Bayesian Multi-target Tracking with Merged Measurements Using Labelled Random Finite Sets

Michael Beard, Ba-Tuong Vo, and Ba-Ngu Vo

***Abstract*—Most tracking algorithms in the literature assume that the targets always generate measurements independently of each other, i.e. the sensor is assumed to have infinite resolution. Such algorithms have been dominant because addressing the presence of merged measurements increases the computational complexity of the tracking problem, and limitations on computing resources often make this infeasible. When merging occurs, these algorithms suffer degraded performance, often due to tracks being terminated too early. In this paper, we use the theory of random finite sets (RFS) to develop a principled Bayesian solution to tracking an unknown and variable number of targets in the presence of merged measurements. We propose two tractable implementations of the resulting filter, with differing computational requirements. The performance of these algorithms is demonstrated by Monte Carlo simulations of a multi-target bearings-only scenario, where measurements become merged due to the effect of finite sensor resolution.**

***Index Terms*—Multi-target tracking, merged measuremnts, unresolved targets, random finite sets**

I. Introduction

THE sensor models used by the majority of traditional multi-target tracking algorithms are based on the assumption that each target generates measurements independently. While this facilitates the development of computationally cheaper tracking algorithms, most real-world sensors violate this assumption when targets become close together in the measurement space. A common situation that involves merged measurements is when the sensor has finite resolution [1]. For example, most radar and sonar systems divide the measurement space into discrete detection cells, and when multiple targets fall within the same cell, they result in a merged measurement. Another example where merging occurs is in computer vision, where detection algorithms often produce merged measurements for groups of objects appearing close together or occluding one-another in an image. In such cases, sensors often generate fewer measurements than there are targets present, where groups of closely spaced targets produce merged measurements. If the tracking algorithm assumes that each target generates a measurement independent of each other, it will usually infer that some targets have disappeared, when in fact their measurements have become merged.

This work was supported in part by the Australian Research Council under Future Fellowship FT0991854, and Discovery Project DE120102388.

M. Beard is with the Defence Science and Technology Organisation, Rockingham, WA, Australia, and the Department of Electrical and Computer Engineering, Curtin University, Bentley, WA, Australia (email: michael.beard@dsto.defence.gov.au).

B.-T. Vo and B.-N. Vo are with the Department of Electrical and Computer Engineering, Curtin University, Bentley, WA, Australia (email: {ba-tuong.vo,ba-ngu.vo}@curtin.edu.au).

Since merged measurements are unavoidable when processing real world sensor data, and methods that assume conditionally independent measurements perform poorly when this occurs, it is fundamentally important to investigate algorithms that can deal with this phenomenon. A number of Bayesian multi-target tracking algorithms have been proposed to handle merged measurements. One such algorithm is Joint Probabilstic Data Association (JPDA), which was first applied to a finite resolution sensor model in [2]. The proposed algorithm (JPDAM) used a grid-based model for computing merging probabilities for two targets, which were then incorporated into the JPDA update. This technique was limited to cases where at most two targets are merged at a time. A useful extension to this was proposed in [3], [4] which can handle an arbirtary number of merged targets, by defining a more general resolution model based on combining pairwise merging probabilities. This provides a tractable approximation to the probabilities of different resolution events, which are again incorporated into the JPDA update. The inclusion of the resolution model in JPDA has been shown to improve its performance. However, the fundamental limitation of these methods is that JPDA implicitly assumes a fixed and known number of targets.

Several other methods have also been applied to merged measurement tracking. An approach based on Multiple Hypothesis Tracking (MHT) was proposed in [5], which used a two-target resolution model to maintain tracks in the presence of merged measurements. A similar resolution model was applied to Multiple Model JPDA in [6], [7], which is again limited to a maximum of two merged targets. Various numerical data association techniques have been developed for merged measurements in [8], [9] (Probabilistic Data Association), and [10] (Markov Chain Monte Carlo). In addition, Probabilistic MHT (PMHT) [11], and existence based methods such as Linear Multitarget Integrated PDA (LM-IPDA) [12], and Integrated Track Splitting (ITS) [13] have also been applied to this problem. These are all useful techniques, however, with the exception of LM-IPDA (which trades off performance for reduced complexity), they only handle a small number of targets. Moreover, due to the nature of their formulation, it is difficult to establish any results regarding their Bayes optimality, in the sense of minimising the posterior Bayes risk [14].

In recent times, the concept of random finite sets (RFS) and the tools provided by Finite Set Statistics (FISST) have received a great deal of attention, as they provide a principled framework for multi-object estimation for an unknown and time-varying number of objects [14]. One of the first meth-

ods to be derived using this framework was the Probability Hypothesis Density (PHD) filter [15], and Mahler proposed a technique in [16] for applying this filter to unresolved targets by modelling the multi-target state as an RFS of point clusters. This is a principled approximation which can deal with a variable number of targets. However, it does not provide labelled track estimates, and the model is limited to cases where merged measurements arise only when the targets are close together in the state space, as opposed to the measurement space. The latter is important in cases where the measurement space is a lower dimensional projection of the state space (e.g. bearings-only sensors), because points which are well separated in the state space may become close together (and therefore unresolvable) when projected onto the measurement space.

One critisism of the RFS framework has been that it resulted in algorithms that do not maintain target labels over time, i.e. performing multi-object filtering as opposed to tracking. It was shown in [17], [18] that the RFS framework does allow estimation of target tracks, and a computationally feasible multi-target tracker for the standard sensor model was derived, known as the generalised labelled multi-Bernoulli (GLMB) filter. Our goal in this paper is to further generalise the result of [17] to a sensor model that includes merged measurements, thereby making it suitable for application to a wider variety of real world problems. To achieve this, we introduce a multi-target likelihood function that takes into account the possible merging of target generated measurements, by considering feasible partitions of the target set, and the feasible assignments of measurements to groups within these partitions. To our knowledge, this is the first provably Bayes optimal multi-object tracker that accommodates merged measurements, with time varying and unknown number of targets. An advantage of our approach is that it can be parallelised to facilitate potential real-time implementation.

Preliminary results of this work have been presented in [20], which this paper builds upon by including a more complete exposition of the derivation and implementation of the merged measurement GLMB filter, including detailed pseudo-code. Furthermore, we propose a computationally cheaper variant of the algorithm using relaxed measurement assignment, and a more extensive simulation study is presented.

The paper is organised as follows. Section II provides some background on labelled random finite sets and their application to tracking with standard sensor models. In Section III we propose a merged measurement likelihood model, and use labelled random finite sets to develop a filter based on this model. Section IV presents an approximation technique, and introduces two methods to aid in its tractable implementation. Further details and pseudo-code are given in Section V. Section VI contains simulation results for a bearings-only multi-target tracking scenario, and finally, some concluding remarks are given in Section VII.

II. Background: RFS-based Tracking with Standard Sensor Models

The goal of recursive multi-object Bayesian estimation is to estimate a finite set of states $X_k \subset \mathbb{X}$, called the *multi-object state*, at each time k. The multi-object states X_k and *multi-object observations* $Z_k \subset \mathbb{Z}$ are modelled as random finite sets, and FISST is a framework for working with RFSs [14] based on a notion of integration/density that is consistent with point process theory [19].

