Temporal Decomposition for Online Multisensor-Multitarget Tracking

Jason L. Pacheco
Department of Computer Science
Brown University
Providence RI, 02912
Email: pachecoj@cs.brown.edu

Meinolf Sellmann
Department of Computer Science
Brown University
Providence RI, 02912
Email: sello@cs.brown.edu

***Abstract*—In this work we develop a novel temporal decomposition scheme for solving data association in multisensor-multitarget tracking problems. Given a set of noisy measurements from any number of sensors the data association problem is to determine which measurements originate from which targets. The problem is traditionally posed as an N-dimensional assignment problem, which is NP-hard for $N \geq 3$ dimensions. Our approach limits the dimensionality of the assignment problem and scales well to all problem instances. The formulation is implemented as an online algorithm and results are compared to existing algorithms.**

I. Introduction

At the heart of multitarget tracking is the data association problem. Given a set of measurements at each detection cycle the data association problem asks us to associate the measurements with the tracks they originate from. This problem is typically formulated as an N-dimensional version of the linear assignment problem known as the multidimensional assignment problem (MAP), which is NP-hard for $N \geq 3$.

In addition to the NP-hard nature of the problem there are also many other factors prohibiting good results. False detections make it difficult to determine which measurements emanate from tracks and they drastically increase the number of variables in the MAP needed to be solved. In addition to false detections sensors generally have a probability of detection $P_D < 1$ causing missed detections to become a concern. Finally, the existence of measurement noise requires the use of an estimator, such as the Kalman filter, to estimate latent state variables.

Multitarget tracking algorithms largely began with the pioneering work of Morefield. In [3] Morefield formulated the data association problem as linear integer set cover and set packing problems. Poore extended Morefield's work in [4] and formulated the data association problem as a MAP. Existing techniques for solving the data association problem largely make use of this MAP formulation.

Current algorithms typically fall into two categories, sequential and deferred logic. Two of the most popular sequential algorithms are the Kalman filter with nearest neighbor (KFNN) and the joint probabilistic data association (JPDA). Recently an extension of JPDA known as Interactive Multiple Model JPDA (IMMJPDA) has also become popular. In general, sequential algorithms can be thought of as online algorithms.

Deferred logic techniques can be thought of as offline algorithms which solve the data association problem for some set of detection periods. Batch processing is a fully offline algorithm which considers all track hypothesis over all times and chooses the most likely set of hypotheses. Multiple hypothesis tracking, on the other hand, was developed by Reid [5] and solves data association for a subset of detection periods over time.

We begin in section II with a brief derivation of the multidimensional assignment formulation introduced in [4]. In section III we discuss the spatial and temporal decomposition schemes used in our algorithm. The temporal decomposition presented in this section is the main contribution of this work. In section IV we give an introduction to the algorithm along with a formalization of our temporal decomposition scheme. Section V contains the results of running our implementation on simulated data. We conclude in section VI.

II. Multidimensional Assignment Formulation

In this section we will briefly derive the multidimensional assignment formulation used to solve the data association problem. For a more in-depth discussion of the derivation see [4] [2]. We begin by introducing some notation.

A. Preliminary Notation

At each time t_k we receive a collection of noise contaminated measurements denoted $Z(k) = \{z_{i_k}^k\}_{i_k=1}^{M_k}$. Assuming all sensors produce the same types of measurements we do not need to concern ourselves with which sensors produced which measurements. The cumulative set of all measurements from time t_1 to t_N is denoted $Z^N = \{Z(1), \ldots, Z(N)\}$.

At each time t_k we must determine which measurements of $Z(k)$ are false detections (e.g. noise) and which measurements emanate from tracks. Of these track measurements, the data association problem then is to determine which measurements originated from which tracks. In other words, to find the proper association of measurements to tracks. This is generally posed as a maximum likelihood problem

$$\max_\gamma \left\{ \frac{P(\Gamma = \gamma | Z^N)}{P(\Gamma = \gamma^0 | Z^N)} \Big| \gamma \in \Gamma^* \right\} \tag{1}$$

where Z^N is the set of all measurements as defined above, Γ^* is the set of all track association hypotheses, Γ is a random variable of the set Γ^*, γ is a particular track association hypothesis, and γ^0 is a reference hypothesis where all measurements emanate from noise. $P(\Gamma = \gamma^0|Z^N)$ is a normalizing constant for the likelihood value.

To formalize the idea of a track hypothesis γ consider the set of indices for measurements at time k denoted $I(k) = \{i_k\}_{i_k=1}^{M_k}$. Let the cumulative set of all indices be $I^N = \{I(1), \ldots, I(N)\}$. Consider a track hypothesis γ as a partition of the index set I^N which induces a partition of the measurements Z^N as

$$Z_{\gamma_i} = \{\{z_{i_k}^k\}_{i_k \in \gamma_i}\}_{k=1}^N \tag{2}$$

$$Z_\gamma = \{Z_{\gamma_1}, \ldots, Z_{\gamma_{n(\gamma)}}\}. \tag{3}$$

Therefore $\gamma_i \in \gamma$ is simply the i^{th} track. Similarly, Z_{γ_i} are the measurements associated with the i^{th} track. Finally γ represents a full track association and Z_γ is the partition of measurements induced by the hypothesis γ. The goal of the data association problem is to find the most likely partition γ.

An implementation detail exists regarding missed detections. If a sensors' probability of detection is less than unity (i.e. $P_D < 1$) then a target may not be detected at a particular detection period. We ignore this case to simplify the formulation provided. For details regarding missed detections and how to account for them see [4].

B. The Formulation

For our MAP formulation we introduce the following set of boolean variables,

$$z_{i_1,\ldots,i_N} = \begin{cases} 1 & \text{if}(i_1, \ldots, i_N) \in \gamma \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

In this formulation each partition γ is represented by a boolean variable $z_{i_1,\ldots,i_N}$.

Let $P(\gamma_i|Z^N)$ denote the posterior probability of the measurements induced by the partition $\gamma_i \in \gamma$ conditioned on the set of measurements up to time t_N. Through some independence assumptions outlined in [4] and Baye's formula we have that

$$\frac{P(\Gamma = \gamma|Z^N)}{P(\Gamma = \gamma^0|Z^N)} \equiv \prod_{\gamma_i \in \gamma} \frac{P(\gamma_i|Z^N)}{P(\gamma_i^0|Z^N)} \tag{5}$$

where $P(\gamma_i^0|Z^N)$ is the likelihood that the measurements induced by γ_i are noise measurements. We define the cost of assigning a set of measurements to a track as the negative log of the posterior probability

$$c_{i_1,\ldots,i_N} = -\ln\left(\frac{P(\gamma_i|Z^N)}{P(\gamma_i^0|Z^N)}\right). \tag{6}$$

Where $(i_1, \ldots, i_N) = \gamma_i$ are the indices specified in the partition γ_i. We then have

$$-\ln\left(\frac{P(\gamma|Z^N)}{P(\gamma^0|Z^N)}\right) = \sum_{\gamma_i \in \gamma} c_{i_1,\ldots,i_N}. \tag{7}$$

Finally, from (7) we have the objective for assigning a single track. Extending this we can state the MAP as

$$\begin{aligned}
\text{Minimize} \quad & \sum_{i_1=1}^{M_1}\sum_{i_2=1}^{M_2}\cdots\sum_{i_N=1}^{M_N} c_{i_1,\ldots,i_N} z_{i_1,\ldots,i_N} \\
\text{Subject To} \quad & \sum_{i_2=1}^{M_2}\sum_{i_3=1}^{M_3}\cdots\sum_{i_N=1}^{M_N} z_{i_1,i_2,\ldots,i_N} \le 1 \qquad \forall i_1 \in I(1) \\
& \sum_{i_1=1}^{M_1}\sum_{i_3=1}^{M_3}\cdots\sum_{i_N=1}^{M_N} z_{i_1,i_2,\ldots,i_N} \le 1 \qquad \forall i_2 \in I(2) \\
& \vdots \\
& \sum_{i_1=1}^{M_1}\sum_{i_2=1}^{M_2}\cdots\sum_{i_{N-1}=1}^{M_{N-1}} z_{i_1,i_2,\ldots,i_N} \le 1 \qquad \forall i_N \in I(N) \\
& \sum_{i_1=1}^{M_1}\sum_{i_2=1}^{M_2}\cdots\sum_{i_N=1}^{M_N} z_{i_1,i_2,\ldots,i_N} = N_T \\
& z_{i_1,i_2,\ldots,i_N} \in \{0,1\} \qquad \forall i_1, \ldots, i_N.
\end{aligned} \tag{8}$$

where the number of tracks N_T is assumed known for our purposes. The number of tracks is also assumed to be constant throughout the problem, i.e. we do not consider the creation or deletion of tracks.

