Beyond MAP estimation with the track-oriented multiple hypothesis tracker

Andrew Frank, Padhraic Smyth, Member, IEEE, and Alexander Ihler, Member, IEEE

Abstract—The track-oriented multiple hypothesis tracker (TOMHT) is a popular algorithm for tracking multiple targets in a cluttered environment. In tracking parlance it is known as a multi-scan, maximum a posteriori (MAP) estimator – multi-scan because it enumerates possible data associations jointly over several scans, and MAP because it seeks the most likely data association conditioned on the observations. This paper extends the TOMHT, building on its internal representation to support probabilistic queries other than MAP estimation. Specifically, by summing over the TOMHT’s pruned space of data association hypotheses one can compute marginal probabilities of individual tracks. Since this summation is generally intractable, any practical implementation must replace it with an approximation. We introduce a factor graph representation of the TOMHT’s data association posterior and use variational message-passing to approximate track marginals. In an empirical evaluation, we show that marginal estimates computed through message-passing compare favorably to those computed through explicit summation over the \(k\)-best hypotheses, especially as the number of possible hypotheses increases. We also show that track marginals enable parameter estimation in the TOMHT via a natural extension of the expectation maximization algorithm used in single-target tracking. In our experiments, online EM updates using approximate marginals significantly increased tracker robustness to poor initial parameter specification.

Index Terms—Radar tracking, belief propagation, expectation-maximization, parameter estimation

I. INTRODUCTION

Multi-target tracking is a core component of many real-world sensing systems in applications including air defense, autonomous navigation, video surveillance, and robotics. Due to imperfect or low-information sensors, many such systems must handle an uncertain correspondence between sensor observations and real-world targets. Resolving this uncertainty is known as the data association problem, and it is the fundamental reason why optimal estimators for multi-target state estimation are generally intractable [1].

Despite its intractability, the problem’s practical importance has motivated numerous algorithms based on simplifying assumptions, approximations, and computational shortcuts. The track-oriented multiple hypothesis tracker (TOMHT) is one such algorithm. The TOMHT is generally considered to be among the most effective approaches for tracking in cluttered environments, given medium to high computational resources [2]. Broadly speaking, the TOMHT works by implicitly representing all possible joint data associations within a temporal sliding window, postponing hard decisions until observations fall behind the window’s trailing edge. At each step the TOMHT reports the most likely data association for all observations processed up to that point, placing it in the category of MAP-based algorithms for data association.

Although the TOMHT makes hard data association decisions outside the sliding window, its internal data structures define a full posterior distribution over the space of possible associations within the window. The MAP estimate typically computed by the TOMHT is only one of several possible summaries of this posterior. The distribution’s entropy, for instance, conveys a measure of uncertainty that could be used to dynamically adjust the length of the sliding window. Marginal probabilities of individual tracks could be useful as part of a real-time display for the tracker operator or, as we explore in the sequel, as a component of an expectation maximization (EM) algorithm for estimating system parameters.

Here we focus on the latter possibility: computation of marginal probabilities within the pruned data association space of the TOMHT. Even in the pruned data association space, marginalization is intractable. To this end we first identify two families of approximate estimators for the track marginals. The first technique follows a well-known approach based on exact computation.

In this paper we use the word \textit{track} to mean a sequence of observations generated by a single target. See the discussion at the top of page 3 for more on this terminology.
of the $k$-best data association hypotheses. The second uses a novel application of variational message-passing algorithms for marginalization. We conduct an empirical comparison of these estimators in terms of their accuracy on simulated sensor data. Finally, we show how track marginals enable unsupervised system identification via a multi-target generalization of the standard, single-target EM algorithm.

II. MULTI-TARGET TRACKING

A. Target, sensor, and environment models

Targets are modeled as points $x$ in a state space $\mathcal{X} \subset \mathbb{R}^{d_x}$. The semantics of $\mathcal{X}$ will vary depending on the particular tracking application, but it typically includes components corresponding to the target’s kinematic state such as position, velocity, and acceleration. Target states evolve in discrete time according to a first-order Markov dynamics model, $f_d(x_{k+1}^n | x_k^n)$, which specifies a probability distribution over a target’s next state given its current state.

At each time step $k$ a target may be detected by the sensor, yielding an observation $z$ in the observation space $\mathcal{Z} \subset \mathbb{R}^{d_z}$. Observations are distributed according to the observation model, $f_o(z_k^k | x_k^k)$. In general, the observation domain $\mathcal{Z}$ need not be the same as the target state domain. For example, when the state vectors represent both position and velocity the sensor may only observe its position.

The choice of an appropriate dynamics model is application-dependent, but due to its analytic tractability the linear Gaussian model is a common choice:

$$f_d(x_{k+1}^n | x_k^n) = \mathcal{N}(Ax_k^n, Q),$$
$$f_o(z_k^k | x_k^k) = \mathcal{N}(Hx_k^k, R),$$

where $A$ is a $d_x \times d_x$ transition matrix, $H$ is the $d_z \times d_x$ observation matrix, and $Q$ and $R$ are the process noise and observation noise covariance matrices, respectively. Use of this model results in the well-known Kalman filtering and smoothing recursions for single-target tracking, which is an important subroutine in the application-dependent, but due to its analytic tractability Markov filtering and smoothing recursions for single-target tracking, which is an important subroutine in the

In this paper we make standard assumptions regarding missed detections and false alarms (clutter), as follows. At each time step $k$, a target is detected with probability $p_D$. The sensor also generates $n_\phi$ clutter observations not corresponding to any target, where $n_\phi$ is Poisson distributed with rate parameter $\lambda_\phi$.