At the previous time step $k-1$, the multi-object state is assumed to be distributed according to a *multi-object density* $\pi_{k-1}(\cdot|Z_{1:k-1})$, where $Z_{1:k-1}$ is an array of finite sets of measurements recieved up to time $k-1$. Each Z_k is assumed to be generated through a process of thinning of misdetected objects, Markov shifts of detected objects, and superposition of false measurements. The *multi-object prediction* to time k is given by the Chapman-Kolmogorov equation

$$\pi_{k|k-1}(X_k|Z_{1:k-1}) = \int f_{k|k-1}(X_k|X)\,\pi_{k-1}(X|Z_{1:k-1})\,\delta X, \tag{1}$$

where $f_{k|k-1}(X_k|X)$ is the multi-object transition kernel from time $k-1$ to time k, and the integral is the set integral,

$$\int f(X)\,\delta X = \sum_{i=0}^{\infty} \frac{1}{i!} \int_{\mathbb{X}^i} f(\{x_1,\ldots,x_i\})\,d(x_1,\ldots,x_i) \tag{2}$$

for any function f that takes $\mathcal{F}(\mathbb{X})$, the collection of all finite subsets of $\mathbb{X}$, to the real line. A new set of observations Z_k is received at time k, which is modelled by a *multi-object likelihood function* $g_k(Z_k|X_k)$. Thus the *multi-object posterior* at time k is given by Bayes rule

$$\pi_k(X_k|Z_{1:k}) = \frac{g_k(Z_k|X_k)\,\pi_{k|k-1}(X_k|Z_{1:k-1})}{\int g_k(Z_k|X)\,\pi_{k|k-1}(X|Z_{1:k-1})\,\delta X}. \tag{3}$$

Collectively, (1) and (3) are referred to as the *multi-object Bayes filter*. In general, computing the exact multi-object posterior is numerically intractable, and approximations are required to derive practical algorithms.

One of the first algorithms to be derived based on these concepts was the probability hypothesis density (PHD) filter [15], which achieves a tractable approximation to (3) by propagating only the first moment of the multi-object density. This was followed by the cardinalised PHD (CPHD) filter [22], which propagates the probability distribution on the number of targets, in addition to the first moment of the multi-object density. Another type of algorithm based on RFSs involves approximating the density as a multi-Bernoulli RFS, known as multi-object multi-Bernoulli (MeMBer) filtering [14], [23]. A common feature of the PHD and multi-Bernoulli approaches is that they do not require data association. However, a significant drawback is that they do not inherently produce target tracks, instead providing a set of unlabelled target estimates at each time step. A recently proposed technique to address this problem, which maintains the mathematical rigor of the RFS framework, is the concept of labelled random finite sets [17]. This technique involves assigning a distinct label to each element of the target set, so that the history of each object's trajectory can be naturally identified.

In this work, we are interested in algorithms that can produce continuous tracks, so we restrict our attention to methods based on labelled random finite sets. In [17] and [18],

an algorithm was proposed for solving the standard multi-object tracking problem, based on a type of labelled RFS called *generalised labelled multi-Bernoulli* (GLMB). We now review the main points of this technique, and in the next section we propose a generalisation, enabling it to handle merged measurements.

We begin by introducing some notation and definitions relating to labelled random finite sets. The multi-object exponential of a real valued function h raised to a set X is defined as $[h]^X \triangleq \prod_{x\in X} h(x)$, where $[h]^{\emptyset} = 1$, and the elements of X may be of any type such as scalars, vectors, or sets, provided that the function h takes an argument of that type. The generalised Kronecker delta function is defined as

$$\delta_Y(X) \triangleq \begin{cases} 1, & \text{if } Y = X \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where again, X and Y may be of any type, such as scalars, vectors, or sets. Finally, the set inclusion function is

$$1_Y(X) \triangleq \begin{cases} 1, & \text{if } X \subseteq Y \\ 0, & \text{otherwise} \end{cases}. \tag{5}$$

In general, we adopt the notational convention that labelled sets are expressed in bold upper case ($\boldsymbol{X}$), unlabelled sets in regular upper case (X), labelled vectors in bold lower case ($\boldsymbol{x}$), and unlabelled vectors or scalars in regular lower case (x). The pseudo-code in Section V does not strictly adhere to this convention, however, any exceptions should be clear from the context. We also use the notation $x_{1:n} \equiv (x_1, \ldots, x_n)$ to abbreviate arrays, and $\{x_{1:n}\} \equiv \{x_1, \ldots, x_n\}$ for sets.

Definition 1. A *labelled RFS* with state space $\mathbb{X}$ and discrete label space $\mathbb{L}$, is an RFS on $\mathbb{X}\times\mathbb{L}$, such that the labels within each realisation are always distinct. That is, if $\mathcal{L}_{\boldsymbol{X}}$ is the set of unique labels in $\boldsymbol{X}$, and we define the distinct label indicator function as $\Delta(\boldsymbol{X}) = \delta_{|\boldsymbol{X}|}(|\mathcal{L}_{\boldsymbol{X}}|)$, then a realisation $\boldsymbol{X}$ of a labelled RFS always satisfies $\Delta(\boldsymbol{X}) = 1$.

Definition 2. A *generalised labelled multi-Bernoulli* (GLMB) RFS is a labelled RFS with state space $\mathbb{X}$ and discrete label space $\mathbb{L}$, which satisfies the probability distribution

$$\pi(\boldsymbol{X}) = \Delta(\boldsymbol{X}) \sum_{c\in\mathbb{C}} w^{(c)}(\mathcal{L}_{\boldsymbol{X}}) \left[p^{(c)}(\cdot)\right]^{\boldsymbol{X}} \tag{6}$$

where $\mathbb{C}$ is an arbitrary index set, $\sum_{L\subseteq\mathbb{L}} \sum_{c\in\mathbb{C}} w^{(c)}(L) = 1$, and $\int_{x\in\mathbb{X}} p^{(c)}(x,l)\,dx = 1$.

The GLMB, as defined above, has been shown to be a *conjugate prior* with respect to the standard multi-object transition and multi-object observation functions [17] (i.e. the form of the density is retained through both prediction and update), which facilitates implementation of a Bayesian GLMB filter.

A. Multi-object Transition Kernel

Let $\boldsymbol{X}$ be the labelled RFS of objects at the current time with label space $\mathbb{L}$. A particular object $(x,l) \in \boldsymbol{X}$ has probability $p_S(x,l)$ of surviving to the next time with state (x_+, l_+) and probability density $f(x_+|x,l)\,\delta_l(l_+)$ (where $f(x_+|x,l)$ is the single target transition kernel), and probability, $q_S(x,l) = 1 - p_S(x,l)$ of being terminated. Thus, the set $\boldsymbol{S}$ of surviving objects at the next time is distributed as

$$f_S(\boldsymbol{S}|\boldsymbol{X}) = \Delta(\boldsymbol{S})\,\Delta(\boldsymbol{X})\,1_{\mathcal{L}_{\boldsymbol{X}}}(\mathcal{L}_{\boldsymbol{S}})\,[\Phi(\boldsymbol{S};\cdot)]^{\boldsymbol{X}} \tag{7}$$

where

$$\Phi(\boldsymbol{S};x,l) = \sum_{(x_+,l_+)\in\boldsymbol{S}} \delta_l(l_+)\,p_S(x,l)\,f(x_+|x,l) + (1 - 1_{\mathcal{L}_{\boldsymbol{S}}}(l))\,q_S(x,l). \tag{8}$$

Now let $\boldsymbol{B}$ be the labelled RFS of newborn objects with label space $\mathbb{B}$, where $\mathbb{L}\cap\mathbb{B} = \emptyset$. To ensure that $\mathbb{L}$ and $\mathbb{B}$ are disjoint, we adopt a labelling scheme such that each new birth is labelled with a pair (t_b, l_b), where t_b is the current time and l_b is a unique index. Since the time changes from scan to scan, the space of birth labels is always disjoint from the space of surviving labels. Since the births have distinct labels, and assuming that their states are independent, we model $\boldsymbol{B}$ according to a labelled multi-Bernoulli distribution (LMB)

$$f_B(\boldsymbol{B}) = \Delta(\boldsymbol{B})\,w_B(\mathcal{L}_{\boldsymbol{B}})\,[p_B(\cdot)]^{\boldsymbol{B}}. \tag{9}$$

where $p_B(\cdot,l)$ is the single target birth density, and $w_B(\cdot)$ is the birth weight (see Section IV-D of [17] for more details). We present the prediction based on an LMB birth model, however, this can easily be extended to the case of a GLMB birth model. The overall multi-object state at the next time step is the union of survivals and new births, i.e. $\boldsymbol{X}_+ = \boldsymbol{S}\cup\boldsymbol{B}$. The label spaces $\mathbb{L}$ and $\mathbb{B}$ are disjoint, and the states of new born objects are independent of surviving objects, hence $\boldsymbol{S}$ and $\boldsymbol{B}$ are independent. It was shown in [17] that the multi-object transition can be expressed as

$$f(\boldsymbol{X}_+|\boldsymbol{X}) = f_S(\boldsymbol{X}_+\cap(\mathbb{X}\times\mathbb{L})\,|\boldsymbol{X})\,f_B(\boldsymbol{X}_+ - (\mathbb{X}\times\mathbb{L})), \tag{10}$$

and that a GLMB density of the form (6) is closed under the Chapman-Kolmogorov prediction (1) with the transition kernel defined by (10).