Another implementation detail regarding ignoring missed detections is evident in (8) and worth noting. We have replaced the constraints normally posed as equality constraints by inequality constraints. Where most implementations constrain every measurement to belong to *exactly* one track, we are stating that they must belong to *no more than* one track. The final equality constraint is added to constrain the number of hypothesis to the previously known number of tracks N_T to omit the obvious optimal solution of assigning zero tracks.

III. Problem Decomposition

Looking at the optimization problem in (8) we see that simply stating the MAP is NP-hard, since the number of variables $z_{i_1,\ldots,i_N}$ is exponential in the size of the problem. This is an inherent trait of the multidimensional assignment problem, and one of the reasons why decomposition schemes become important in making the problem tractable. There are two types of decomposition considered here, spatial and temporal.

A. Spatial Decomposition

Spatial decomposition is a method of breaking down the problem into sub-problems by imposing certain spatial constraints. Typical spatial decomposition makes use of two concepts "gating" and "clustering" which will be discussed in further detail in this section. The success of spatial decomposition is contingent upon the quality of an assumed motion model.

The model used in our approach is a typical discrete time linear dynamic system described by a vector difference equation with additive white Gaussian noise. The state transition and measurement matrices are time invariant. At each time t_{k+1} the state of a target can be described by the following recursive equation

$$x(k+1) = Fx(k) + v(k) \tag{9}$$

where $x(k+1)$ is the target state vector at time t_{k+1}, F is the time-invariant state transition matrix, and $v(k)$ is the model noise at time t_k assumed to be zero-mean white Gaussian noise with covariance Q. Similarly the measurement at time t_k is given by

$$z(k) = Hx(k) + w(k) \tag{10}$$

where $z(k)$ is the measurement vector at time t_k, H is the time invariant measurement matrix, and $w(k)$ is the measurement noise at time k also assumed zero mean white Gaussian with covariance R.

Equations (9) and (10) are in the form necessary for a Kalman filter. The details of the Kalman filter are beyond the scope of this paper, but can be found in [1].

The first step in spatial decomposition is gating. Given a set of measurements hypothesized to belong to a particular track from t_1 to t_{k-1} gating is the process of determining the possible measurements at time t_k which could be assigned to the track according to our motion model, within some number of standard deviations.

We let $\overline{x}(k)$ be the estimated state of the target at t_k as predicted by the Kalman filter. Also, let $\overline{P}(k)$ be the estimated covariance matrix at time t_k. Consider a particular measurement $z_i^k \in Z(k)$. We measure the covariance of the random variable $\nu = z_i^k - H\overline{x}(k)$ as

$$S = H\overline{P}(k)H^T + R \tag{11}$$

where R is our known measurement covariance. The measurement z_i^k is within an "η-sigma" validation region if the following inequality holds

$$(z_i^k - H\overline{x}(k))^T S^{-1}(z_i^k - H\overline{x}(k)) \leq \eta^2. \tag{12}$$

Let $\overline{z}(k) = H\overline{x}(k)$ be the estimated measurement at time t_k as predicted by the Kalman filter. We can then simplify (12) to

$$(z_i^k - \overline{z}(k))^T S^{-1}(z_i^k - \overline{z}(k)) \leq \eta^2. \tag{13}$$

Given a track hypothesis γ_i for measurements in Z^{k-1} we use the Kalman filter to predict the measurement $\overline{z}(k)$. We then iterate over all measurements in $Z(k)$ and determine if they are in the η-sigma validation region of the track. This completes the gating process for the hypothesized track.

Once all tracks are gated we can perform clustering. For each set of tracks that share a measurement in their gate we form a cluster containing each of those tracks. If at any time two clusters share a measurement we form a super cluster containing all of the tracks in each of the original clusters.

B. Temporal Decomposition

Temporal decomposition, like spatial decomposition, is a method of limiting the number of hypotheses that need to be considered, and thus limiting the number of variables in the MAP. Consider the set of detections Z^N for N detection periods.

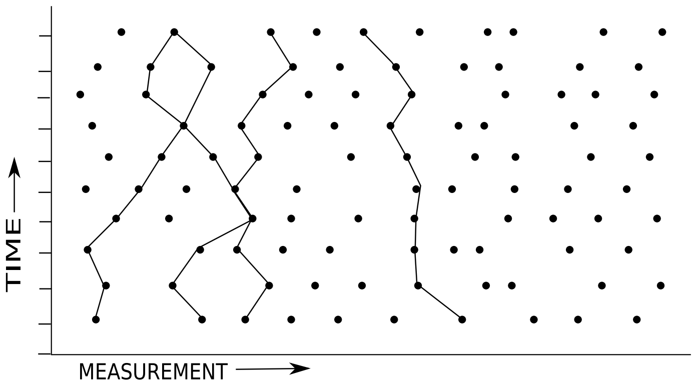

Fig. 1. Variables for batch processing

One method of generating variables for the multi-assignment problem is to use a batch process. Figure 1 shows an example of some of the hypothesis variables that would be generated. Ignoring gating effects the batch process would generate $O(N^M)$ variables, where M is the average number of detections per detection period.

An existing method of temporal decomposition is known colloquially as the "sliding window" approach. This approach generates a "window" of size ω_s detection periods. At each new detection period the window slides forward and solves a batch problem consisting of the last ω_s detection periods. Figure 2 shows an example when $\omega_s = 3$.

Using the sliding window approach breaks the dependency on previous detection periods. One method used to compensate for this is to remember the k-best hypotheses from previous detection cycles. Our method takes this approach a step further by decomposing the entire problem into a set of variables with dependency constraints enforced.

Using our approach we consider the entire data set Z^N. The problem is then decomposed into variables representing ω_s sequential measurements. For the case $\omega_s = 3$ the variable $z_{i_k,i_{k+1},i_{k+2}}$ represents measurements $i_k \in I(k)$, $i_{k+1} \in I(k+2)$, and $i_{k+2} \in I(k+2)$.

Dependency between variables is enforced using a continuity constraint. The constraint states that an ω_s variable, if assigned, must overlap an assigned variable by $\omega_s - 1$ measurements. More formally, for $\omega_s = 3$, we can only assign variable $z_{i_k,i_{k+1},i_{k+2}}$ if we assign some variable $z_{i_{k-1},i_k,i_{k+1}}$ for some $i_{k-1} \in I(k-1)$.