The number of targets in the surveillance region can change over time. At each time step $k$ new targets may enter the surveillance region and some or all of the preexisting targets may leave; we call these events track births and track deaths, respectively. At a given time, we model the number of track births as a Poisson random variable with parameter $\lambda_b$ and assume that each track has probability $p_b$ of dying. For simplicity we assume that the locations of track births and deaths are distributed uniformly over $\mathcal{X}$.

B. The data association problem

The difficulty of multi-target tracking stems from the lack of an observed correspondence between observations and targets. A common strategy is to take a data augmentation approach, introducing auxiliary variables that represent the (unobserved) sources of the observations. In this section we introduce these auxiliary variables and formulate the data association problem as an optimization over their posterior.

At each time step, the tracker receives as input all newly generated observations – both actual detections and clutter – grouped together into a single set called a scan. The scene in Figure 1, for example, shows three scans: $\{z_{1,1}^1, z_{1,2}^1\}$, $\{z_{2,1}^2\}$, and $\{z_{3,1}^3, z_{3,2}^3\}$, where we use the notation $z_{k,j}^{k,j}$ to represent the $j$th observation in scan $k$. Note that while the first index (scan number) conveys a meaningful temporal ordering of the observations, the second (within-scan) index is arbitrary – it does not contain any information regarding the identity of the target that generated that observation. We use the notation $z_{k}^k$ to represent the scan at time $k$, and $z$ for the union of all scans up to the present time.

A data association hypothesis is a partitioning of $z$ into false alarms and tracks: sets of observations corresponding to individual targets. A track is simply a set of index pairs $(k, j)$ from consecutive scans, each corresponding to an observation $z_{k,j}$. For example, $\{(1, 2), (2, 1), (3, 2)\}$, $\{(3, 1)\}$, and $\{(2, 1), (3, 1)\}$ are three possible tracks in Figure 1. The second index of each pair is restricted to the range $0 \leq j \leq m_k$, where $m_k$ is the number of observations in scan $k$. The resulting $(k, 0)$ pairs refer to pseudo-observations we introduce to represent missed detections.

Fig. 1. A small example scenario with three scans of data. Observations $z_{1,1}^1$ and $z_{3,2}^3$ are false alarms, but when they are processed by the tracker there are no labels identifying them as such.
A word on terminology: although the process of tracking is generally understood as the estimation of state sequences, the term track is used here to refer to a sequence of observations. This usage may seem unusual at first, but it is standard in the context of the TOMHT [4], [5]. Of course, with linear Gaussian observation and dynamics functions it is trivial to compute the filtered state sequence (e.g., estimated target positions) generated every observation in a complete track indicator vector $\tau$.

The modeling assumptions made thus far result in the posterior shown in Equation 3, where $x^u = [x^{u,1}, \ldots, x^{u,T_u}]$ is the vector of target states corresponding to track $T_u$ and $h(\tau)$ is a binary function that assigns invalid hypotheses zero probability [4]. The key feature of this posterior is the way it decomposes as a product over possible tracks. The logarithm of the terms inside the square brackets, which we will denote $s_u$, is called the track score. It can be thought of as a myopic measure of how likely the track is to correspond to a target. Taking the log and rewriting (3) in terms of track scores yields this simple form:

$$\log \Pr(\tau \mid z) = \sum_{u=1}^{\mid T \mid} s_u + \log h(\tau) + C$$  (4)

Thus, the optimization problem in Equation 2 can be formulated as an integer linear program:

$$\tau^* = \arg \max_{\tau} \quad s \cdot \tau$$
subject to \quad $\Omega \tau \leq 1$,

where $\Omega$ is a sparse matrix in which rows and columns correspond to observations and tracks, respectively, and $\Omega_{ij} = 1$ if track $i$ includes observation $j$ [1] [4].

### III. Track-oriented multiple hypothesis tracker

Integer linear programs are NP-complete, and the cost of solving (5) grows exponentially with the length of $\tau$. To maintain tractability the TOMHT takes an incremental approach, solving a restricted instance of (5) after incorporating each new scan and using the solution to greedily prune away unlikely tracks. Thus, each instance of (5) is an optimization over a subset of tracks that remains small even after many scans have been processed. To construct and maintain this candidate set, the TOMHT uses a data structure called a track tree.

### A. Track trees

A track tree is a rooted, tree-structured graph in which nodes correspond to observations and every path from the root to a leaf corresponds to a possible track. The
TOMHT maintains a collection of track trees, which together represent all candidate tracks. Each time a new scan is received, the leaves of these trees are extended with children corresponding to the new observations. Each observation of the new scan also serves as the root of a new track tree.

