a dramatic computational benefit.

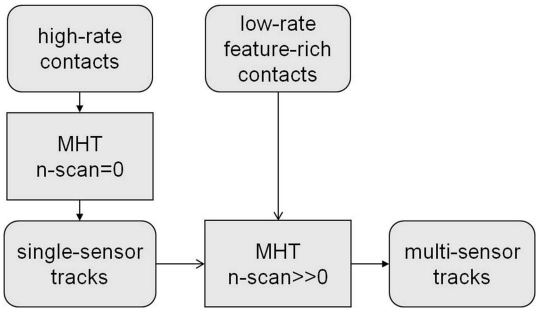

Fig. 3. *Track-before-fuse:* multi-stage processing with logical track breakage as needed in second tracker module.

The *track-break-track* architecture includes a first stage of tracking, followed by removal of all track label information and a second stage of MHT tracking applied only to those contacts that are included in the first-stage tracks. Thus, the first tracking stage can be regarded as a *filter* that identifies target-originated contacts. Multi-target association ambiguities are resolved in the second stage. The motivation for this architecture is that the first stage of processing can be executed quite effectively with no or small hypothesis tree depth, while the second stage requiring a larger hypothesis tree depth contends with much less contact data. Thus, we expect and find comparable performance to single-stage MHT in crossing-target scenarios, but at significant computation savings.

We now study the *percentage of success* for the crossing-target scenario, with *track-while-fuse* and *track-break-fuse* architectures and a range of hypothesis tree depths (*n-scan*). A key issue in this study is how we define *success* in a way that captures successful tracking through the target-crossing event. For the scenario of interest, this is well-captured by requiring a *track hold* or *track* P_D that exceeds 75 + % (note that in the case of a track swap, tracks are classified as false.)

Key parameters in this simulation are the following:

- *Target:* angle of approach = 22 deg; speed = 76.5 m/s;
- *Sensor:* $P_D = 0.7$, FAR = 10/scan; positional measurement error standard deviation—1 m in both x and y; scan rate = 1 Hz; number of scans = 150;
- *Tracker:* process noise = 10^{-3} m^2s^{-3}; initiation rule: 4-of-4; termination rule: 4 misses; association gate = 99%;
- *Monte Carlo settings:* 500 realizations of sensor data are generated. For each, six *track-while-fuse* tracker executions are performed (with *n-scan* from 0 to 5), as well as six *track-break-track* executions (*n-scan* = 0 in first stage, *n-scan* from 0 to 5 in the second stage).

Fig. 4 illustrates execution timing results. As expected, for small *n*-scan values, centralized tracking is faster. For larger *n*-scan values, *track-break-track* is faster. Fig. 5 illustrates tracking performance. With both tracking architectures, we find that there is increased tracking performance with increasing hypothesis depth, at the cost of increased execution time. We see

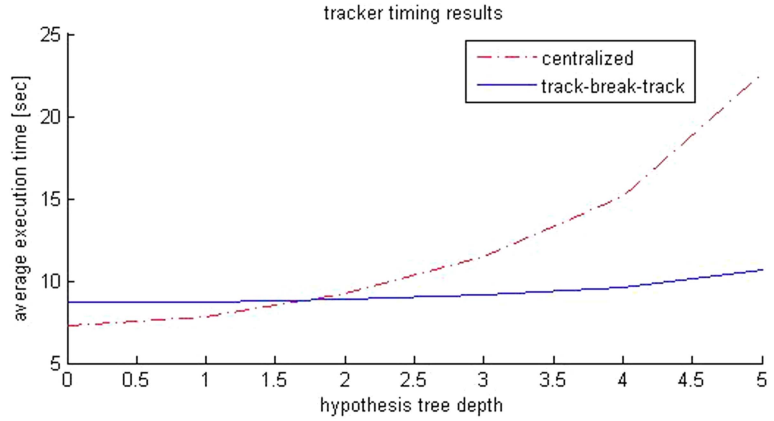

Fig. 4. Tracker timing results as a function of architecture and *n-scan* setting.