Using this method the number of variables generated is on the same order of magnitude as with the sliding window

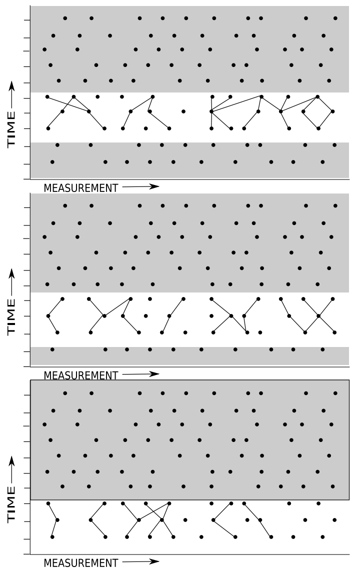

Fig. 2. Variables for MHT with sliding window of 3

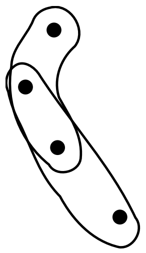

Fig. 3. Overlapping variable assignment

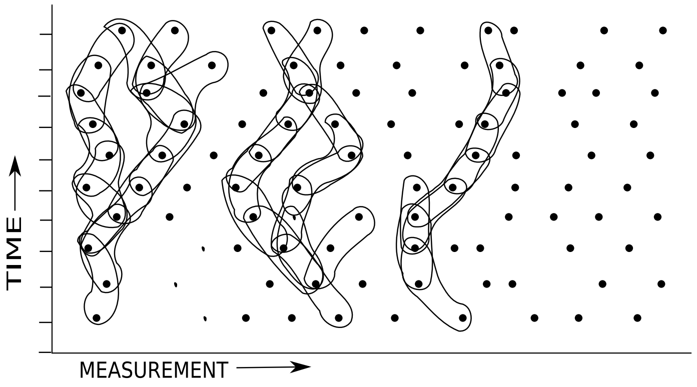

Fig. 4. Partial decomposition for $\omega_s = 3$

approach. For a data set Z^N with a variable size of ω_s we generate $O(M^{\omega_s+1})$ assuming $N > M$. However, for reasons that will be explained in section IV-C, we are much more likely to obtain an optimal solution than with the sliding window.

IV. The Algorithm

The tracker is implemented as an online algorithm, as data arrives it is processed in-situ. The algorithm scales well with problem size and the goal is a fully online real-time tracker. Batch processing is performed offline to resolve regions of interest. Currently there is no limit on the size and number of regions of interest that can be created so it is conceivable that tracker performance can fall below real-time in dense environments. However it would be trivial to impose such a limit. Figure 5 depicts a high level flowchart of the algorithm.

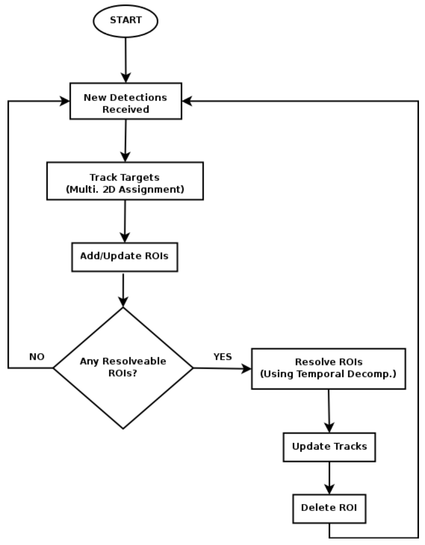

Fig. 5. Algorithm flowchart

In the following subsection we discuss how the algorithm creates regions of interest and makes use of them. We then formalize the temporal decomposition introduced in section III-B.

A. Unresolved Tracks and Regions of Interest

In target tracking the term *region of interest (ROI)* is typically used to refer to a set of measurements, which are spatially and temporally contiguous, to which we would like to pay particular attention. For our application we make use of the concept of *unresolved tracks* to further refine our use of ROIs.

Unresolved tracks are those which have come into contention and require further processing. Using the notion of clustering from section III-A we define tracks to be in contention with one another when they belong to the same cluster. Figure 6 shows two tracks crossing, a classic example of track contention.

At any detection period t_k if two or more tracks come into contention that were not previously in contention they are marked as unresolved. The tracker marks the time at which these tracks came into contention and a new ROI is formed. The tracker continues to process updates in the ROI using a polynomial time tracker which solves the two-dimensional assignment problem.

If at any point $t_{k'}$ another track enters the previously created ROI the new track is also marked as unresolved. The start time of the ROI remains at the original t_k and the new track is added to the list of unresolved tracks in that ROI.

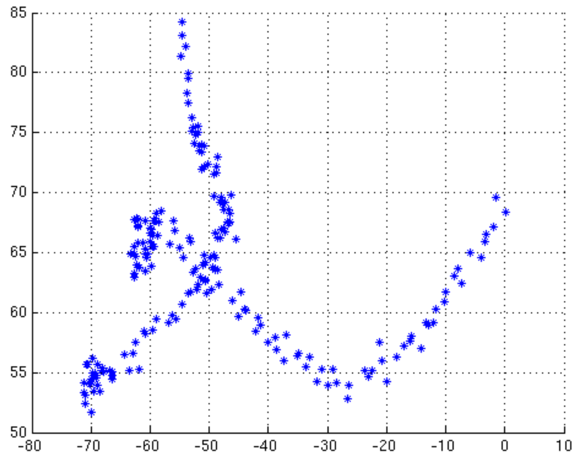

Fig. 6. Track crossing

The algorithm has the ability to maintain multiple ROIs and update them in an online fashion. The tracks in these ROIs will continue to be updated using the polynomial time tracker, but they remain marked as unresolved tracks.

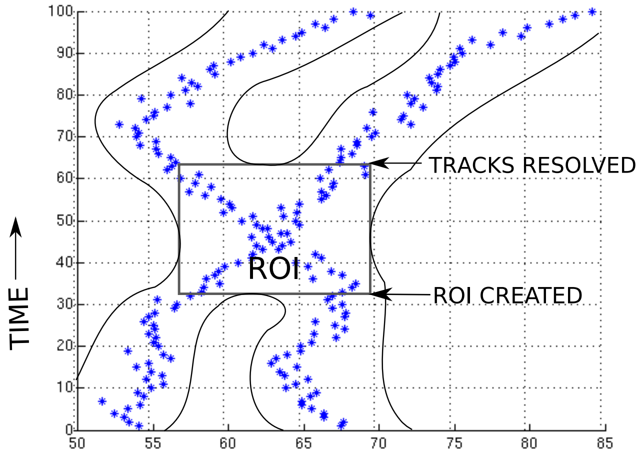

Fig. 7. Region of interest creation and track resolution

At some point $t_{k''}$ either the tracks will drift far enough apart such that they are no longer in contention, or the tracker will reach the end of the data scenario. In either case the tracker will resolve the ROI using the temporal decomposition from section III-B. At this point the tracks are updated, the ROI is destroyed, and the new clusters are computed. Figure 7 depicts these events on a simple two-track crossing.

B. Temporal Decomposition Formulation

When an ROI resolution is initiated the algorithm decomposes the sub-problem in the manner described in section III-B. An optimization problem is formulated and solved by a standard linear optimizer. In this case we are using CPLEX to solve the ROI sub-problem.

The algorithm decomposes the ROI into a set of ω_s-measurement variables. We will use $z_{i_k,\ldots,i_{k+\omega_s-1}}$ to denote a variable consisting of measurements from times $t_k, t_{k+1}, \ldots, t_{k+\omega_s-1}$ whose indices are $i_k \in I(k), i_{k+1} \in I(k+1), \ldots, i_{k+\omega_s-1} \in I(k+\omega_s-1)$.

Our objective in the decomposition is identical to that of the MAP formulation. Namely, we wish to minimize the negative log likelihood of the association hypothesis. We have,

$$\min \sum_{k=1}^{N-2} \sum_{i_k=1}^{M_k} \cdots \sum_{i_{k+\omega_s-1}=1}^{M_{\omega_s-1}} c_{i_k,\ldots,i_{k+\omega_s-1}} z_{i_k,\ldots,i_{k+\omega_s-1}} \tag{14}$$

where N is the number of detection cycles in the ROI. For notational convenience here we have assumed that the ROI begins at t_1 and ends at t_N.