B. Standard Multi-object Observation Model

Let $\boldsymbol{X}$ be the labelled RFS of objects that exist at the observation time. A particular object $\boldsymbol{x} \in \boldsymbol{X}$ has probability $p_D(\boldsymbol{x})$ of generating a detection z with likelihood $g(z|\boldsymbol{x})$, and probability $q_D(\boldsymbol{x}) = 1 - p_D(\boldsymbol{x})$ of being misdetected. Let D be the set of target detections. Assuming the elements of D are conditionally independent, then D is a multi-Bernoulli RFS with parameter set $\{(p_D(\boldsymbol{x}), g(\cdot|\boldsymbol{x})) : \boldsymbol{x} \in \boldsymbol{X}\}$, and we write its probability density as

$$\pi_D(D|\boldsymbol{X}) = \{(p_D(\boldsymbol{x}), g(\cdot|\boldsymbol{x})) : \boldsymbol{x} \in \boldsymbol{X}\}(D). \tag{11}$$

Let K be the set of clutter observations, which are independent of the target detections. We model K as a Poisson RFS with intensity $\kappa(\cdot)$, hence K is distributed according to

$$\pi_K(K) = e^{-\langle\kappa,1\rangle}\kappa^K. \tag{12}$$

The overall multi-object observation is the union of target detections and clutter observations, i.e. $Z = D\cup K$. Since

D and K are independent, the multi-object likelihood is

$$g\left(Z|\boldsymbol{X}\right) = \sum_{D\subseteq Z} \pi_D\left(D|\boldsymbol{X}\right)\pi_K\left(Z-D\right). \tag{13}$$

As demonstrated in [14], this can be equivalently expressed as

$$g\left(Z|\boldsymbol{X}\right) = e^{-\langle\kappa,1\rangle}\kappa^Z \sum_{\theta\in\Theta(\mathcal{L}_{\boldsymbol{X}})} \left[\psi_Z\left(\cdot;\theta\right)\right]^{\boldsymbol{X}} \tag{14}$$

where $\Theta\left(\mathcal{L}_{\boldsymbol{X}}\right)$ is the set of all one-to-one mappings of labels in $\boldsymbol{X}$ to measurement indices in Z, (i.e. $\theta : \mathcal{L}_{\boldsymbol{X}} \to \{0,1,...,|Z|\}$, such that $\left[\theta\left(i\right)=\theta\left(j\right)>0\right] \Rightarrow \left[i=j\right]$), and $\psi_Z\left(\cdot;\theta\right)$ is

$$\psi_Z\left(x,l;\theta\right) = \begin{cases} \frac{p_D(x,l)g\left(z_{\theta(l)}|x,l\right)}{\kappa\left(z_{\theta(l)}\right)}, & \theta\left(l\right)>0 \\ q_D\left(x,l\right), & \theta\left(l\right)=0 \end{cases} \tag{15}$$

It was demonstrated in [17] that a GLMB density of the form (6) is closed under the Bayes update (3) with likelihood function defined by (14).

III. RFS-based Tracking with Merged Measurements

A. Multi-object Likelihood Model for Merged Measurements

In [14], Mahler proposed an RFS-based technique for filtering with unresolved targets, which modelled the multi-target state as a set of point clusters. A point cluster was defined as a group of unresolved targets, and was represented as a location in the single-target state space, augmented with a positive integer which holds the number of targets that are effectively co-located at that point (relative to the sensor resolution). The likelihood function for this model contains a sum over partitions of the measurement set, and therefore a direct implementation would be computationally prohibitive. A PHD filter based on this model was proposed in [16], however, it has not been implemented in practice.

The point cluster model is intuitively appealing in cases where the measurements only become merged when the targets are close together in the state space. However, this implicitly assumes that the merging only depends on the target states. There are important cases where this assumption is too restrictive, examples of which are bearings-only tracking, and visual tracking. In these cases, targets that cannot be resolved along the line of sight may be separated by a considerable distance in the state space. The sensor's position clearly has an impact on whether measurements become merged, but this cannot be accounted for using the point cluster model. This occurs in general when the measurement space is a lower dimensional projection of the state space, such that well separated states can potentially give rise to closely spaced measurements.

To address these cases, we require a likelihood function that accommodates measurement merging, not only when targets are nearby in the state space, but more generally when they are nearby in the measurement space. To achieve this, we must consider partitions of the target set, where each group within a partition produces at most one (merged) measurement.

Definition 3. A *partition* of a set A is a disjoint collection of subsets of A, whose union is equal to A.

Figure 1 contains an example of some of the partitions of a set of five targets. In the diagram, only six of the total of 15 partitions are shown. In general, the number of partitions of a set of N targets is given by the N-th Bell number, which becomes extremely large with increasing N. However, in the context of multi-target tracking, it is usually possible to eliminate many partitions from consideration, since they are physically impossible or extremely unlikely to occur.

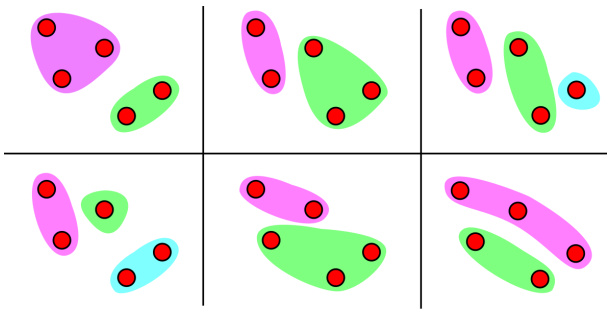


Figure 1. Some of the partitions of a set of five targets. Each shaded group of targets can give rise to at most one measurement.

We define our model for the multi-object measurement likelihood as a sum over partitions of the target set as follows,

$$g\left(Z|\boldsymbol{X}\right) = \sum_{\mathcal{U}(\boldsymbol{X})\in\mathcal{P}(\boldsymbol{X})} \mathring{g}\left(Z|\mathcal{U}\left(\boldsymbol{X}\right)\right) \tag{16}$$

where $\mathcal{P}\left(\boldsymbol{X}\right)$ is the set of all partitions of $\boldsymbol{X}$, and $\mathring{g}\left(Z|\mathcal{U}\left(\boldsymbol{X}\right)\right)$ is the measurement likelihood conditioned on the targets being observed according to partition $\mathcal{U}\left(\boldsymbol{X}\right)$. This bears some similarity to the approach taken in [4], insofar as they both consider partitions of the targets in order to form resolution events. However, the key difference is that our likelihood is defined as a function of a set of target states, instead of a vector with fixed dimension. This allows us to utilise FISST principles to arrive at an expression for the posterior density.