Figure 2 shows the collection of track trees resulting from the three scans of data in Figure 1. Note that while track deaths do not correspond to observations, it is convenient to represent them as nodes in the trees so that there is a one-to-one correspondence between leaves and tracks. The track trees in Figure 2 represent 24 possible tracks: 12 of length 3, 7 of length 2, and 5 of length 1.

B. Pruning

As described, the number of leaves in a track tree grows exponentially with the number of scans. The TOMHT limits this growth using a strategy called n-scan pruning. After incorporating the kth scan, the TOMHT solves (5) to find \( \tau^* \). It then prunes all tracks that do not satisfy at least one of the following two conditions:

1. Share a common ancestor with one of the tracks selected by \( \tau^* \) within the n most recent scans.
2. Belong to a tree with height at most n.

In Figure 2, for example, 2-scan pruning would eliminate all of the light gray nodes and branches, reducing the number of candidate tracks from 24 to 10. Notice that n-scan pruning effectively eliminates all branching of the track trees above the most recent n scans.

Most implementations of the TOMHT apply additional pruning techniques such as gating, track score thresholding, and track merging [5]. Since these are standard techniques we omit a detailed discussion, but they are trivially compatible with the methods developed in this paper. In our implementation we use only n-scan pruning and elliptical gating. We set the gate size such that only 0.1% of true track extensions will be erroneously pruned (assuming our observation and dynamics models are correct). Also note that the modeling of missed detections provides some level of robustness to false positives in gating – if a true track extension is erroneously pruned, the TOMHT can continue extending the track under the assumption that the lack of observation is due to a missed detection.

IV. Estimating Track Marginals

In this section we consider the task of computing the marginal probability of a single track:

\[
b_u = \Pr(\tau_u = 1 \mid z) = \sum_{\tau \in \{0,1\}^T} \Pr(\tau \mid z) \mathbb{1}_{\tau_u = 1}
\]

The sum in Equation 6 ranges over the TOMHT hypothesis space, i.e., there is one element of \( \tau \) for each leaf in the TOMHT’s track trees. Since explicit summation is generally intractable, we consider two approaches to approximate marginalization: one based on the \( k \)-best hypotheses, and one based on variational inference algorithms operating on a factor graph representation of the track posterior.

A. \( k \)-best hypothesis estimator

As mentioned above, even with n-scan pruning it is infeasible to sum over the entire space of hypotheses in (6). However, it is often the case in practice that the posterior probability mass is concentrated on a relatively small fraction of the hypotheses. The \( k \)-best estimator takes advantage of this tendency by restricting the sum to a tractable subset of hypotheses with high mass. Intuitively, if the combined mass of discarded hypotheses is small enough, the estimated track marginals will be close to the exact values. Formally, the \( k \)-best estimator is defined as follows:

\[
b_{u}^{k\text{best}} = \sum_{\tau \in \{\tau^{(1)}, \ldots, \tau^{(k)}\}} \Pr(\tau \mid z) \mathbb{1}_{\tau_u = 1} \sum_{\tau \in \{\tau^{(1)}, \ldots, \tau^{(k)}\}} \Pr(\tau \mid z)
\]

where \( \tau^{(i)} \) is the hypotheses with the \( i \)th highest posterior probability mass. Note that the hypothesis probabilities, \( \Pr(\tau \mid z) \), are themselves intractable since they must be normalized over the entire hypothesis space. Fortunately, in Equation 7 the normalizing constants cancel and \( b_{u}^{k\text{best}} \) is easily computable for moderate \( k \).

The most expensive piece of this estimator is the computation of the \( k \)-best hypotheses. Recall the integer program used to compute the most likely hypothesis (5). The same approach can be used to compute the next best hypothesis by adding a single linear constraint excluding \( \tau^{(1)} \) from the solution space:

\[
\sum_{u: \tau_u^{(1)} = 0} \tau_u + \sum_{u: \tau_u^{(1)} = 1} (1 - \tau_u) > 0
\]  

Repeating this process, alternately solving the integer program and excluding the previous solution, yields the top \( k \) hypotheses. Each successive integer program is more complex than the previous, so in practice this approach scales worse than linearly with \( k \). Faster algorithms exist to approximate the \( k \)-best hypotheses [6] [7] [8] [9], but are not explored here.

The number of hypotheses, \( k \), serves as a tuning parameter to trade off speed and accuracy. Setting \( k \) to the total number of valid hypotheses results in an exact – but intractable – algorithm. Setting \( k \) to 1, on the other hand, corresponds to the MAP estimator.
Fig. 2. (Top) The complete set of track trees resulting from the example scenario in Figure 1. Empty and crossed circles represent missed detections and track deaths, respectively. (Bottom) If the most likely hypothesis at time step 3, $\tau^*$, contains just the single track $\{ (1, 2), (2, 1), (3, 2) \}$, 2-scan pruning will prune away the light gray portions of the track trees. Since the leftmost tree has height greater than 2 and does not contain any tracks in $\tau^*$, the entire tree is pruned. The rightmost three trees all have height less than or equal to 2, so they are preserved entirely. The remaining tree has depth 3, so it is vulnerable to pruning; the unpruned tracks are those that share a common ancestor with the track in $\tau^*$ in scan 2.

probability 1 to every track in $\tau^*$ and 0 to the rest. The accuracy of the estimator at small values of $k$ depends the fraction of probability mass concentrated in the top $k$ hypotheses.