To enforce a track structure of the variable assignments we constrain the assigned variables to overlap by exactly $\omega_s - 1$ measurements. This is called the continuity constraint and is implemented as,

$$\sum_{i_k} z_{i_k,\ldots,i_{k+\omega_s-1}} \geq \sum_{i_{k+\omega_s}} z_{i_{k+1},\ldots,i_{k+\omega_s}} \forall k. \tag{15}$$

We make the assumption that the number of tracks remains fixed throughout the ROI; no track additions or deletions are allowed. This is enforced using the following assignment constraint,

$$\sum_{\substack{i_k \\ \cdots \\ i_{\omega_s-1}}} z_{i_k,\ldots,i_{k+\omega_s-1}} = N_T \quad \forall k \tag{16}$$

where N_T is the number of tracks in the ROI.

An additional assignment constraint states that, for any two variables sharing a measurement, only one variable may be assigned. This is equivalent to all of the constraints in the original MAP formulation (8). The constraints can be condensed in the following form.

$$\sum_{i_j}^{M_j} \cdots \sum_{i_{j+k-1}}^{M_{j+k-1}} \sum_{i_{j+k+1}}^{M_{j+k+1}} \cdots \sum_{i_{j+\omega_s-1}}^{M_{j+\omega_s-1}} z_{i_j,\ldots,i_{j+\omega_s-1}} \leq 1 \quad \forall j, k, i_k \tag{17}$$

We also assume that the first two measurements of each track are known. That is, we know measurements i_1 and i_2 for each track. We use these measurements to determine a trajectory into the ROI and for calculating likelihoods. Let i_1^j and i_2^j be the first two measurements, assumed known, for track j. A prior constraint is enforced as,

$$\sum_{\substack{i_3 \\ \cdots \\ i_{\omega_s-1}}} z_{i_1^j, i_2^j, i_3, \ldots, i_{\omega_s-1}} = 1 \quad \forall j \in J \tag{18}$$

where J is our set of known tracks, and $|J| = N_T$. Again, for notational convenience, we have assumed that the ROI begins at t_1.

From (14),(15),(16), (17), and (18) we have the following optimization for the temporal decomposition,

$$
\begin{aligned}
&\text{Min} \quad \sum_{k=1}^{N-2} \sum_{i_k=1}^{M_k} \cdots \sum_{i_{k+\omega_s-1}=1}^{M_{\omega_s-1}} c_{i_k,\ldots,i_{k+\omega_s-1}} z_{i_k,\ldots,i_{k+\omega_s-1}} \\
&\text{Subject To} \\
&\sum_{i_k} z_{i_k,\ldots,i_{k+\omega_s-1}} \geq \sum_{i_{k+\omega_s}} z_{i_{k+1},\ldots,i_{k+\omega_s}} \forall k \\
&\sum_{\substack{i_k \\ \cdots \\ i_{\omega_s-1}}} z_{i_k,\ldots,i_{k+\omega_s-1}} = N_T \quad \forall k \\
&\sum_{i_j}^{M_j} \cdots \sum_{i_{j+k-1}}^{M_{j+k-1}} \sum_{i_{j+k+1}}^{M_{j+k+1}} \cdots \sum_{i_{j+\omega_s-1}}^{M_{j+\omega_s-1}} z_{i_j,\ldots,i_{j+\omega_s-1}} \leq 1 \quad \forall j,k,i_k \\
&\sum_{\substack{i_3 \\ \cdots \\ i_{\omega_s-1}}} z_{i_1^j,i_2^j,i_3,\ldots,i_{\omega_s-1}} = 1 \quad \forall j \in J \\
&z_{i_k,i_{k+1},i_{k+2}} \in \{0,1\} \quad \forall k.
\end{aligned}
\tag{19}
$$

To resolve the ROI we first solve this optimization. Knowing the measurements that each variable represents we then reconstruct the tracks based on the solution. The computationally intensive part is actually generating the variables. Once the variables are constructed the problem is formulated and solved in milliseconds.

C. Variable Analysis

We first introduce some notation for analysis purposes. As in Section II let Γ^* represent the set of all possible track associations. Let $\gamma \in \Gamma^*$ be a particular track association, and thus a particular feasible solution for an instance of MAP. Finally, let $\gamma_i \in \gamma$ be a particular track, and is therefore represented by a variable $z_{i_1,\ldots,i_N}$ in a MAP instance.

To formalize we let

$$V \triangleq \{\gamma_i : \gamma_i \in \gamma, \forall \gamma \in \Gamma^*\} \tag{20}$$

be the set of all possible hypotheses for each particular track. Therefore $|V|$ is the number number of possible tracks, and therefore the number of variables in the MAP instance. Observe that $|\Gamma^*|$ is the number of feasible solutions in the MAP instance, and will be referred to as the size of the solution space for the MAP instance.

Our goal is to minimize the number of variables $|V|$ in the MAP instance, but maximize the size of the solution space $|\Gamma^*|$.

Consider a straightforward batch computation for N detection cycles. For a batch process we have that $|V| = O(M^N)$ where M is the average number of detections per cycle. For simplicity we have ignored any gating effects. The size of the solution space in this case is also $|\Gamma^*| = O(M^N)$.

Now consider the sliding window approach discussed in section III-B. Let V_{ω_s} be the variables created for the each ω_s-cycle sub-problem. We have that $|V_{\omega_s}| = O(M^{\omega_s})$. Since we must solve $O(N)$ sub-problems we have that $|V| = N|V| = O(M^{\omega_s+1})$ for $N \geq M$.

Similarly let $\Gamma^*_{\omega_s}$ be the solution space for each ω_s-cycle MAP instance. We have that $|\Gamma^*_{\omega_s}| = O(M^{\omega_s})$. By similar reasoning we have that $|\Gamma^*| = O(M^{\omega_s+1})$ for $N \geq M$.

In the sliding window case we have decreased the number of variables $|V|$ by $N - \omega_s$ orders of magnitude. However, we have also decreased the solution space by the same amount. It is therefore likely that we have eliminated globally optimal solutions from the solution space Γ^*.

Finally consider our temporal decomposition approach. The number of variables required for solving N detection cycles is $|V| = O(M^{\omega_s+1})$, the same as in the sliding window above. The size of the solution space however is $|\Gamma^*| = O(M^N)$. We have minimized the number of variables by $N - \omega_s$ orders of magnitude, as in the sliding window approach, but we have maintained the size of the solution space in the batch problem. These results are summarized in the following table.

TABLE I
TEMPORAL DECOMPOSITION SCHEME ANALYSIS RESULTS

Algorithm	Num. Vars $\|V\|$	Size Sol. Spc. $\|\Gamma^*\|$
Batch	$O(M^N)$	$O(M^N)$
Sliding Window	$O(M^{\omega_s+1})$	$O(M^{\omega_s+1})$
Our approach	$O(M^{\omega_s+1})$	$O(M^N)$

V. RESULTS

A. Test Description

Our implementation consists of 2,000 lines of C++ code and 100 lines of MATLAB code. We chose CPLEX as the linear solver for optimizing the assignment problems. Finally, a simulator was written to generate test scenarios. The simulator consists of approximately 1,000 lines of MATLAB code and can generate data with varying parameters.

We focus our results on data sets with 100 seconds worth of data and 10 targets in a 200x200 volume. The noise parameter is specified in noise measurements per volume. A noise measurement of 8/vol, for instance, would average 8 noise measurements per detection cycle. Noise measurements are distributed uniformly at random throughout the volume.

A set of 60 random test scenarios were generated for noise measurements of 8/vol, 16/vol, and 24/vol. Figures 8, 9, and 10 show examples of these scenarios for each of the noise parameters. Tracks are displayed in green while noise measurements are in blue.