To obtain an expression for $\mathring{g}\left(Z|\mathcal{U}\left(\boldsymbol{X}\right)\right)$, we generalise the standard measurement likelihood in (14) to target groups as follows[1]

$$\mathring{g}\left(Z|\mathcal{U}\left(\boldsymbol{X}\right)\right) = e^{-\langle\kappa,1\rangle}\kappa^Z \sum_{\theta\in\Theta(\mathcal{U}(\mathcal{L}_{\boldsymbol{X}}))} \left[\check{\psi}_Z(\cdot;\theta)\right]^{\mathcal{U}(\boldsymbol{X})} \tag{17}$$

where $\Theta\left(\mathcal{U}\left(\mathcal{L}_{\boldsymbol{X}}\right)\right)$ is defined as the set of all one-to-one mappings of groups of target labels in $\mathcal{U}\left(\mathcal{L}_{\boldsymbol{X}}\right)$ to measurement indices in Z (i.e. $\theta : \mathcal{U}\left(\mathcal{L}_{\boldsymbol{X}}\right) \to \{0,1,...,|Z|\}$, where $\left[\theta\left(L\right)=\theta\left(J\right)>0\right] \Rightarrow \left[L=J\right]$), and $\check{\psi}_Z\left(\boldsymbol{Y};\theta\right)$ is a 'group likelihood' defined by

$$\check{\psi}_Z\left(\boldsymbol{Y};\theta\right) = \begin{cases} \frac{\check{p}_D(\boldsymbol{Y})\check{g}\left(z_{\theta(\mathcal{L}_{\boldsymbol{Y}})}|\boldsymbol{Y}\right)}{\kappa\left(z_{\theta(\mathcal{L}_{\boldsymbol{Y}})}\right)}, & \theta\left(\mathcal{L}_{\boldsymbol{Y}}\right)>0 \\ \check{q}_D(\boldsymbol{Y}), & \theta\left(\mathcal{L}_{\boldsymbol{Y}}\right)=0 \end{cases} \tag{18}$$

where $\check{p}_D\left(\boldsymbol{Y}\right)$ is the detection probability for target group $\boldsymbol{Y}$, $\check{q}_D\left(\boldsymbol{Y}\right) = 1-\check{p}_D\left(\boldsymbol{Y}\right)$ is the misdetection probability for group $\boldsymbol{Y}$, and $\check{g}\left(z_{\theta(\mathcal{L}_{\boldsymbol{Y}})}|\boldsymbol{Y}\right)$ is the likelihood of measurement $z_{\theta(\mathcal{L}_{\boldsymbol{Y}})}$

[1]For notational convenience, we use $\mathcal{U}\left(\boldsymbol{X}\right)$ to denote a partition of the label-state pairs in the labelled set $\boldsymbol{X}$, and $\mathcal{U}\left(\mathcal{L}_{\boldsymbol{X}}\right)$ to denote the equivalent partition of the labels only.

given target group $\boldsymbol{Y}$. For our simulations in Section VI, $\breve{g}(\cdot|\boldsymbol{Y})$ is modelled as a Gaussian centered around the mean of the group, with group size-dependent standard deviation. However, the form of this function in general will depend on the sensor characteristics, and there are no restrictions placed on its modelling, provided that it can be tractably computed. Note that in (17), the exponent $\mathcal{U}(\boldsymbol{X})$ is a set of target sets, and the base is a real valued function whose argument is a target set. Substituting (17) into (16) yields the likelihood

$$g(Z|\boldsymbol{X}) = e^{-\langle \kappa,1\rangle}\kappa^{Z}\sum_{\substack{\mathcal{U}(\boldsymbol{X})\in\mathcal{P}(\boldsymbol{X})\\ \theta\in\Theta(\mathcal{U}(\mathcal{L}_{\boldsymbol{X}}))}}\left[\breve{\psi}_Z(\cdot;\theta)\right]^{\mathcal{U}(\boldsymbol{X})}. \quad (19)$$

B. A General Form for a Merged Measurement Tracker

Definition 4. A *labelled RFS mixture* density on state space $\mathbb{X}$ and discrete label space $\mathbb{L}$, is a density of the form

$$\pi(\boldsymbol{X}) = \Delta(\boldsymbol{X})\sum_{c\in\mathbb{C}} w^{(c)}(\mathcal{L}_{\boldsymbol{X}})\, p^{(c)}(\boldsymbol{X}) \quad (20)$$

where $\sum_{L\subseteq\mathbb{L}}\sum_{c\in\mathbb{C}} w^{(c)}(L) = 1$, $p^{(c)}(\boldsymbol{X})$ is symmetric in the elements of $\boldsymbol{X}$, and $p^{(c)}(\{(\cdot,l_1),...,(\cdot,l_n)\})$ is a joint pdf in $\mathbb{X}^n$.

A labelled RFS mixture density differs from the GLMB density of definition 2 in that each $p^{(c)}(\boldsymbol{X})$ is a joint density of all the targets in $\boldsymbol{X}$, whereas in definition 2, it is restricted to a product of single-target densities. Note that any labelled RFS density can be written in the mixture form (20). We use the mixture form because it is convenient for the update step of our proposed tracker.

Proposition 5. *If the multi-object prior is a labelled RFS mixture density with probability distribution of the form (20), then the predicted multi-object density under the transition kernel (10) is given by*

$$\pi_+(\boldsymbol{X}_+) = \Delta(\boldsymbol{X}_+)\sum_{c\in\mathbb{C}}\sum_{L\subseteq\mathbb{L}} w^{(c)}_{+,L}\left(\mathcal{L}_{\boldsymbol{X}_+}\right) p^{(c)}_{+,L}(\boldsymbol{X}_+) \quad (21)$$

where

$$w^{(c)}_{+,L}(J) = 1_L(J\cap\mathbb{L})\, w_B(J-\mathbb{L})\, w^{(c)}(L)\,\eta^{(c)}_S(L) \quad (22)$$

$$p^{(c)}_{+,L}(\boldsymbol{X}_+) = [p_B(\cdot)]^{\boldsymbol{X}_+-\mathbb{X}\times\mathbb{L}}\, p^{(c)}_{S,L}(\boldsymbol{X}_+\cap\mathbb{X}\times\mathbb{L}) \quad (23)$$

$$p^{(c)}_{S,L}(\boldsymbol{S}) = \frac{\int p^{(c)}_{l_{1:|L|}}(x_{1:|L|})\prod_{i=1}^{|L|}\Phi(\boldsymbol{S};x_i,l_i)\,dx_{1:|L|}}{\eta^{(c)}_S(L)} \quad (24)$$

$$\eta^{(c)}_S(L) = \int\int p^{(c)}_{l_{1:|L|}}(x_{1:|L|})\prod_{i=1}^{|L|}\Phi(\boldsymbol{S};x_i,l_i)\,dx_{1:|L|}\delta\boldsymbol{S} \quad (25)$$

$$p^{(c)}_{l_{1:|L|}}(x_{1:|L|}) \equiv p^{(c)}\left(\left\{(x_1,l_1),\ldots(x_{|L|},l_{|L|})\right\}\right) \quad (26)$$

Note that for any function $f:\mathcal{F}(\mathbb{X}\times\mathbb{L})\to\mathbb{R}$, we have used the convention $f_{l_{1:n}}(x_{1:n}) = f(\{(x_1,l_1),\ldots,(x_n,l_n)\})$ to denote it as a function on $\mathbb{X}^n$, given the labels $l_{1:n}$.