The $k$-best estimator can be viewed as a partial translation from the track-oriented representation of the TOMHT to the hypothesis-oriented representation of its predecessor, the HOMHT [10]. The TOMHT’s advantage over the HOMHT stems from its ability to represent a number of hypotheses that is exponential in the number of candidate tracks. Since the $k$-best approach discards this advantage by explicitly converting to the hypothesis-oriented representation, it makes sense to consider marginalization methods that operate directly in the track-oriented data association space.

**B. Variational message passing for marginalization**

Variational message passing algorithms for graphical models offer an alternate approach to approximating the marginalization in Equation 6. By taking advantage of conditional independencies present in the model, these algorithms can often obtain quick marginal estimates even when exact marginalization is very costly. This section briefly introduces two such algorithms – belief propagation and generalized belief propagation – and describes their application to the track posterior distribution.

Let $y = [y_1, \ldots, y_n]$ be a random vector distributed as:

$$
\Pr(y) = \frac{1}{Z} \prod_{\alpha=1}^{\mathcal{F}} f_\alpha(y_\alpha),
$$

where $\mathcal{F} = \{ f_\alpha \}$ is a set of nonnegative functions – often called factors – each defined over a subset of the variables, and $Z$ is a normalizing constant. The factor graph representation of $\Pr(y)$ is a bipartite graph with one node for each variable $y_\alpha \in y$ and factor $f_\alpha \in \mathcal{F}$. Edges connect factors to the variables in their domains. Belief propagation operates by “passing messages” between factor nodes and variable nodes. Each message is a function of a single variable, represented as a real-valued vector with one element for each value in the variable’s domain. Message-passing then corresponds to the following fixed-point iteration [11]:

$$
m_{i\rightarrow\alpha}(y_i) \propto \prod_{\beta \in \mathcal{N}(i) \setminus \alpha} m_{\beta\rightarrow i}(y_i),
$$

$$
m_{\alpha\rightarrow i}(y_\alpha) \propto \sum_{y_{\beta \mid Y_{i}=y_i}} f_\alpha(y_\alpha) \prod_{j \in \mathcal{N}(\alpha) \setminus i} m_{j\rightarrow \alpha}(y_j),
$$

where $m_{i\rightarrow \alpha}$ is the message from variable $i$ to factor $\alpha$, $m_{\alpha\rightarrow i}$ is the reverse message, and $\mathcal{N}(\cdot)$ is the graph neighborhood function. Once the message-passing reaches convergence, approximate marginals of individual variables (called beliefs in BP parlance) can be computed as the normalized product of incoming messages:

$$
b_i(y_i) \propto \prod_{\beta \in \mathcal{N}(i)} m_{\beta\rightarrow i}(y_i).
$$
The above message-passing procedure is designed to minimize a function known as the Bethe free energy [12]. When the factor graph is tree-structured, this iteration is guaranteed to converge and the resulting beliefs are exactly equal to the true marginals. For non-tree-structured (loopy) graphs, convergence is not always guaranteed [13], and even at convergence the beliefs no longer correspond to the true marginals. The quality of BP marginals on loopy graphs can vary from extremely accurate to extremely inaccurate depending on the numerical and structural properties of the factor graph. Some theoretical results exist to bound the error in BP’s beliefs [14], but in practice empirical evaluations are commonly used to determine whether BP is suitable for a given problem domain.

Generalized belief propagation (GBP), an algorithm related to BP, trades computational complexity for increased accuracy and robustness [15]. Theoretically, GBP replaces the Bethe free energy with a more complex free energy that explicitly accounts for some of the cycles in the graph. Algorithmically, this results in a fixed-point algorithm similar to BP but which computes beliefs over larger clusters of variables. The selection of these clusters is an important tuning parameter of the GBP algorithm; larger clusters generally lead to more accurate inference, but the complexity of the algorithm is exponential in the size of the largest cluster. There are a number of common heuristics for cluster selection, e.g. [15], [16], [17]. In our implementation we use the automated cluster selection routine described in [16]. This routine is itself parameterized by a maximal cluster size, resulting in a family of algorithms where we use GBP-i to indicate a GBP algorithm with no more than \(i + 1\) variables in a given cluster.

To use these algorithms for track marginal estimation, we first formulate the track posterior distribution (4) as a factor graph. Figure 3 shows part of the factor graph corresponding to the example data of Figure 1. The full factor graph contains one binary variable \(y_i\) corresponding to each track tree node within the last \(n\) scans (since the TOMHT has made hard data association decisions for earlier scans, there is no uncertainty to model). Each variable \(y_i\) serves as an indicator for the partial track terminating in its corresponding node in the track tree. Note that the variables corresponding to track tree leaf nodes are indicators for complete tracks; there is a one-to-one correspondence between these track indicator variables and the elements of \(\tau\), and we refer to the set of these variables as \(\mathcal{T}\).