Each scenario contains 100 seconds worth of data with 10 tracks. We compare the results of three algorithms, the baseline two-dimensional assignment tracker, the offline batch tracker using our temporal decomposition, and finally the online fully adaptive tracker with ROI decomposition. The test machine consists of two 2.01GHz Athlon 64 processors with 1.9GB of RAM running Linux version 2.6.18 SMP.

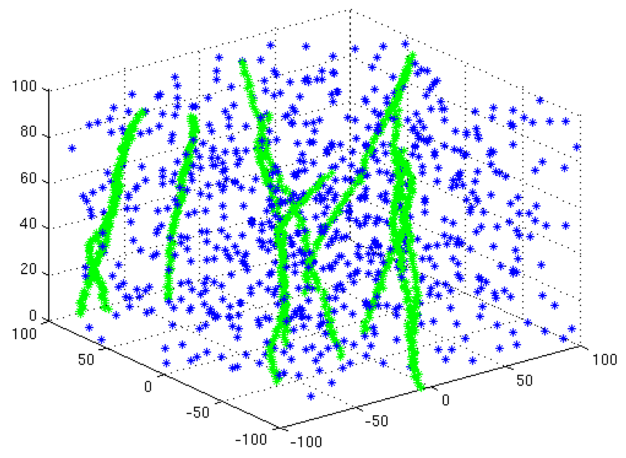
Fig. 8. Sample data set for noise 8/vol

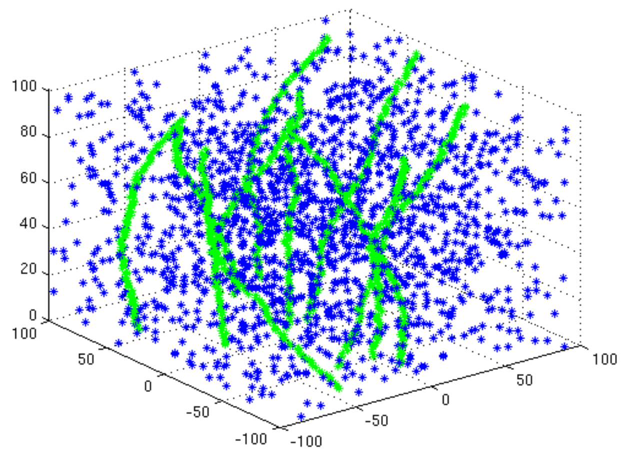
Fig. 9. Sample data set for noise 16/vol

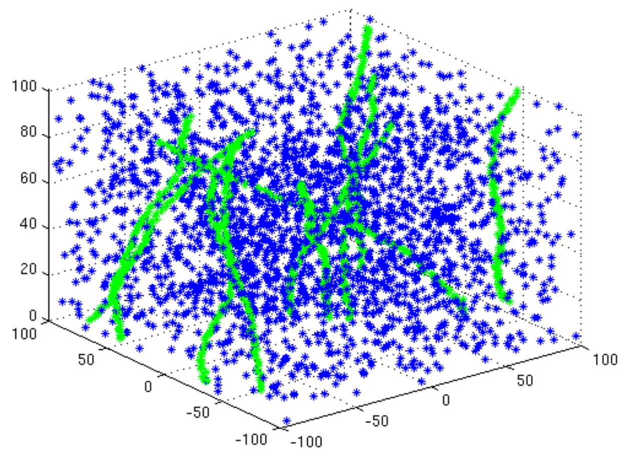
Fig. 10. Sample data set for noise 24/vol

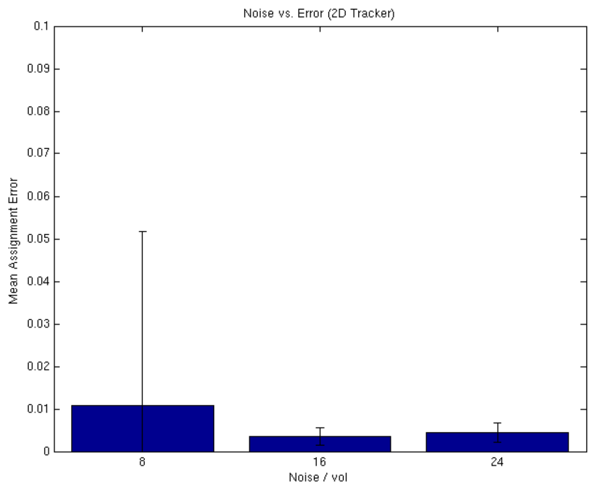

Fig. 11. 2D Tracker: Mean Assignment Error

B. *Tracker Error*

The error metric we chose was mean assignment error (MAE), which is the average number of erroneous measurement assignments. More formally, at some time t_k if the measurement assigned by the tracker does not exactly match the ground truth measurement, then it is considered an assignment error.

Tracking algorithms are typically measured in a distance-based metric such as mean squared error (MSE). However, because missed detections are not considered, in observing the results we found that the algorithm only fails when it switches tracks. In addition, the tracker rarely follows noise measurements for any significant period of time. We wish to focus on failed track crossings and feel that MAE is a better measurement of this. In retrospect there may be better error measurements to directly measure this behavior.

There are a total of 10 tracks in each scenario spanning 100 seconds. Therefore, the total number of assignments is 1000. The MAE is normalized by this number and reported as a percentage of the total number of assignments in the scenario. Therefore an MAE of 0.1 means that 10% of assignments were erroneous.

1) Two-Dimensional Assignment Tracker: Figure 11 shows the mean assignment error for the baseline two-dimensional assignment tracker. On the horizontal axis we have the noise parameter for the scenario data set. The vertical axis shows the mean percentage of incorrect assignments.

The two-dimensional tracker showed low error and variance in scenarios with noise parameter 16/vol and 24/vol. The results for scenarios with noise 8/vol was less favorable due to a failed track crossing. The output of the tracker is shown in Figure 19. Failures in track crossings are analyzed in Section

V-C.

2) Batch Tracker (Without ROI): This section focuses on our temporal decomposition scheme applied to an offline batch tracker. The entire scenario is decomposed and solved offline. Figure 12 shows error results for this tracker.

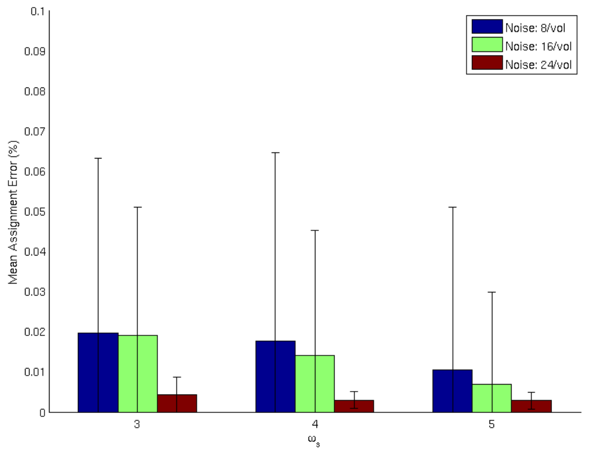

Fig. 12. Batch Tracker Error

There is a downward trend in error with respect to variable size ω_s as expected. The error measurements for $\omega_s = 5$ are comparable to those of the two-dimensional tracker and fall below 1% assignment error in all scenarios.

The variance is quite high for scenarios with 8/vol and 16/vol noise. The standard deviation of noise for these scenarios is on the order of several percent. This is an open issue and is the topic of future work.

3) Fully Adaptive Tracker (With ROI): The fully adaptive tracker is the online version of our algorithm. This tracker uses the same temporal decomposition as the batch tracker in the preceding section. Additionally, we use the regions of interest described in Section IV-A.

Figure 13 shows the error measurements for our fully adaptive tracker. As with the batch tracker we see similar variance issues with scenarios of noise 8/vol and 16/vol. This will be addressed in future work.