Proposition 6. *If the prior is a labelled RFS mixture density with probability distribution of the form (20), then the posterior multi-object density under the likelihood function (19) is given by*

$$\pi(\boldsymbol{X}|Z) = \Delta(\boldsymbol{X})\sum_{c\in\mathbb{C}}\sum_{\substack{\mathcal{U}(\boldsymbol{X})\in\mathcal{P}(\boldsymbol{X})\\ \theta\in\Theta(\mathcal{U}(\mathcal{L}_{\boldsymbol{X}}))}} w^{(c,\theta)}_Z(\mathcal{U}(\mathcal{L}_{\boldsymbol{X}}))\, p^{(c,\theta)}(\mathcal{U}(\boldsymbol{X})|Z) \quad (27)$$

where

$$w^{(c,\theta)}_Z(\mathcal{U}(L)) = \frac{w^{(c)}(L)\,\eta^{(c,\theta)}_Z(\mathcal{U}(L))}{\sum_{c\in\mathbb{C}}\sum_{J\subseteq\mathbb{L}}\sum_{\substack{\mathcal{U}(J)\in\mathcal{P}(J)\\ \theta\in\Theta(\mathcal{U}(J))}} w^{(c)}(J)\,\eta^{(c,\theta)}_Z(\mathcal{U}(J))} \quad (28)$$

$$p^{(c,\theta)}(\mathcal{U}(\boldsymbol{X})|Z) = \frac{\left[\breve{\psi}_Z(\cdot;\theta)\right]^{\mathcal{U}(\boldsymbol{X})} p^{(c)}(\boldsymbol{X})}{\eta^{(c,\theta)}_Z(\mathcal{U}(\mathcal{L}_{\boldsymbol{X}}))} \quad (29)$$

$$\eta^{(c,\theta)}_Z(\mathcal{U}(L)) = \int\left[\breve{\psi}_Z(\cdot;\theta)\right]^{\mathcal{U}(L)} p^{(c)}_{l_{1:|L|}}(x_{1:|L|})\,dx_{1:|L|} \quad (30)$$

Proofs of Propositions 5 and 6 are given in the appendix. Technically, the family of labelled RFS mixture densities can be regarded as a conjugate prior with respect to the merged measurement and standard likelihood functions, due to closure under the Chapman-Kolmogorov prediction and Bayes update. However, this is a very broad class of priors, and is generally numerically intractable. For this reason, calling the labelled RFS mixture a conjugate prior may be misleading, so we refrain from using this term.

Although it may be possible to directly implement the filter from Propositions 5 and 6, it would quickly become intractable because there is no clear way of truncating the sum over the space of measurement-to-group associations $\Theta(\mathcal{U}(\mathcal{L}_{\boldsymbol{X}}))$. The reason is that each component in the overall density (20) consists of a single joint density encompassing all targets, so there may be dependencies between target groups, meaning that standard ranked assignment techniques cannot be applied. The next section proposes an approximation and two alternative methods for implementing a tractable version of the GLMB filter for the merged measurement likelihood.

IV. Tractable Approximations

The exact form of the filter as specified in Propositions 5 and 6, is intractable due to the presence of joint densities $p^{(c)}(\boldsymbol{X})$ in the prior (20). To derive an algorithm that can be implemented in practice, we approximate the labelled RFS mixture prior by a GLMB of the form (6). Applying the merged measurement likelihood function to a GLMB prior yields a posterior that is no longer a GLMB, but a labelled RFS mixture. However, by marginalising the joint densities post-update, we can approximate the posterior as a GLMB, so that a recursive filter may be implemented. Moreover, using a recent result on GLMB density approximation [21], it can be shown that our GLMB approximation preserves the first moment and cardinality distribution.

For a prior density in the form of a GLMB as in (6), applying the likelihood function (19) yields a posterior that is a labelled RFS mixture given by

$$\pi(\boldsymbol{X}|Z) \tag{31}$$
$$= \Delta(\boldsymbol{X}) \sum_{c\in\mathbb{C}} \sum_{\substack{\mathcal{U}(\boldsymbol{X})\in\mathcal{P}(\boldsymbol{X}) \\ \theta\in\Theta(\mathcal{U}(\mathcal{L}_{\boldsymbol{X}}))}} w_Z^{(c,\theta)}(\mathcal{U}(\mathcal{L}_{\boldsymbol{X}})) \left[p^{(c,\theta)}(\cdot|Z)\right]^{\mathcal{U}(\boldsymbol{X})}$$

where

$$w_Z^{(c,\theta)}(\mathcal{U}(L)) = \frac{w^{(c)}(L)\left[\eta_Z^{(c,\theta)}(\cdot)\right]^{\mathcal{U}(L)}}{\sum_{c\in\mathbb{C}}\sum_{J\subseteq\mathbb{L}}\sum_{\substack{\mathcal{U}(J)\in\mathcal{P}(J) \\ \theta\in\Theta(\mathcal{U}(J))}} w^{(c)}(J)\left[\eta_Z^{(c,\theta)}(\cdot)\right]^{\mathcal{U}(J)}} \tag{32}$$

$$p^{(c,\theta)}(\boldsymbol{Y}|Z) = \frac{\check{p}^{(c)}(\boldsymbol{Y})\,\check{\psi}_Z(\boldsymbol{Y};\theta)}{\eta_Z^{(c,\theta)}(\mathcal{L}_{\boldsymbol{Y}})} \tag{33}$$

$$\check{p}^{(c)}(\boldsymbol{Y}) = \left[p^{(c)}(\cdot)\right]^{\boldsymbol{Y}} \tag{34}$$

$$\eta_Z^{(c,\theta)}(L) = \left\langle \check{p}_{l_{1:|L|}}^{(c)}(\cdot), \left(\check{\psi}_Z\right)_{l_{1:|L|}}(\cdot;\theta)\right\rangle \tag{35}$$

and $\check{\psi}_Z(\boldsymbol{Y};\theta)$ is given by (18). The derivation of this is given in the appendix. In order to bring the posterior density back to the required GLMB form, we marginalise the joint densities within each component of the labelled RFS mixture as follows

$$p^{(c,\theta)}(\boldsymbol{Y}|Z) \approx \left[\tilde{p}^{(c,\theta)}(\cdot|Z)\right]^{\boldsymbol{Y}},$$
$$\tilde{p}^{(c,\theta)}(x_i,l_i|Z) = \int p_{l_{1:|L|}}^{(c,\theta)}\left(x_{1:|L|}|Z\right) d\left(x_{1:i-1}, x_{i+1:|L|}\right).$$

This allows us to make the following approximation, since the set of joint densities representing the groups in partition $\mathcal{U}(\boldsymbol{X})$ reduces to a set of independent densities representing the targets in $\boldsymbol{X}$,

$$\left[p^{(c,\theta)}(\cdot|Z)\right]^{\mathcal{U}(\boldsymbol{X})} \approx \left[\tilde{p}^{(c,\theta)}(\cdot|Z)\right]^{\boldsymbol{X}}.$$

This leads to the following GLMB approximation to the multi-object posterior

$$\pi(\boldsymbol{X}|Z) \tag{36}$$
$$\approx \Delta(\boldsymbol{X}) \sum_{c\in\mathbb{C}} \sum_{\substack{\mathcal{U}(\boldsymbol{X})\in\mathcal{P}(\boldsymbol{X}) \\ \theta\in\Theta(\mathcal{U}(\mathcal{L}_{\boldsymbol{X}}))}} w_Z^{(c,\theta)}(\mathcal{U}(\mathcal{L}_{\boldsymbol{X}})) \left[\tilde{p}^{(c,\theta)}(\cdot|Z)\right]^{\boldsymbol{X}}.$$

It is clear that computing this posterior will still be very computationally demanding, since it consists of a triple-nested sum over prior components, partitions of the target set, and measurement-to-group associations. In the following subsections, we propose two methods of truncating this sum to achieve an efficient implementation of the filter.

A. Partitioning with Constrained Measurement Assignment

The most obvious way to compute the terms in (36) is to list all possible partitions of the target set, and then generate assignments of measurements to target groups within each partition. It is clear that this procedure this will be computationally prohibitive in cases with a large number of targets, due to the exponentially increasing number of partitions. However, it is possible to remove many of the unlikely partitions by exploiting two pieces of information; the cluster structure of the target set, and a model of the sensor's resolution capabilities. Target clustering can be exploited by constructing disjoint subsets of the targets that are known to be well separated, and which have no possibility of sharing measurements. An independent filter is executed for each cluster, and since each of these filters only needs to partition a smaller set of targets, this can lead to a significant reduction in the total number of partitions to be computed [24], [25].