The factors are grouped into three classes: tree constraints, global constraints, and track score factors. Together, the two classes of constraint factors encode our assumptions limiting each observation to at most one track and each track to at most one observation per scan. Denote by \(y_{ch(i)}\) the variables that are children of an arbitrary variable \(y_i\) (borrowing parent-child relationships from the corresponding track tree). Then we define tree constraints \(f^t_i\) and global constraints \(f^g_{k,j}\) as follows:

\[
f^t_i(y_i, y_{ch(i)}) = \begin{cases} 1 & : (y_i = 0 \text{ and } \sum_{y_l \in y_{ch(i)}} y_l = 0) \\ 0 & : \text{otherwise} \end{cases}
\]

\[
f^g_{k,j}(y^{k,j}) = \begin{cases} 1 & : \sum_{y_k \in y^{k,j}} y_k \leq 1 \\ 0 & : \text{otherwise} \end{cases}
\]

where \(y^{k,j}\) denotes the set of variables corresponding to the observation \(z^{k,j}\). There is one tree constraint factor for each non-leaf node and one global constraint for each unique observation. Every instantiation of the leaf variables corresponds to a hypothesis, and the constraint factors assign zero probability to all invalid hypotheses. Track score factors, \(f^s_i\), weight the hypotheses according to the scores of their constituent tracks:

\[
f^s_i(y_i) = \begin{cases} \exp(s_i) & : y_i = 1 \\ 1 & : y_i = 0. \end{cases}
\]

There is one score factor for each track indicator variable \(y_i \in \mathcal{T}\).

Thus, the probability mass function represented by the factor graph may be written as:

\[
\Pr(y) \propto \prod_{y_i \notin \mathcal{T}} f^t_i(y_i, y_{ch(i)}) \prod_{z^{k,j} \in \mathcal{T}} f^g_{k,j}(y^{k,j}) \prod_{y_i \in \mathcal{T}} f^s_i(y_i).
\]

Any instantiation of the variables will evaluate to the exponentiated sum of the selected track scores if it corresponds to a valid hypothesis, and zero otherwise. Thus, the marginal distribution of the track indicator variables is identical to the track posterior in (4).

The constraint factors may be defined over a large number of variables, making direct computation of the factor message in Equation 11 intractable. Fortunately, the sparsity structure of these factors admits an efficient reformulation that reduces the complexity from exponential to linear in the number of variables. See [18] for a general treatment of optimizations of this sort. The exact forms of the efficient message updates for this model are given in the Appendix.

Thus, each iteration of BP takes time \(O(|\mathcal{T}|)\). Since solving the IP – a crucial component of the TOMHT – has worst-case complexity \(O(2^{|\mathcal{T}|})\), using BP to estimate track marginals incurs only a modest overhead. On the
\[
\log \Pr(z, \tau, x) = C + \sum_{u=1}^{\lvert T \rvert} \tau_u \left[ \log \frac{\lambda_v}{\lambda_\phi} + \log \Pr(x^{u,1}) + \log f_o(z^{\tau_u,1} \mid x^{u,1}) \right] + \sum_{v=2}^{\lvert T \rvert_1} \left( \log f_d(x^{u,v} \mid x^{u,v-1}) + \mathbb{1}[\tau_{u,v} \neq (\ast, 0)] \log \left( \frac{p_D f_o(z^{\tau_{u,v}} \mid x^{u,v})}{\lambda_\phi} \right) + \mathbb{1}[\tau_{u,v} = (\ast, 0)] \log (1 - p_D) \right) \]
\]

\[
\log f_d(x^{u,v} \mid x^{u,v-1}) = -\frac{d_x}{2} \log (2\pi) - \frac{1}{2} \log |Q| - \frac{1}{2} (x^{u,v} - Ax^{u,v-1})^\top Q^{-1} (x^{u,v} - Hx^{u,v-1})
\]

\[
\log f_o(z \mid x) = -\frac{d_z}{2} \log (2\pi) - \frac{1}{2} \log |R| - \frac{1}{2} (z - Hx)^\top R^{-1} (z - Hx)
\]

other hand, a naive implementation of GBP-i has complexity \(O(2^\lvert T \rvert)\), which can quickly become significant.

Having formulated the track posterior as a factor graph, we then run the BP and GBP algorithms to compute approximate track marginals. These methods are potentially attractive due to their computational efficiency. However, their accuracy depends heavily on model-specific characteristics. In some cases it is possible to guarantee convergence [19], but even then accuracy must be evaluated empirically [20]. Thus, to assess their accuracy on models generated by the TOMHT we conduct an empirical evaluation relative to known ground truth marginals in Section VI.

Graphical models and approximate inference methods have also been applied to other formulations of the data association problem. The most closely related work is Williams et al. [20], which considers a graphical model formulation of the data association problem and uses BP to estimate marginal associations. Their treatment of multi-scan association probabilities is similar in spirit to ours, but instead of the popular track-oriented MHT representation it uses a hybrid “target-oriented” representation. Chen et al. [21] uses BP to approximate association probabilities in a sensor network, but only considers associations within a single scan. To the best of our knowledge, this is the first work to compute marginal multi-scan association probabilities using the representation of the TOMHT.