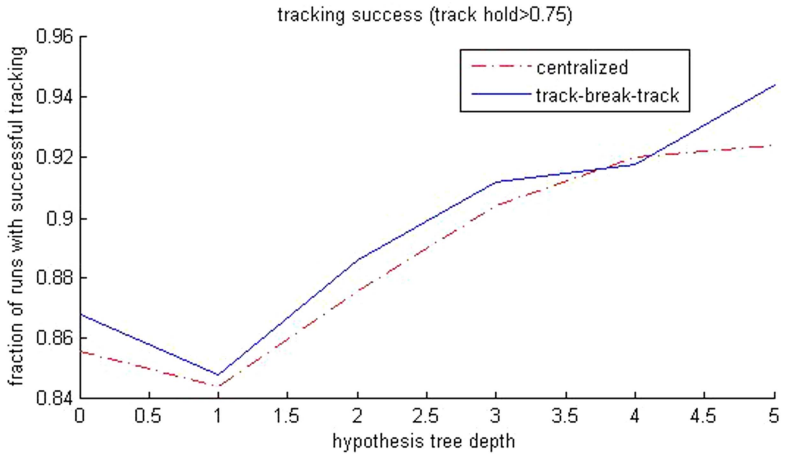

Fig. 5. Tracking performance as a function of architecture and *n-scan* setting.

that *track-break-fuse* matches the performance of *track-while-fuse* with significant computational savings.

An illustration of two tracker outputs for one run in the simulation study is illustrated in Figs. 6–7. Fig. 6 illustrates the sensor footprint with false contacts (black dots) and target-induced contacts (magenta dots). Target trajectories are in magenta, *track-while-fuse* tracks are in blue and red, indicating true and false tracks, respectively. (Note that the track swap occurrence is classified as a false track.) The (true) tracks resulting from *track-break-fuse* processing (*n-scan* = 5) are in green.

5. MULTI-SENSOR SURVEILLANCE AND THE TRACK-BEFORE-FUSE ARCHITECTURE

Fig. 8 illustrates the feature-aided tracking problem. We consider a situation where a high revisit rate sensor, e.g. surveillance radar, provides contact data to the fusion center. A second sensor provides contact data intermittently. Examples for the second sensor might include a SAR imaging sensor, a passive signal intelligence (SIGINT) sensor, or a passive transponder-based sensor such as the *automatic identification system* (AIS) in the maritime domain [12]. In the figure, target trajectories are shown in black. Surveillance radar contacts are shown in blue and black, for target detections and false returns, respectively. Intermittent, feature-rich returns are shown in red.

We consider again a target-crossing scenario. Sensor 1 has coverage of the entire surveillance region, while sensor two has coverage over a subset of the region that does not include the target-crossing event. The two sensors have the same nominal revisit rate, but sensor 2 is intermittent due to the more limited coverage. Both sensors provide positional measurements.

Sensor 2 is representative of a transponder-based passive sensor, like AIS. As such, it has a high revisit rate but intermittent coverage. While the detection probability is non-unity due to electromagnetic propagation effects and measurement collision with the time-division message allocation scheme [2], the false alarm rate is zero. Consistent with AIS data, sensor two provides precise target identification information.

Simulation parameters are given in Table I. Note that the track management criteria in the *track-while-fuse* and 1st stage of the multi-stage architectures differ slightly: this is done to achieve comparable data rates, and is required to account for intermittent coverage of sensor 2. The hypothesis tree depths in *track-while-fuse* and in the 2nd stage of the multi-stage architectures are chosen to be the same and sufficient to ensure good tracking