Here we notice a slight instability in the tracking algorithm for scenarios with noise measurements of 8/vol. With $\omega_s = 3$ we see an MAE that is less than the MAE for $\omega_s = 4$ and $\omega_s = 5$ for the same scenarios.

4) Algorithm Comparison: We compare errors between the three algorithms. We set our variable length at $\omega_s = 5$ because tracker error is lowest for this setting. Figure 14 shows a comparison for the three algorithms over each of the noise settings.

For a noise parameter of 8/vol and 24/vol we see similar results. In each case the variance is comparable between all of the algorithms. Also the batch tracker exhibits the minimum error for these scenarios, with the fully adaptive tracker showing a slightly higher error. For a noise parameter

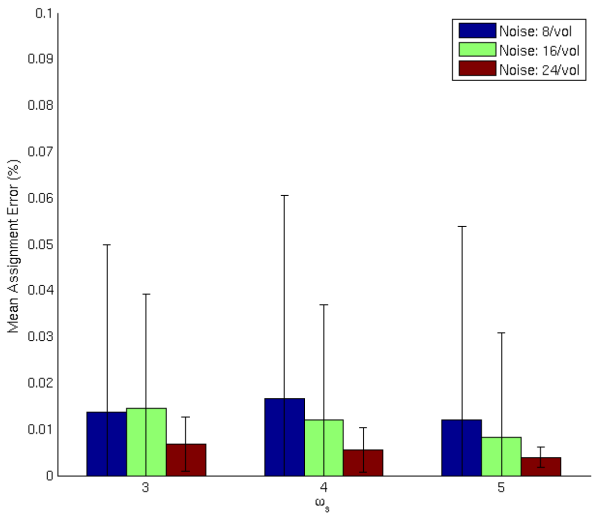

Fig. 13. Fully Adaptive Tracker Error

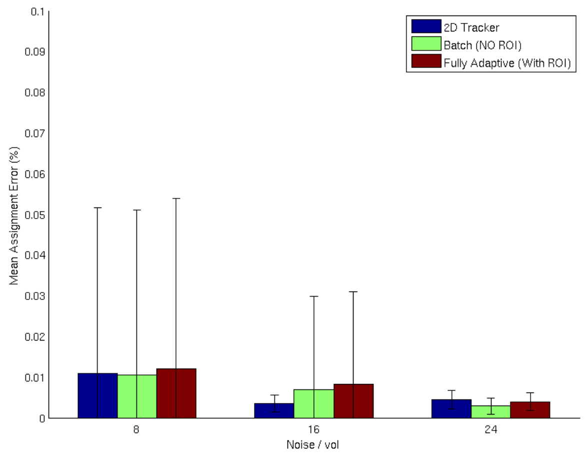

Fig. 14. Algorithm Comparison ($\omega_s = 5$)

of 16/vol we see that the two-dimensional tracker performed the best, both in error and variance.

Next we analyze the effect of variable size on tracker error. Figure 15 shows an error comparison for scenarios of noise 8/vol between the batch tracker and the fully adaptive tracker. Figures 16 and 17 show similar results for noise parameters of 16/vol and 24/vol, respectively.

In general we see that increasing the variable size decreases the tracker error. The trade-off, of course, is increased computation time. Surprisingly, the fully adaptive tracker exhibits error below the batch tracker in several instances. Though, for $\omega_s = 5$ the batch tracker consistently shows lower error.

Finally, we compare our algorithm to an implementation of the sliding window. Figure 18 shows the error comparison for $\omega_s = 3$. Where, ω_s is also the size of the sliding window. In

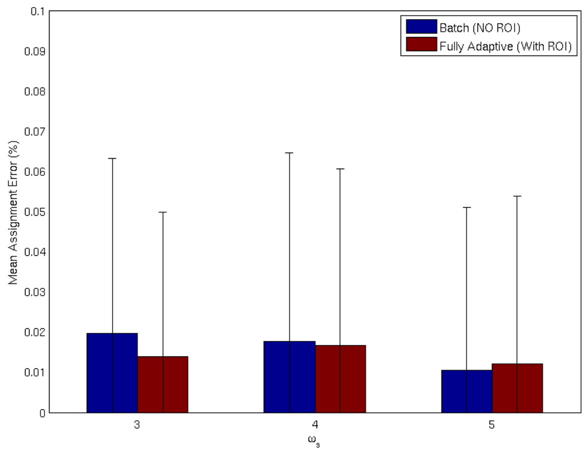

Fig. 15. Mean assignment error, noise 8/vol

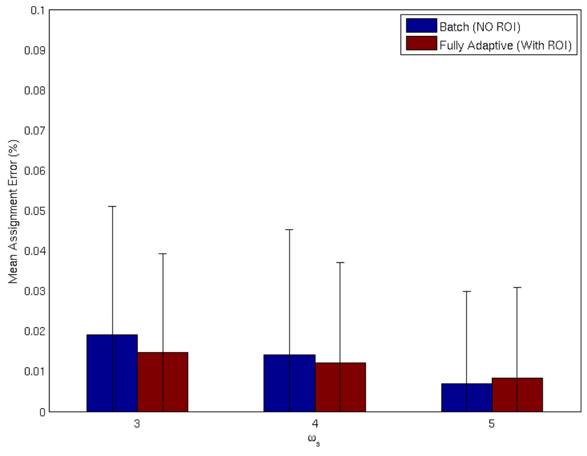

Fig. 16. Mean assignment error, noise 16/vol

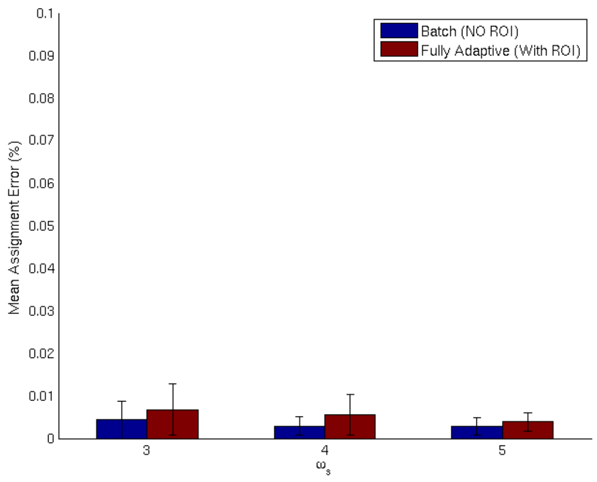

Fig. 17. Mean assignment error, noise 24/vol

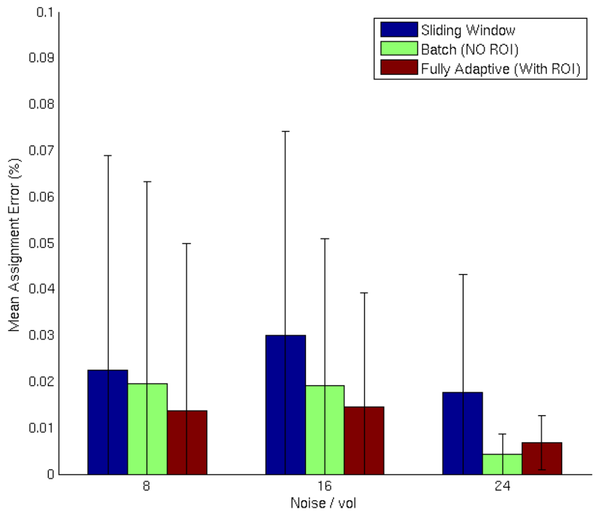

Fig. 18. Comparison to sliding window for $\omega_s = 3$

all cases the sliding window algorithm performed worse than our algorithm.

C. Track Crossing Analysis

It should be noted that the two-dimensional tracker performed unreasonably well in our results. The reason for this, we believe, lies in the quality of the simulation data. The noise is distributed uniformly, which would not be the case in more realistic data.