V. PARAMETER ESTIMATION VIA EM

In single-target tracking, parameter estimation is often carried out using the EM algorithm, treating the target state as “missing data” [3]. In this section we show that the same strategy can be used to estimate parameters in the multi-target setting. The multi-target E-step decomposes naturally into two stages: computing marginal probabilities of the track indicator variables and computing smoothed state estimates for the candidate tracks. The M-step then proceeds as in the single-target case, with tracks weighted by their marginal probabilities.

To see this, consider the complete data log-likelihood (CDLL) in Equation 13. With linear Gaussian observation and dynamics models, the CDLL involves many terms of the form \(\tau_u g(\cdot)\), where \(g(\cdot)\) is some linear or quadratic function of \(x^u\). Computing the expected CDLL (ECDLL) thus involves expectations like the following:

\[
E[\tau_u g(\cdot)] = \int \sum_{\tau_v \in \{0, 1\}} \tau_u g(\cdot) \Pr(x, \tau_v \mid z) = \Pr(\tau_u = 1 \mid z) \int_{x^u} g(\cdot) \Pr(x^u \mid \tau_u = 1, z) \]
\]

The term marked (a) is the marginal probability of a track indicator, a quantity for which we have several tractable estimators (Section IV). The second term, (b), is a low-order moment of a target’s state variables
conditioned on its particular track. Computation of such moments amounts to single-target smoothing – a tractable, well understood problem. In our case, we use standard Kalman smoothing recursions to compute these efficiently and exactly [3]. For additional details of the derivation see [22]. Since computation of the track-conditioned state moments is isolated from the data association uncertainty, track marginals can be viewed as the key ingredient in a reduction from multi-target to single-target parameter estimation.

The TOMHT makes hard data association decisions while processing scans in sequence, so it is critical to update the system parameters online instead of waiting until all of the data has been processed. Since online EM is not the focus of this work, we implement a simple incremental batch approach in our experiments. This has the undesirable effect of making the worst-case time complexity linear in the total number of scans processed, since each Kalman smoothing pass operates over the entire length of a candidate track. In principle, one could avoid this by using a more sophisticated EM algorithm designed for online processing of dependent observations [23]. In our approach we perform 10 iterations of batch EM at each time step, using the full length of all tracks in the TOMHT’s candidate set. Note that while the complexity of smoothing each candidate track is linear in the number of scans processed, the number of candidate tracks is still constrained by n-scan pruning.

Due to the approximate track marginals used in the E-step, some iterations may decrease the likelihood. The hope is that the marginals are sufficiently accurate to produce good parameter estimates in spite of this approximation.

VI. EXPERIMENTAL RESULTS

A. Direct evaluation of marginalization accuracy

Both the k-best estimator and the variational approximations sacrifice correctness to attain tractability, and it is not immediately clear how the respective approximations impact the resulting marginal estimates. We conducted an empirical study using simulated data, enabling direct comparison of the estimated marginals to their exact values.

The input to each algorithm is a collection of track trees representing the state of a TOMHT after processing a sequence of scans. In the context of these experiments, we will refer to such track forests as models. Model generation is a three step process:

1) Simulate a set of target state trajectories.
2) Sample observations at each time step according to a specified sensor model.
3) Process the simulated scans with a TOMHT.

At the end of this process, the track trees stored by the TOMHT are saved to a model file. We generated three groups of models, each corresponding to a different set of underlying target state trajectories as shown in Figure 4. The three groups are designed to explore a range of problem characteristics, subject to the constraint that exact inference must be feasible so that the approximate marginals can be evaluated against exact values.

The simulation used a linear Gaussian observation model with the following parameters:

\[
H = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad R = \begin{bmatrix} .5^2 \end{bmatrix}, \quad \lambda_\phi = 0, \quad \lambda_\nu = 0, \quad p_D = .95, \quad p_\gamma = 0.
\]

Tracking was performed using a TOMHT with the true
Fig. 5. Marginal error vs. running time. The blue diamond represents BP, green squares GBP, and red circles the $k$-best estimator. The numbers next to GBP and $k$-best data points correspond to the maximum cluster size and value of $k$, respectively. The $y$-axis plots the mean squared error in our estimates of the track marginal PMF, $P_t(\tau_u = 1 \mid z)$. Each data point represents the median value over 50 models. In Group A, GBP-15 and GBP-20 are not shown because GBP-10 already achieves perfect accuracy.

Fig. 6. Detailed marginalization performance on a single model from Group B. Each marker plots the estimated vs. exact marginal probability for a single track. BP underestimates the marginal probabilities of most tracks, with the exception of a few high-probability tracks that it overestimates. GBP-10 exhibits similar performance but is more tightly clustered around the zero-error line. The 10-best estimator has a very different profile: it very accurately estimates marginals of the most likely tracks, but assigns zero probability to all of the less likely tracks.