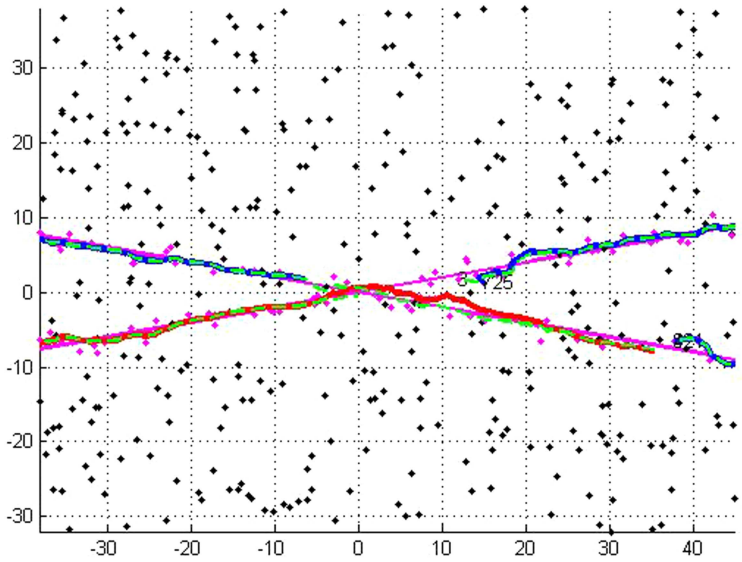

Fig. 6. One realization of sensor data, with two tracker outputs.

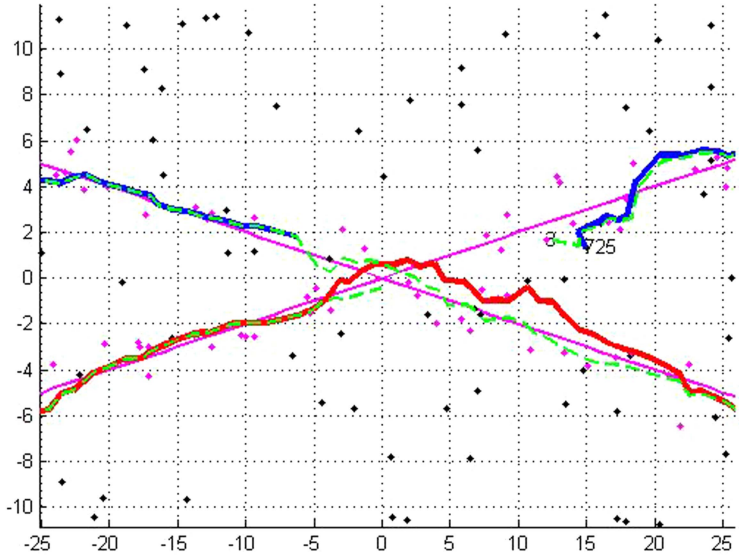

Fig. 7. Close-up view of *track-while-fuse* processing (red, blue) and *track-break-fuse* processing (green).

performance given the gap in sensor 2 coverage. The track break parameter is used to identify infeasible track updates that initiate fused track termination in the track-before-fuse architecture. The track classification metric is relevant to performance evaluation, as it is used to identify true tracks based on average localization error.

One scenario realization is illustrated in Figs. 9–12. In this instance, track swap has occurred in single-sensor (sensor 1) tracking. We see that single-stage *track-while-fuse* architecture does not exhibit the swap, nor does the multi-stage *track-break-fuse* architecture. The *track-before-fuse* architecture recovers from the up-stream track swap by fragmenting the fused tracks, under the track-breakage logic described in Section 3.

As noted above, tracker performance evaluation relies on a track classification step whereby those tracks

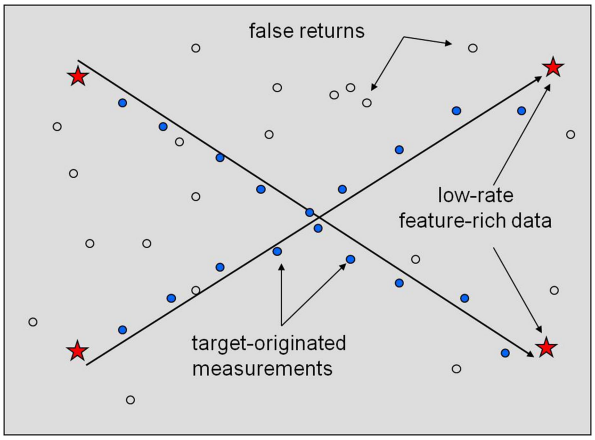

Fig. 8. A notional illustration of the feature-aided tracking problem.