A method of further reducing the partitions for each cluster is to use a sensor resolution model to remove those that are unlikely to occur. Various resolution models have been proposed in the literature in order to approximate the probability of a resolution event (or equivalently, a partition of the targets) [2], [5], [4]. It would be possible to use such a model to eliminate unlikely partitions, by only retaining those that exceed some minimum probability threshold. In this paper, we use a simpler model in order to eliminate the unlikely partitions, since our filter does not require the use of partition probabilities. Our model requires two parameters; the maximum spread of targets in an unresolved group in the measurement space (d_{max}^t), and the minimum distance between centroids of unresolved groups in the measurement space (d_{min}^g). A partition $\mathcal{U}(\boldsymbol{X})$ is considered infeasible if either of the following is true,

$$D_{min}\left(\left\{\bar{Y}; Y\in\mathcal{U}(\boldsymbol{X})\right\}\right) < d_{min}^g \tag{37}$$
$$\max\left(\left\{D_{max}(Y); Y\in\mathcal{U}(\boldsymbol{X})\right\}\right) > d_{max}^t \tag{38}$$

where $\bar{Y}$ is the mean of the set Y, $D_{min}(Y)$ is the minimum pairwise distance between points in Y, and $D_{max}(Y)$ is the maximum pairwise distance between points in Y. This model can be applied separately to each measurement dimension, and a partition is rejected if condition (37) is satisfied in all dimensions or condition (38) is satisfied in any dimension. Intuitively, this model enforces the constraint that any unresolved target group cannot be spread over a large distance, and resolved groups cannot appear too close together. By applying these constraints, the unlikely partitions are removed, thereby allowing the filter to devote a larger proportion of its computational resources to the more likely partitions.

Having generated the feasible partitions, we can now proceed to compute the terms of the posterior GLMB density. In a manner similar to that of the standard GLMB filter [17], a ranked set of alternative measurement assignments is generated using Murty's algorithm [26]. The difference here is that instead of assigning measurements to individual targets, we now assign the measurements to groups of targets within each partition. We refer to this version of the algorithm as the GLMB-MP filter, and a more detailed account of the update routine is given in Section V-C.

B. Relaxed Measurement Assignment

The previous section described a method of generating highly weighted components from the posterior, however, it is very computationally demanding due to the need to

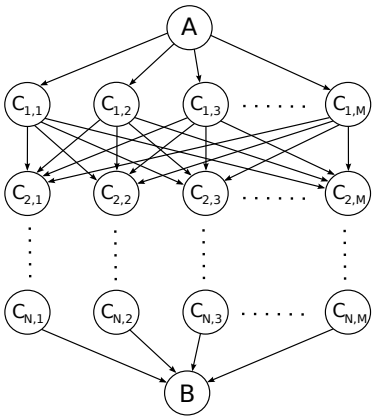


Figure 2. Graph structure for k-shortest path algorithm. Rows correspond to targets and columns correspond to measurements. Each path from A to B represents a solution to a relaxed assignment problem, in which a measurement may be assigned to more than one target, but each target can only be assigned a single measurement. Each solution can also be viewed as a one-to-one mapping of measurements to groups, with the difference being that the grouping is not pre-specified, but arises naturally due to the association of multiple targets to a single measurement.

compute ranked assignments for every feasible partition of every component in the prior. We now propose a cheaper method of generating the posterior GLMB components, which can potentially alleviate this bottleneck.

Let us neglect for a moment the fact that measurements are generated by groups of targets, and instead, assume that targets give rise to measurements independently, but more than one target may generate the same measurement. Although this seems like a contradiction, the effect of this is mitigated by the fact the component weight calculation is still carried out exactly, so in principle it can achieve similar performance to the partitioned method, provided that enough components are generated.

The likelihood of assigning a measurement to a target is based only on that target's prior, which we capture in a cost matrix consisting of one row per target and one column per measurement. We also allow for misdetections by appending one additional column containing the target misdetection costs. A directed graph is constructed using this matrix, in which each element represents a node, and each node has a set of edges connecting it to every node in the following row, as shown in Figure 2. An entry node is placed before the first row (node A), and an exit node after the last row (node B). The cost associated with each edge is the value of the node that the edge points to (the cost of the edges pointing to node B is zero). Using this graph, a ranked list of feasible assignments can be obtained using a k-shortest paths algorithm [28].

It is important to note that due to our target independence assumption, the solutions generated by this method will not be in decreasing order of posterior weight. However, under common conditions, we can expect this method to generate components with significant weights, with the advantage that it is computationally much cheaper than the partitioning approach. We refer to this algorithm as the GLMB-MR filter, and more implementation details are given in Section V-D.

V. Implementation

In this section we describe the steps required to implement both the GLMB-MP and GLMB-MR filters; birth density generation, prediction, update, estimate extraction, and pruning. For the update step, there are two interchangable methods, the choice of which will depend on the scenario and the available computational resources. Throughout the pseudo-code, the symbols Φ, Ψ, or Ω are used to represent a complete GLMB density, which is a data structure comprising the following four items:

- X: collection of single target pdfs,
- L: collection of target labels,
- w: component weights,
- ρ: cardinality distribution.

For a given density Φ, subsets of X and L are referenced by the following notation:

- $\Phi.X^{(n)}$: set of GLMB components of cardinality n,
- $\Phi.X^{(n,i)}$: set of target pdfs in the i-th component of cardinality n,
- $\Phi.X^{(n,i,j)}$: pdf of the j-th target in the i-th component of cardinality n.

The same notation that applies to X also applies to L, except that L contains an integer label for each target instead of a pdf. The weights w follow a similar notation, where $\Phi.w^{(n)}$ is the set of weights of components with cardinality n, and $\Phi.w^{(n,i)}$ is the weight of the i-th component of cardinality n. The cardinality distribution ρ is a vector of length $|\Phi.X|$, in which the n-th element, denoted $\rho^{(n)}$, is the probability that there are n targets present.

A. Step 1: Target Birth Model

In a general multi-target tracking scenario, targets may appear anywhere within the sensor's detection region. It is therefore important that the target birth model not restrict the locations where new tracks may be initiated. A general measurement driven birth model would allow any number of measurements on any scan to initiate new tentative tracks. Since each term in the GLMB needs to capture a complete set of hypothesised births, this approach is computationally very expensive, since it requires enumerating all possible subsets of the measurements on each scan. Here, we make a few reasonable assumptions in order to reduce this computation to a more managable level.

1) A maximum of one new target can appear on each scan.
2) Measurements that are likely to have originated from existing targets do not give rise to new birth terms.
3) New targets are always detected on their first two scans.

Applying these assumptions reduces the number of terms in the birth density, which in turn significantly reduces the number of components in the prediction. Pseudo-code for generating the birth density is given in Figure 3, where the following functions are used:

- `TargetPrior`(z): Generate a prior pdf for a single target given a single measurement z.
- `Predict`(X): Propagate the target pdf X from the time of the previous scan to the current time.

- `ValidationGate` (X): Returns the measurement validation region for a single target pdf X.

and the `Normalize` function is defined in Figure 4.

Input: $Z_{k-1}, P_{A,k-1}, Z_k, P_{birth}, l_{next}$
Output: Ω, l_{next}

$c = 1$;
for $i = 1 : |Z_{k-1}|$ **do**
  if $P^{(i)}_{A,k-1} < P_{thresh}$ **then**
    $X =$ `TargetPrior` $\left(z_{k-1,i}\right)$;
    $X_+ =$ `Predict` (X);
    $V =$ `ValidationGate` (X_+);
    `keep` $=$ `false`;
    for $j = 1 : |Z_k|$ **do**
      if $\boldsymbol{z}_{k,j} \in V$ **then**
        `keep` $=$ `true`;
    if `keep` **then**
      $\Omega.X^{(1,c,1)} = X_+$;
      $\Omega.L^{(1,c,1)} = l_{next}$;
      $w^{(1,c)} = P_{birth}$;
      $c = c + 1$;
      $l_{next} = l_{next} + 1$;
$(\Omega.w, \Omega.\rho) =$ `Normalise` (w);

Figure 3. Pseudo-code for `Birth` function. Input parameters; Z_{k-1} (previous measurement set), $P_{A,k-1}$ (probabilities that each previous measurement is from a target), P_{thresh} (maximum value of $P_{A,k-1}$to generate birth), Z_k (current measurement set), P_{birth} (probability that an unassigned measurement is from a new target), l_{next} (next target label). Output parameters; Ω (GLMB birth density), l_{next} (next target label).