Fig. 7. Effect of increasing model size on the $k$-best estimator. These plots show estimated vs. exact track marginals for the BP, 10-best, and 100-best estimators on a scene with ten times as many observations as that used in Figure 6. The performance profile of BP remains qualitatively the same, but the concentration of the $k$-best estimator on the most likely tracks increases sharply. The 10-best estimator assigns zero probability mass to several tracks with true marginals exceeding 0.1. Even the more expensive 100-best estimator does little to improve the estimates.
sensor model and remaining parameters set as follows:

\[
A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \quad Q = \begin{bmatrix} .5^2 & 0 \\ 0 & .2^2 \end{bmatrix} \quad \lambda_\phi = 1 \quad \lambda_\nu = 1 \quad p_\gamma = .1
\]

The tracker used 4-scan pruning; models generated with 3-scan pruning were too small to be interesting, and those generated with 5-scan pruning were too large to solve exactly. We generated 50 models in each group, and for each model computed approximate marginals using \( k \)-best, BP, and GBP, as well as the exact marginals via variable elimination. The median number of variables per model is 63 for Group A, 197 for B, and 264 for C.

Figure 5 plots the running time and marginalization error for several \( k \)-best and messaging-passing estimators. As expected, increasing \( k \) for the \( k \)-best estimator and the cluster size for GBP both decrease marginalization error. The 1-best estimator universally has the highest error and GBP-20 the lowest. BP typically produces marginals with error between 1-best and 10-best.

BP is, by far, the fastest algorithm. While the computational complexity of GBP is tunable, the implementation used in our experiments is from a general-purpose inference package. It is written in C++ and reasonably efficient, but does not take advantage of the sparsity structure of the factors as our BP implementation does, resulting in an overhead that dominates the computation time for small cluster sizes. On the other hand, the \( k \)-best estimator scales poorly with \( k \); with \( k = 100 \) it already takes 10 seconds to run in groups B and C. Values up to \( k = 10 \) may be usable in practice, especially if using a more sophisticated \( k \)-best solver, but even the 1-best estimator is slower than BP. This is not surprising – the 1-best estimator finds the exact MAP configuration, whereas BP only approximates the marginals. The appropriate algorithm for a particular application will depend both on its performance requirements and its sensitivity to error in the marginals. If the application can tolerate moderate error, BP is attractive due to its high efficiency. If higher accuracy is required, then a more expensive GBP or \( k \)-best estimator may be preferable. Finally, note that we only restricted ourselves to models of this size to enable exact inference for evaluation purposes; the approximate algorithms could all be run on much larger problems. For example, the overhead incurred by BP is less than the time it takes to solve the IP, so any real-time TOMHT deployment could add in BP-based marginal estimation for a relatively small additional cost.

To better illustrate the accuracy profiles of the various estimators, Figure 6 plots estimated vs. exact marginals for the individual tracks of a single model from Group B. Note that the 10-best estimator is very accurate for the highest probability tracks, but assigns zero probability to all tracks with probability less than 0.06. BP and GBP, on the other hand, are less accurate on high probability tracks but do better on the majority of less likely tracks.

As model size increases, the \( k \)-best estimator’s concentration of mass in the few most likely tracks becomes even more exaggerated. In the extreme case, the top \( k \) hypotheses may actually be very minor perturbations of \( \tau^{(1)} \), in which case the \( k \)-best estimator will be little better than the 1-best estimator. Demonstrating this effect experimentally is non-trivial: exact marginalization is generally intractable for large models. To circumvent this difficulty we construct a somewhat artificial scene, replicating the crossing paths example from Figure 4 across space. The result is a scene with 10 simultaneous pairs of crossing tracks, each pair spatially separated from the others such that the TOMHT, with gating, produces a set of completely independent sub-models. This independence allows us to compute exact marginals despite the larger model size.

Figure 7 plots estimated vs. exact track marginals for this larger model. The BP results are qualitatively unaffected, but the 10-best estimator is notably worse: it does not support any tracks with marginal probability less than 0.3. The 100-best estimator performs only slightly better. In models like this, with completely separated components, a modified \( k \)-best estimator could leverage the independence to improve its accuracy. However, this proliferation of likely hypotheses is likely to arise in any large tracking scenario (not just those with completely separated subproblems) and thus highlights a significant weakness of the \( k \)-best approach to marginalization.

B. Effect of parameter estimation on track quality

In this section we demonstrate the use of EM for estimating the dynamics noise matrix while performing real-time tracking. We begin by simulating data with linear Gaussian dynamics and observation models parameterized as follows:

\[
A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \quad Q = \begin{bmatrix} .1^2 & 0 \\ 0 & .2^2 \end{bmatrix} \quad H = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad R = \begin{bmatrix} .25^2 \end{bmatrix} \\
\lambda_\nu = .1 \quad \lambda_\phi = 3 \quad p_D = .95 \quad p_\gamma = .05
\]

To encourage frequent track crossings, we slightly bias targets’ velocity state components toward the origin. We generated 50 such scenarios, each 200 scans in duration. A segment of one simulated scenario is shown in Figure 8. Note that we are not computationally restricted to...
examples of this size. However, large tracking simulations are harder to interpret and often not substantially more difficult – a page of criss-crossing tracks with two spatial dimensions might actually involve very few association ambiguities. For these reasons we prefer smaller examples that are easier to visually dissect.

Very minor parameter misspecification will not affect the data association decisions made by the TOMHT, so the only impact will be slightly degraded target state estimates. More significant misspecification may affect the optimal solution to (5), leading to false tracks, split tracks, and missed tracks in addition to state estimation error. By reducing the level of misspecification, EM has the potential to mitigate errors of both types.