TABLE I
Parameter Settings for the Simulation Study

Parameter	Setting
Monte Carlo realizations	200
Scenario duration	150 sec
Target number	2
Target start locations	$(-75$ m, 5 m), $(-75$ m, -5 m)
Target velocities (until crossing)	(1 m/s, -0.067 m/s), (1 m/s, 0.067 m/s)
Target velocities (after crossing)	(1 m/s, -0.33 m/s), (1 m/s, 0.33 m/s)
Sensor 1 footprint	$(180 \text{ m})^2$
Sensor 1 scan rate	1 Hz
Sensor 1 detection probability	0.8
Sensor 1 false alarm rate per scan	20
Sensor 1 measurement error covariance	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ m^2
Sensor 2 footprint	$(180 \text{ m})^2$ minus central swath, $\lvert x \rvert < 5$ m
Sensor 2 scan rate	1 Hz
Sensor 2 detection probability	0.8
Sensor 2 false alarm rate per scan	0
Sensor 2 measurement error covariance	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ m^2
Track filter process noise parameter	0.1 m^{2}s^{-3}
Track filter prior velocity covariance	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ m^{2}s^{-2}
Track correlation gate	99%
Track break (2nd stage track-before-fuse)	99.99%
Track initiation (track-while-fuse)	3-of-4
Track initiation (1st stage track-before-fuse & track-break-fuse)	4-of-4
Track kill (track-while-fuse)	6 misses
Track kill (1st stage track-before-fuse & track-break-fuse)	4 misses
Track kill (2nd stage track-before-fuse & track-break-fuse)	6 misses
N-scan (track-while-fuse)	10
N-scan (1st stage track-before-fuse & track-break-fuse)	0
N-scan (2nd stage track-before-fuse & track-break-fuse)	10
Track classification distance threshold	2 m

with sufficiently large average localization error with respect to all target trajectories are classified as false; otherwise, the closest target trajectory is identified. Subsequently, the following performance metrics are identified:

- *Track hold* (PD): ratio of total true track duration and total trajectory duration;
- *False track rate* (FAR): average number of false tracks [hr^{-1}];
- *Track fragmentation* (FRAG): average number of true tracks per tracked target;
- *Track localization error* (ERROR): average positional error between a true track and the corresponding target trajectory [m];
- *Tracker execution time* (TIME): average tracker execution time on a DELL OPTIPLEX GX620 with Intel Pentium D processor [sec]; note that 150 sec corresponds to real-time processing.

TABLE II
Performance Results

	Track-While-Fuse	Track-Before-Fuse	Track-Break-Fuse
PD	0.6067	0.9866	0.9861
FAR	23.40	22.07	42.35
FRAG	1.2229	2.1575	1.4175
ERROR	1.0348	1.061	0.9449
TIME	119.58	21.12	35.74

Performance results for the three feature-aided tracking architectures of interest are in Table II. An assessment of these results leads to the following conclusion:

- In terms of *track detection performance* (PD, FAR), *track-before-fuse* outperforms both *track-while-fuse* and *track-break-fuse*;
- In terms of *track continuity* (FRAG), the finding is reversed: *track-while-fuse* and *track-break-fuse* outperform *track-before-fuse*;
- In terms of *track accuracy* (ERROR), the architectures are comparable;
- In terms of *track computational load* (TIME), both multi-stage architectures perform significantly better than *track-while-fuse*.

Overall, we find that the *track-while-fuse* architecture has good performance, but it is not scalable to large hypothesis tree depths as would be required if the feature-rich sensor has a very low revisit rate, or has intermittent coverage with significant special gaps.

From a target-detection perspective, *track-break-fuse* performs comparably to *track-while-fuse*. In the simulation study, both track PD and track FAR are higher in *track-break-fuse*, since the effective track-level receiver operating characteristics (ROC) curve operating point is different between the two architectures. Similarly, the track fragmentation rate (FRAG) is roughly comparable. We conclude that the two architectures yield comparable tracking performance, but the *track-break-fuse* architecture exhibits significant computational savings.