Another reason for this behavior lies in a property of the Kalman filter as a likelihood estimator. As a linear estimator the Kalman filter assigns higher likelihood to objects moving in straight lines. Because the two-dimensional tracker relies on the Kalman filter for longer sequences of data, the tracker also expects things to move in a linear fashion. As a result, when two tracks meet, the two-dimensional tracker expects them to cross.

In most of the scenarios generated this is exactly the case, so the two-dimensional tracker is correct in these instances. However, when two tracks meet but do not cross, as in figure 19, the two-dimensional tracker fails consistently.

To illustrate this point Figure 20 shows the output of the two-dimensional tracker on such a crossing. Figure 21 shows the output of the batch tracker with a variable size of $\omega_s = 3$. Here, the batch tracker failed to correctly solve the track crossing. However, increasing the variables size to $\omega_s = 5$ allows us to receive more informative likelihood estimates from the Kalman filter. Figure 22 shows the result of the batch tracker for $\omega_s = 5$.

While increasing the variable size allows us to correctly estimate track crossings more reliably, there is also the trade-off that track "kissing" is tracked less reliably. A track kissing

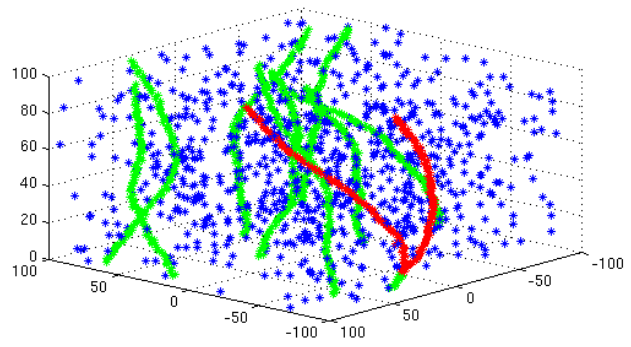

Fig. 19. Track switch

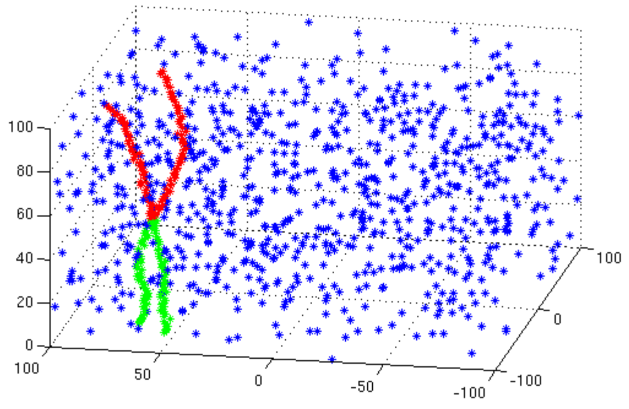

Fig. 21. Two-track Crossing: Batch tracker $\omega_s = 3$

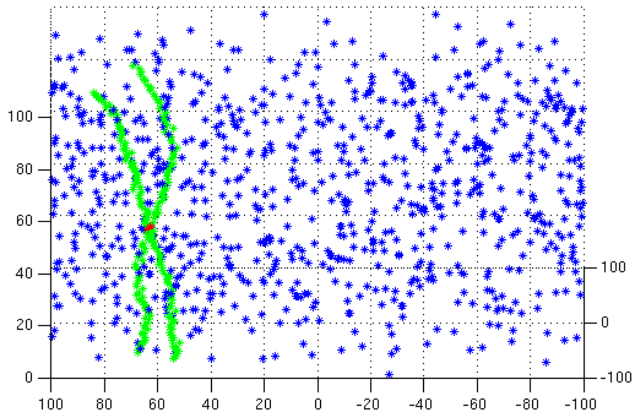

Fig. 20. Two-track Crossing: Two-dimensional Assignment Tracker

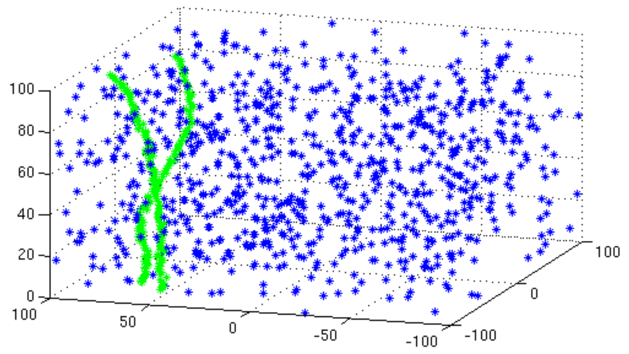

Fig. 22. Two-track Crossing: Batch tracker $\omega_s = 5$

is when two or more tracks come into contention and then move away from each other without crossing.

Figure 19 shows such an instance where the two-dimensional tracker fails to track after these two tracks touch one another. Figures 23 and 24 show results of the batch tracker on the same scenario for variable sizes $\omega_s = 3$ and $\omega_s = 5$, respectively.

The result of this analysis is that better likelihood estimates are needed. The Kalman filter makes the implicit assumption that objects move linearly. When this is the case performance is generally good. When objects do not move linearly, however, errors are encountered. Perhaps one approach would be to incorporate learning techniques to estimate the optimal variable size in-situ. As tracks begin maneuvers the variable size would be decreased automatically, to allow for more non-linear motion dynamics. Another approach would be to incorporate an extended Kalman filter or particle filter for calculating likelihood estimates.

D. *Computation Time*

The computation time of the two-dimensional tracker is quadratic. The time for our batch tracker is exponential in the variable size, namely $O(M^{\omega_s+1})$, where M is the average number of measurements per detection period. In our case M = noise/vol + # of tracks. The computation time of the fully adaptive tracker is dependent on the number of ROIs and the duration of ROIs, but it is equal to that of the batch tracker in the worst case.

Figures 25, 26, and 27 show computation time for each of the variable sizes tested. Also in each graph is the computation time for the two-dimensional tracker, as a reference. The horizontal line at 100 indicates the real-time threshold. Anything