To test this hypothesis, we processed each scenario with seven different initial values of $Q$. All initial values of $Q$ were diagonal, corresponding to independent noise on the position and velocity state components. We began with the covariance matrix used in the simulation process – call this matrix $Q^*$. Since states were perturbed to encourage track crossings these should not be thought of as “true” parameter values, but they are likely close to optimal since the perturbations were not large. From there we generated three covariance matrices with overestimates of the noise variance, doubling the standard deviations (SDs) each time. Finally, we generated three corresponding matrices underestimating the noise variance, halving the SDs rather than doubling them. The result is a set of covariance matrices indexed by the scaling factors shown along the x-axis of Figure 9.

For each scenario and initial value of $Q$ we ran a TOMHT with four different EM configurations: (1) no EM; (2) EM using BP for marginalization; (3) EM using the 1-best estimator for marginalization; and (4) EM with 10-best. Comparing tracker performance under these four configurations allows us to determine whether online EM is useful and which marginalization algorithms work best in that context.

Evaluating the output of a multi-target tracking system is a complex task in itself. There are many classes of potential error, including missed tracks, spurious tracks, merged or split tracks, and state estimation error. We use a metric called *optimal subpattern alignment for tracks* (TOSPA) to weight and combine all such errors into a single number summarizing the deviation of the tracker’s output state trajectories from the ground truth [24].

Figure 9 plots tracker error as a function of the initial parameter values for the four different EM configurations. Looking at the “No EM” line, we see that the optimal noise standard deviations are near 1 and 2, as expected. Without EM, initializing the noise SDs to either larger or smaller values leads to worse tracker performance. With online EM, however, the initial setting of $Q$ has little impact on tracker error, indicating that the parameters quickly converge to near-optimal values. All marginalization methods seem to perform comparably, except for the 1-best estimator which causes catastrophic failure with the largest initial noise SDs.

Figure 10 shows an example of how parameter misspecification and EM affect tracker output. The left-most and center ellipses highlight successes of EM. On the left, we see that inflated variance parameters cause the tracker to miss a track entirely, but with EM it is able to recover and identify the track. The middle shows an instance where inflated variance parameters cause the
Our results show that BP produces marginals with only modest error very quickly. The $k$-best approach could, in principle, achieve arbitrarily high accuracy, but is impractically slow for large $k$. Also, as the problem size grows, $k$-best estimators quickly degrade in quality.

We also demonstrated the utility of approximate track marginals in the context of an EM algorithm for parameter estimation. Without online estimation, poor initial specification of the process noise covariance matrix leads to increased data association and state estimation error. Updating the covariance matrix online, using approximate marginals in the E-step of the EM algorithm, made the TOMHT much more robust to the initial parameter values. In our experiments, the BP marginals were just as effective as the GBP and 10-best marginals at improving tracker performance. As the fastest algorithm we considered, BP should be preferred whenever its output is sufficiently accurate for the task at hand.

**APPENDIX**

**Efficient factor messages for BP**

Naïve computation of the factor messages as defined in (11) has complexity exponential in the number of neighbors. However, on the factor graph defined in Section IV-B the sparsity structure in the constraint factors admits the following equivalent linear-time updates:

Tree constraint factor $\alpha$ to parent variable $i$:

$$m_{\alpha \rightarrow j}(0) \propto \prod_{j \in Ch(\alpha)} m_{j \rightarrow \alpha}(0)$$

$$m_{\alpha \rightarrow j}(1) \propto \sum_{j \in Ch(\alpha)} \left[ m_{j \rightarrow \alpha}(1) \prod_{k \in Ch(\alpha) \setminus j} m_{k \rightarrow \alpha}(0) \right]$$

Tree constraint factor $\alpha$ to child variable $j$:

$$m_{\alpha \rightarrow j}(0) \propto m_{Pa(\alpha) \rightarrow \alpha}(0) \prod_{k \in Ch(\alpha) \setminus j} m_{k \rightarrow \alpha}(0) + \sum_{k \in Ch(\alpha) \setminus j} \left[ m_{Pa(\alpha) \rightarrow \alpha}(1) m_{k \rightarrow \alpha}(1) \prod_{l \in Ch(\alpha) \setminus \{j,k\}} m_{l \rightarrow \alpha}(0) \right]$$

$$m_{\alpha \rightarrow j}(1) \propto m_{Pa(\alpha) \rightarrow \alpha}(1) \prod_{k \in Ch(\alpha) \setminus j} m_{k \rightarrow \alpha}(0)$$

Global constraint factor $\alpha$ to neighbor variable $i$:

$$m_{\alpha \rightarrow i}(0) \propto \prod_{j \in \mathcal{N}(\alpha) \setminus i} m_{j \rightarrow \alpha}(0)$$

$$+ \sum_{j \in \mathcal{N}(\alpha) \setminus i} \left[ m_{j \rightarrow \alpha}(1) \prod_{k \in \mathcal{N}(\alpha) \setminus \{i,j\}} m_{k \rightarrow \alpha}(0) \right]$$

$$m_{\alpha \rightarrow i}(1) \propto \prod_{j \in \mathcal{N}(\alpha) \setminus i} m_{j \rightarrow \alpha}(0)$$