The *track-before-fuse* architecture exhibits better target-detection performance than *track-break-fuse*. This

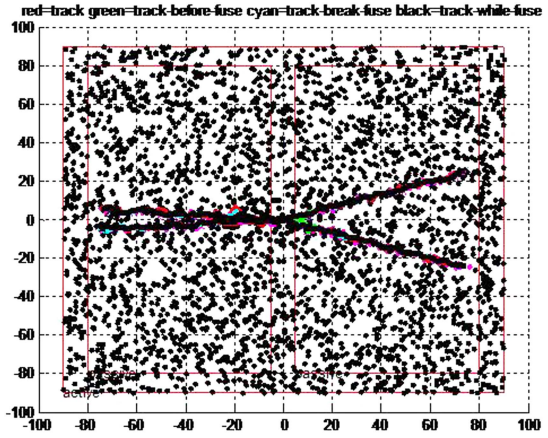

Fig. 9. One scenario realization.

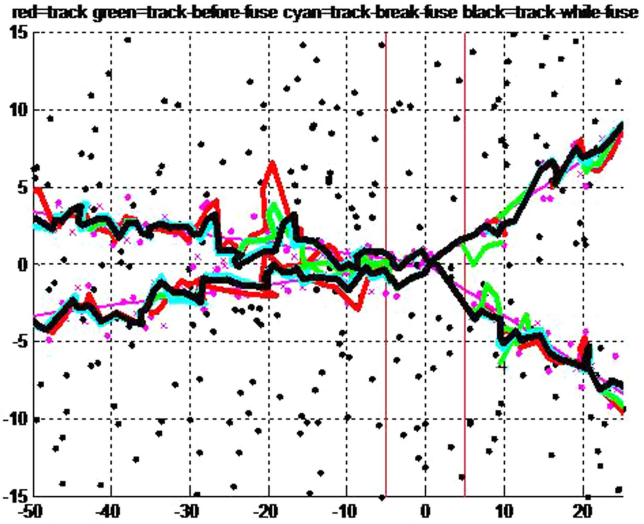

Fig. 10. Same realization as Fig. 9, with close-up on target crossing. Single-sensor (red) tracks exhibit swapping; track-before-fuse (green) exhibits track fragmentation after targets enter region of sensor 2 (feature-rich) coverage; both *track-break-fuse* (cyan) and *track-while-fuse* (black) are successful.

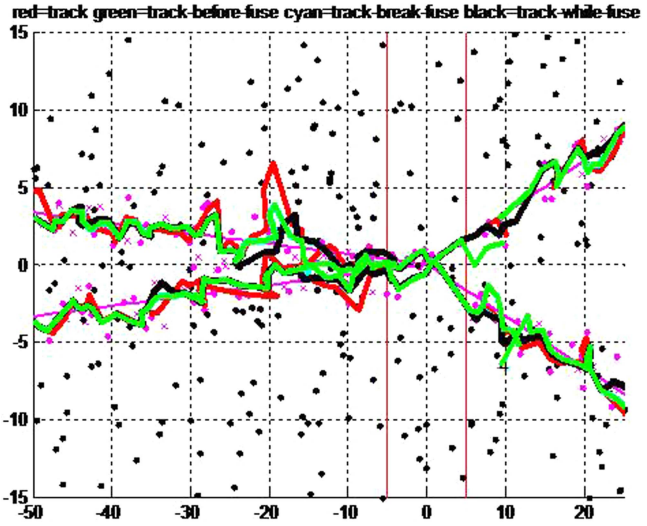

Fig. 11. Same as Fig. 9, with *track-before-fuse* result on top overlay (note fragmentation).