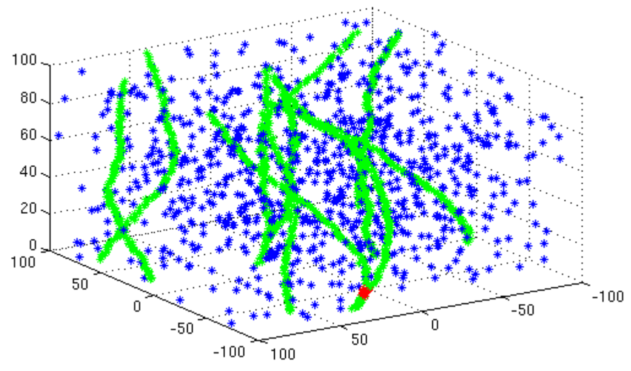

Fig. 23. Two-track Kiss: Batch tracker $\omega_s = 3$

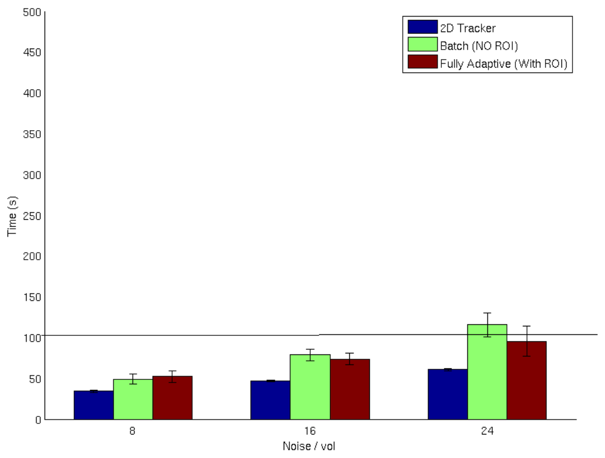

Fig. 25. Computation time for $\omega_s = 3$

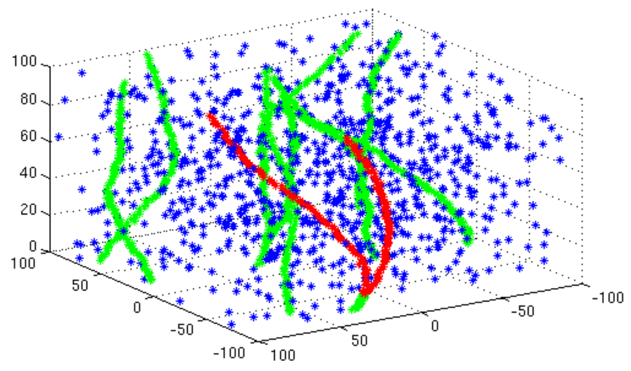

Fig. 24. Two-track Kiss: Batch tracker $\omega_s = 5$

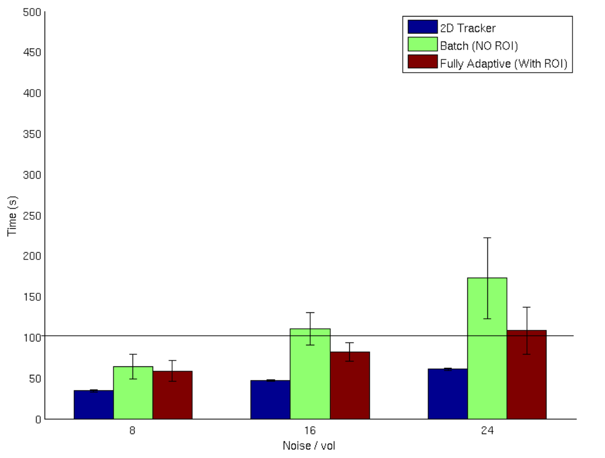

Fig. 26. Computation time for $\omega_s = 4$

below this line operates in real-time if the detection period of the data is considered to be 1 second.

For the batch tracker there is a linear upward trend with respect to scenario noise. The same also holds for the fully adaptive tracker, but is dependent on the number of ROIs created.

The batch tracker also shows an exponential trend with respect to variable size, as expected. The fully adaptive tracker shows a much more gradual slope, but is still exponential in the worst case. However, the fully adaptive tracker makes $\omega_s = 4$ a plausible option for real-time scenarios in dense environments. Additionally $\omega_s = 5$ is possible for less noisy scenarios. It is conceivable that the algorithm can be made more efficient to make use of $\omega_s = 5$ for real-time computation in noisy scenarios. This will be a focus of future work.

VI. Conclusion

We have presented a temporal decomposition scheme for multisensor multitarget tracking applications. The number of variables created is exponential in the variable size, rather than in the total scenario time as in a traditional batch computation. However, the size of the feasible solution space is on the same order of magnitude as a traditional batch computation, so optimal solutions are not likely to be pruned from the feasible solution space.

The algorithm was implemented with and without region of interest decomposition. Results were compared between the two versions of the algorithm and a baseline tracker which solves the two-dimensional assignment problem. Additionally a sliding window algorithm was implemented and errors were compared for $\omega_s = 3$. The sliding window failed to track several scenarios for $\omega_s = 4$ and $\omega_s = 5$ so these results were not compared.

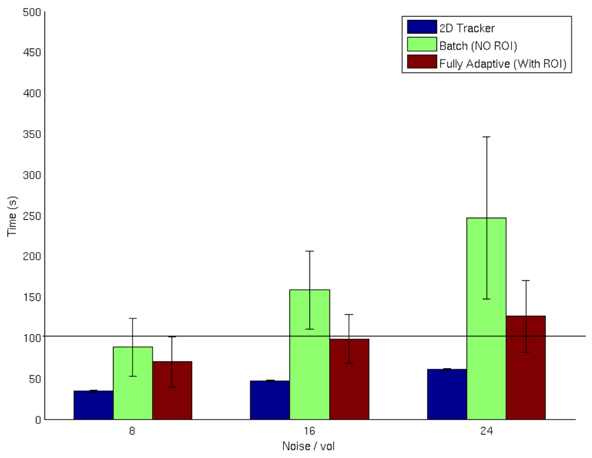

Fig. 27. Computation time for $\omega_s = 5$

We believe that the methods introduced here show promising results, but more realistic data is needed for testing. Noise in our test data was distributed uniformly at random, which does not simulate realistic data. In more realistic data noise would be normally distributed around each target, and the two-dimensional tracker would exhibit a tendency to lose track. This behavior was not observed on our synthetic test data.

Putting the quality of the test data aside we draw several conclusions from the experiments conducted. Regarding variable size we conclude that a variable size of $\omega_s = 5$ exhibits the lowest error among our experiments. We expect $\omega_s > 5$ to improve accuracy at the cost of computation time.

Computation time is acceptable for $\omega_s = 5$ in less noisy environments. With further optimizations and additional computing resources, we believe that it will be possible to lower the computation time for $\omega_s = 5$ below the real-time threshold in noisy scenarios. Additionally it would be possible to implement learning techniques which allow the tracker to gracefully degrade the variable size to meet real-time limits in noisy scenarios.

Regarding the comparison between the fully adaptive tracker and the batch tracker, we found that the difference in error is negligible. The fully adaptive tracker exhibited slightly higher error measurements in most cases, but the speedup in computation time outweighs the incurred error penalty.

Finally, the obvious drawback of relying on a linear estimator for likelihood values is an open issue. As explained in section V-C the Kalman filter makes the implicit assumption that objects move linearly. The effect can be minimized by decreasing the variable size, but the trade-off is made between correctly solving track crossings and track kissings. Future work will involve replacing the Kalman filter with an Extended Kalman filter or particle filter for better nonlinear likelihood estimation.

ACKNOWLEDGMENTS

I would like to thank my adviser, Prof. Meinolf Sellmann, for his research guidance and personal wisdom. He has made my time at Brown an unforgettable experience that will live with me forever.

I would also like to thank the cast and crew at the Naval Undersea Warfare Center (NUWC) Division Newport. Specifically I would like to thank Dr. Michael Benjamin for opening the pathway for me to further my education here at Brown. Without knowing it you have been the catalyst for change in my life.

I would also like to thank the tracker community at NUWC. Among them, Dr. Tod Luginbuhl, who always made time for my questions and provided remote guidance when I needed it. I would also like to thank Tod for the MATLAB code necessary to get me started. Additionally, I would like to thank Dr. Phillip Ainsleigh and Dr. Christian Hempel, also at NUWC, for responding to my inquiries and helping out when help was needed.

Finally, I would like to thank the Long Term Training review committee at NUWC for making this opportunity available to me. I hope that I've fulfilled the potential you saw in me.

REFERENCES

[1] Y. Bar-Shalom, X. Rong Li, T. Kirubarajan, *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*, Wiley Interscience, 2001

[2] S. Deb, Y. Bar-Shalom, K. Pattipati, M. Yeddanapudi, *A Generalized S-D Assignment Algorithm for Multisensor-Multitarget State Estimation*, IEEE Trans. Automat. Contr., vol. AC-33, No. 2, pp. 523-538.

[3] C. L. Morefield, *Application of 0-1 Integer Programming to Multitarget Tracking Problems*, IEEE Trans. Automat. Contr., vol. AC-22, pp. 302-311, June 1977.

[4] A. B. Poore, *Multidimensional Assignments and Multitarget Tracking*, DIMACS 199, pp. 169-196.

[5] D. B. Reid, *An Algorithm for Tracking Multiple Targets*, IEEE Trans. Automat. Contr., AC-24 (1979), no. 6, pp. 843-854.