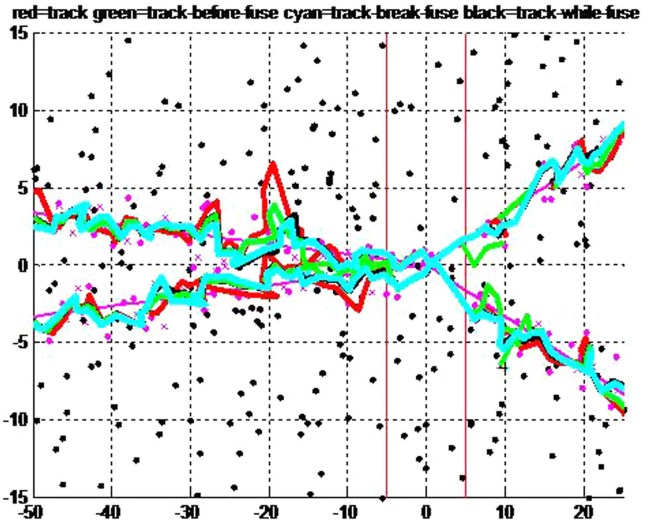

Fig. 12. Same realization as Fig. 9, with *track-break-fuse* result on top overlay (similar to *track-while-fuse*, with no fragmentation).

can be explained as follows: unlike track-break-fuse, there is no need to reacquire track in the second processing stage, since track associations are preserved. On the other hand, its track fragmentation rate is worse than that achieved with *track-break-fuse*, since first-stage association errors, when detected, lead to track termination and correspondingly to an increase in the overall fragmentation rate. As expected, the track-before-fuse architecture is the most efficient from a computational perspective.

To conclude, *track-while-fuse* is not scalable to large scenarios and large hypothesis-tree depths. Two feasible alternatives are *track-break-fuse* and *track-before-fuse*. Depending on the application, one or the other of these may be best.

6. CONCLUSIONS AND FUTURE DIRECTIONS

This paper proposes two multi-stage architectures for challenging surveillance problems that include dense target scenarios and multi-scale or intermittent multi-sensor coverage. In single-sensor benchmarking, the *track-break-fuse* architecture provides the same performance as track-while-fuse but with significant computational savings. In multi-sensor benchmarking, both multi-stage architectures achieve comparable track-level detection, localization, and track-continuity performance as single-stage, *track-while-fuse* processing. However, both architectures do so with dramatically reduced execution times. Further, the multi-stage architectures are extensible to larger hypothesis tree depths, while the single-stage architecture is not. The first multi-stage architecture, *track-before-fuse*, has higher track fragmentation than single-stage processing. The second multi-stage architecture, *track-break-fuse*, achieves comparable track fragmentation as in single-stage processing, at the cost of a small computational increase over *track-before-fuse*.

Future work will include analysis of an architecture that includes *some* track breakage after first-stage tracking, but less than the complete breakage prescribed under the track-break-fuse architecture. As such, this *hybrid* architecture should trade off the benefits of the *track-break-fuse* architecture (limited sensitivity to first-stage tracking errors) with those of the *track-before-fuse* architecture (computational savings, particularly in scenarios where multi-target association ambiguities persist for a long time).

Such a (hybrid) architecture would provide improved surveillance performance and would be particularly applicable to large sensor surveillance networks, where ambiguities may persist for a very large number of sensor scans, thus providing a flexible architecture for large-scale surveillance. The approach shares similarities with [4] but without requiring group-tracking logic.

REFERENCES

[1] S. Blackman and R. Popoli
Design and Analysis of Modern Tracking Systems.
Artech House, 1999.

[2] T. Calmettes, R. Challamel, O. Autran, and B. Garnier
Space-based AIS: Solutions to the simultaneous access issue.
In *Proceedings of the 3rd Annual Maritime Systems and Technology Conference*, Cadiz, Spain, Nov. 2008.

[3] C. Carthel, S. Coraluppi, K. Bryan, and G. Arcieri
Wide-area feature-aided tracking with intermittent multi-sensor data.
In *Proceedings of the SPIE Conference on Signal and Data Processing of Small Targets*, Orlando FL, Apr. 2010.

[4] C-Y. Chong, et al.
Efficient multiple hypothesis tracking by track segment graph.
In *Proceedings of the 12th International Conference on Information Fusion*, Seattle, WA, July 2009.

[5] S. Coraluppi and C. Carthel
Recursive track fusion for multi-sensor surveillance.
Information Fusion, **5**, 1 (Mar. 2004).

[6] S. Coraluppi, C. Carthel, P. Willett, M. Dingboe, O. O'Neill, and T. Luginbuhl
The track repulsion effect in automatic tracking.
In *Proceedings of the 12th International Conference on Information Fusion*, Seattle, WA, July 2009.

[7] S. Gadaleta, A. Poore, S. Roberts, and B. Slocumb
Multiple hypothesis clustering and multiple frame assignment tracking.
In *Proceedings of the SPIE Conference on Signal and Data Processing of Small Targets*, San Diego, CA, Aug. 2004.

[8] P. Horridge and S. Maskell
A scalable method of tracking targets with dependent distributions.
In *Proceedings of the 12th International Conference on Information Fusion*, Seattle, WA, July 2009.

[9] T. Kurien
Issues in the design of practical multi-target tracking algorithms.
In Y. Bar-Shalom (Ed.), *Multi-Target Multi-Sensor Tracking: Advanced Applications*, Artech House, 1990.

[10] R. Popp, K. Pattipati, and Y. Bar-Shalom
M-best S-D assignment algorithm with application to multitarget tracking.
IEEE Transactions on Aerospace and Electronic Systems, **37**, 1 (Jan. 2001).

[11] D. Reid
An algorithm for tracking multiple targets.
IEEE Transactions on Aerospace and Electronic Systems, **24**, 4 (Dec. 1979).

[12] B. Tetreault
Use of the automatic identification system (AIS) for maritime domain awareness.
In *Proceedings of OCEANS 2005*, Washington, D.C., Sept. 2005.

[13] J. Tsitsiklis
Decentralized detection.
In *Advances in Signal Processing*, vol. 2, JAI Press, 1993.

Stefano Coraluppi received the B.S. degree in electrical engineering and mathematics from Carnegie Mellon University, Pittsburgh, PA in 1990, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland, College Park, MD in 1992 and 1997.

He has held research staff positions at ALPHATECH Inc. in Burlington, MA (1997–2002), the NATO Undersea Research Centre in La Spezia, Italy (2002–2010), and Compunetix Inc. in Monroeville, PA (since August 2010). He has contributed to numerous programs in ground, undersea, and maritime surveillance for security and defense applications. His research interests include detection and estimation theory, multi-target tracking, data fusion, and stochastic control.

Dr. Coraluppi served as general cochair for the 9th ISIF/IEEE International Conference on Information Fusion (FUSION) in Florence, Italy in 2006 (with Peter Willett) and program chair for the NATO Workshop on Data Fusion and Anomaly Detection for Maritime Situational Awareness in La Spezia, Italy in 2009. He has served as member of the NATO Research and Technology Organization (RTO) Sensors and Electronics Technology (SET) Panel, associate editor for the *IEEE Journal of Oceanic Engineering* and the *ISIF Journal of Advances in Information Fusion (JAIF)*, and Lecturer for the NATO RTO, the Italian Naval Academy, and the University of Pisa. He was the 2010 ISIF President.

Currently, Dr. Coraluppi is associate editor for Target Tracking and Multisensor Systems for the *IEEE Transactions on Aerospace and Electronic Systems*, area editor for Tracking for the ISIF *JAIF*, Member of the Board of Directors of ISIF, and ISIF Vice President of Working Groups. He is a member of the ISIF Multistatic Tracking Working Group and the IEEE AESS Target Tracking Systems Panel. He is a Senior Member of the IEEE.

Craig Carthel received B.S. degrees in physics and mathematics in 1988, an M.S. in mathematics in 1992, and a Ph.D. in mathematics in 1995, all from the University of Houston, where he did research in numerical analysis and optimization theory.

He is a principal scientist at Compunetix Inc. in Monroeville, PA. From 1995 to 1997, he worked at the Institute for Industrial Mathematics at Johannes Kepler University, in Linz, Austria on parameter identification and inverse problems. From 1998 to 2002, he was a senior mathematician at ALPHATECH Inc. in Burlington, MA, where he worked on image processing, multisensor data fusion and ground target tracking. From 2002 to 2010, he was a senior scientist in the Applied Research Department at the NATO Undersea Research Centre in La Spezia, Italy, where he worked on military operations research, simulation, optimization, and data fusion problems associated with maritime environments. In 2006, he served as the technical program chair for the 9th International Conference on Information Fusion.