Input: w
Output: w_*, ρ

$C = 0$;
for $i = 1 : |w|$ **do**
  $\rho^{(i)} = \sum_{j=1}^{|w^{(i)}|} w^{(i,j)}$;
  $w_*^{(i,\cdot)} = w^{(i,\cdot)}/\rho^{(i)}$;
  $C = C + \rho^{(i)}$;
$\rho^{(\cdot)} = \rho^{(\cdot)}/C$;

Figure 4. Pseudo-code for `Normalise` function. Input parameter; w (GLMB component weights). Output parameters; w_* (normalised GLMB component weights), ρ (cardinality distribution).

B. Step 2: Survival Prediction

The prediction for the surviving target density is the same as for the standard GLMB filter in [18]. This step takes the retained components of the multi-object density from the previous time, and uses them to obtain the N-best components of the prior multi-object density at the current time. This is done by iterating through the components of the previous density, generating a set of highly weighted predicted components as we go. The weight of a predicted component is determined by the product of the survival/termination probabilities of its constituent objects, and we may obtain a list of predicted components in decreasing order of weight using k-shortest paths [28].

The k-shortest paths method is applied to each component as follows. First, we generate an $n \times 2$ matrix, where n is the number of targets in the component under consideration. The first column is populated with target survival probabilities, and the second column with target death probabilies. The negative logarithm of this matrix is taken to change the problem from maximum-product form into minimum-sum form:

$$C = -\log \begin{pmatrix} p_S\left(X^{(n,i,1)}\right) & 1 - p_S\left(X^{(n,i,1)}\right) \\ \vdots & \vdots \\ p_S\left(X^{(n,i,n)}\right) & 1 - p_S\left(X^{(n,i,n)}\right) \end{pmatrix}. \quad (39)$$

This matrix is used to generate a graph of the same basic structure as Figure 2, for which we find the k-shortest paths between nodes A and B. The rows in which the first column was visited correspond to predicted target survivals, and rows in which the second column was visited correspond to predicted target deaths. The surviving targets are then collected into a new component, with their pdfs propagated to the current time using the single target transition kernel. The new component weight is calculated by multiplying the previous weight by e^{-c}, where c is the cost of the path used to generate that component. Pseudo-code for the prediction of the surviving target density is given in Figure 5, in which the following functions are used:

- `Pois` (y, λ): Poisson pdf with mean λ computed at each element of the array y.
- `Allocate` (n, w): Randomised proportional allocation of a scalar number of objects n, into a fixed number of bins with weights given by the array w.
- `CostMatrixPredict` $(\boldsymbol{X}, p_s)$: Compute cost matrix for target set $\boldsymbol{X}$ and survival probability function $p_s(\cdot)$, using equation (39).
- `ShortestPaths` (C, k): List the k cheapest path-cost pairs in increasing order of cost, based on cost matrix C.

Input: Φ, N, p_s
Output: Ω

$\lambda =$ `Mean` $(\Phi.\rho)$;
$\mu = |\Phi.X|$;
$M_{1:\mu} =$ `Allocate` $(N,$ `Pois` $(1:\mu;\lambda))$;
$k =$ `Ones` $(1, \mu)$;
for $n = 1 : \mu$ **do**
  $N_{n,1:|\Phi.X^{(n)}|} =$ `Allocate` $\left(N_n, \Phi.w^{(n)}\right)$;
  for $i = 1 : \left|\Phi.X^{(n)}\right|$ **do**
    $C =$ `CostMatrixPredict` $\left(\Phi.X^{(n,i)}, p_s\right)$;
    $P =$ `ShortestPaths` $(C, N_{n,i})$;
    for $(p, c) \in P$ **do**
      $s = \{j; p(j) = 1\}$;
      $m = |s|$;
      $l = 1 : m$;
      $\Omega.X^{(m,k(m),l)} =$ `Predict` $\left(\Phi.X^{(n,i,s(l))}\right)$;
      $\Omega.L^{(m,k(m),l)} = \Phi.L^{(n,i,s(l))}$;
    $w^{(m,k(m))} = \Phi.w^{(n,i)} \times \Phi.\rho^{(n)} \times e^{-c}$;
    $k(m) = k(m) + 1$;
$(\Omega.w, \Omega.\rho) =$ `Normalise` (w);

Figure 5. Pseudo-code for `PredictSurvivals` function. Input parameters; Φ (posterior GLMB density from previous time step), N (number of predicted components to generate), p_S (target survival probability). Output parameter; Ω (predicted GLMB density of surviving targets).

C. Step 3: Partitioning with Constrained Assignment

The update routine for the GLMB-MP filter involves generating a set of posterior components for each component in the prior, by enumerating the feasible partitions and measurement-to-group assignments. Note that this procedure can be easilly parallelised, since there is no dependency between prior components. The first step is to decide how many posterior components to generate for each prior component. We do this by allocating a number of components N_n to each cardinality c, based on a Poisson distribution centered around the mean prior cardinality. This ensures that enough components are generated for a variety of cardinalities, to allow for target births and deaths. For each cardinality, we then split up N_n among the components of cardinality n, where the number $N_{n,i}$ allocated to component i is proportional to its prior weight. Since it is usually not possible to do this exactly for integer numbers of components, we use a randomised proportional allocation algorithm for this purpose.

For each component, we compute the feasible partitions of the target set using the constraints (37) and (38). The number of allocated components $N_{n,i}$ is then divided equally among these partitions, where the number of components allocated to partition $\mathcal{U}$ of the i-th component of cardinality n is denoted $N_{n,i,\mathcal{U}}$. We then compute an assignment cost matrix for each feasible partition.

Let $\boldsymbol{X}$ be a set of targets and $\mathcal{U}(\boldsymbol{X})$ a partition of those targets. Each row of the cost matrix corresponds to an element of $\mathcal{U}(\boldsymbol{X})$ (a target group), and each column corresponds to a measurement, or a misdetection. The cost matrix is of the form $C = [D; M]$, where D is a $|\mathcal{U}(\boldsymbol{X})| \times |Z|$ matrix and M is a $|\mathcal{U}(\boldsymbol{X})| \times |\mathcal{U}(\boldsymbol{X})|$ matrix with elements

$$D_{i,j} = -\log\left(\frac{p_D(\mathcal{U}^{(i)}(\boldsymbol{X}))\breve{y}\left(z_j|\mathcal{U}^{(i)}(\boldsymbol{X})\right)}{\kappa(z_j)}\right) \tag{40}$$

$$M_{i,j} = \begin{cases} -\log\left(q_D(\mathcal{U}^{(i)}(\boldsymbol{X})\right), & i = j \\ \infty, & i \neq j \end{cases} \tag{41}$$

where $\mathcal{U}^{(i)}(\boldsymbol{X})$ denotes the i-th group in partition $\mathcal{U}(\boldsymbol{X})$. We now run Murty's ranked assignment algorithm [26] on the cost matrix C to obtain a list of the cheapest $N_{c,k,\mathcal{U}}$ measurement-to-group assignments in increasing order of cost [2]. For each assignment, we create a new posterior component, consisting of one updated joint density for each target group according to (33), (35) and (18). Finally, the posterior density is approximated in the form of a GLMB by computing the marginal distribution for each target, within each posterior component. Figure 6 shows pseudo-code for the partitioned update routine, where the following new functions are used:

- `Partition` $(\boldsymbol{Y})$: List partitions of the set $\boldsymbol{Y}$.
- `ConstrainPartitions` $(P, d^g_{min}, d^t_{max})$: Remove infeasible partitions in P, according to (37)-(38).
- `CostMatrixGrp` $(Z, \boldsymbol{X}, \mathcal{U})$: Compute cost matrix for measurement set Z and target set $\boldsymbol{X}$ grouped according to partition $\mathcal{U}$, using (40)-(41).
- `Murty` (C, n): Use Murty's algorithm to list the top n ranked assignments based on cost matrix C.
- `JointDensity` $(\boldsymbol{X})$: Join the pdfs in the set $\boldsymbol{X}$ to create a joint density.
- `MeasUpdate` (Y, z): Update the prior density Y using measurement z.
- `Marginal` (Y, i): Compute the marginal distribution of the i-th target from the joint density Y.
- `SumWeightSubsets` (w, θ): For each set of component indices in θ, add up the corresponding weights in w.

```
Input: Φ, Z, N, λ, V, d^g_min, d^t_max
Output: Ω, P_A
ρ̄ = Mean(Φ.ρ);
N_{1:|Φ.X|} = Allocate(N, Pois(0 : |Φ.X|, ρ̄));
χ(i) = ∅, ∀i ∈ {1, .., |Z|};
for n = 1 : |Ω.X| do
    j = 1;
    N_{n,1:|Φ.X^(n)|} = Allocate(N_n, Φ.w^(n));
    for i = 1 : |Φ.X^(n)| do
        P = Partition(Φ.X^(n,i));
        P = ConstrainPartitions(P, d^g_min, d^t_max);
        N_{n,i,1:|P|} = Allocate(N_{n,i}, Ones(1, |P|)/|P|);
        for U ∈ P do
            C = CostMatrixGrp(Z, Φ.X^(n,i), U);
            A = Murty(C, N_{n,i,U});
            for (a, c) ∈ A do
                t = 1;
                w^(n,j) = ρ^(n) λ^|Z| V^-1 e^-c;
                for G ∈ U do
                    Y = JointDensity(Φ.X^(n,i,G));
                    Y_L = Φ.L^(n,i,G);
                    if a(G) > 0 then
                        Y = MeasUpdate(Y, z_a(G));
                        χ(a(G)) = {χ(a(G)), (n, j)};
                    for k = 1 : |G| do
                        Ω.X^(n,j,t) = Marginal(Y, k);
                        Ω.L^(n,j,t) = Y_L^(k);
                        t = t + 1;
                j = j + 1;
(Ω.w, Ω.ρ) = Normalise(w);
P_A = SumWeightSubsets(Ω.w, χ);
```

Figure 6. Pseudo-code for `PartitionedUpdate`. Input parameters; Φ (prior GLMB density), Z (measurement set), N (number of posterior components to generate), λ (clutter intensity), V (measurement space volume), d^g_{min} (minimum spacing between group centroids), d^t_{max} (maximum group spread) . Output parameters; Ω (posterior GLMB density), P_A (probabilities that each measurement is from a target).

D. Alternative Step 3: Relaxed Assignment

As in the partitioned update routine, we generate a set of posterior components for each component in the prior. However, in the GLMB-MR filter, we use a different means of choosing the components, leading to a cheaper algorithm which is scalable to a larger number of targets per cluster. Here we forego the enumeration of target partitions and Murty-based ranked assignment, in favour of a relaxed assignment of measurements to targets. This means that a single measurement may be assigned to multiple targets simultaneously,

[2]In our implementation, we have used the auction algorithm [27] to solve the optimal assignment problem within each iteration of Murty's algorithm.

but each target can still only receive one measurement. This leads to a simpler and computationally cheaper algorithm, but as one would expect, the performance is not as good as the partitioning approach.

For each prior component, the cost matrix consists of one row for each target, and one column for each measurement, with one extra column for misdetections. Thus, the cost matrix is of the form $C = [D; M]$, where D is a $|\boldsymbol{X}| \times |Z|$ matrix and M is a $|\boldsymbol{X}| \times 1$ matrix with elements

$$D_{i,j} = -\log\left(\frac{p_D(\boldsymbol{X}^{(i)})g\left(z_j|\boldsymbol{X}^{(i)}\right)}{\kappa\left(z_j\right)}\right) \tag{42}$$

$$M_{i,1} = -\log\left(q_D(\boldsymbol{X}^{(i)}\right) \tag{43}$$

The k-shortest paths algorithm is applied to this matrix to select components for inclusion in the posterior GLMB. The component weights are computed in the same way as in the partitioned update. However, instead of performing joint updates for each group followed by marginalisation, we approximate the marginal posterior for each target as the independently updated prior. This avoids the computational bottleneck of performing a large number of joint multi-target updates (one for every possible target grouping). Instead, only a single independent update needs to be performed for each unique single target density contained in the prior GLMB. Pseudo-code is provided in Figure 7, where

- $\texttt{CostMatrixIndep}\left(Z, \boldsymbol{X}\right)$: Compute cost matrix for measurement set Z and target set $\boldsymbol{X}$, assuming independent measurements, i.e. using (42) and (43).
- $\texttt{Unique}\left(y\right)$: List the unique elements of the array y.

E. Step 4: Estimate Extraction

Labelled target estimates are extracted from the posterior GLMB density by finding the maximum a-posteriori estimate of the cardinality, and then finding the highest weighted component with that cardinality. Pseudo-code for this is given in Figure 8, where τ_X is the list of estimated tracks, and τ_L is the list of corresponding labels.

F. Step 5: Pruning

To make the algorithm computationally efficient, we need to prune the posterior density after each update so that insignificant components do not consume computational resources. A component is retained if it is in the top N_{cmin} components for its cardinality, or if it is in the top N_{tot} overall and its ratio to the maximum weighted component is less than a threshold R_{max}. Pseudo-code is given in Figure 9, where

- $[c, i] = \texttt{Sort}\left(w, \rho\right)$: Sort components by their absolute weight, which for the i-th component of cardinality n is given by $w^{(n,i)} \times \rho^{(n)}$. The sorted components are indexed by their cardinality c, and their position i.

G. Complete Algorithm

The last function remaining to be defined is that which multiplies the birth and surviving GLMB densities to arrive at the overall predicted density. Pseudo-code for this function is given in Figure 10. The top level recursion for the GLMB-M filter is given in Figure 11.

Input: Φ, Z, N, λ, V
Output: Ω, P_A
$\bar{\rho} = \texttt{Mean}\left(\Phi.\rho\right)$;
$N_{1:|\Phi.X|} = \texttt{Allocate}\left(N, \texttt{Pois}\left(0 : |\Phi.X|, \bar{\rho}\right)\right)$;
$\chi\left(i\right) = \emptyset, \forall i \in \{1, .., |Z|\}$;
for $n = 1 : |\Phi.X|$ **do**
  $j = 1$;
  $N_{n,1:|\Phi.X^{(n)}|} = \texttt{Allocate}\left(N_n, \Phi.w^{(n)}\right)$;
  for $i = 1 : |\Phi.X^{(n)}|$ **do**
    $C = \texttt{CostMatrixIndep}\left(Z, \Phi.X^{(n,i)}\right)$;
    $A = \texttt{ShortestPaths}\left(C, N_{n,i}\right)$;
    for $(a, c) \in A$ **do**
      $t = 1$;
      $w^{(n,j)} = \rho^{(n)}\lambda^{|Z|}V^{-1}$;
      for $m \in \texttt{Unique}\left(a\right)$ **do**
        $G = \{k : a(k) = m\}$;
        $Y = \texttt{JointDensity}\left(\Phi.X^{(n,i,G)}\right)$;
        if $m > 0$ **then**
          $\chi\left(m\right) = \{\chi\left(m\right), (n, j)\}$;
          $w^{(n,j)} = w^{(n,j)}P_D\left(Y\right)g\left(\boldsymbol{z}_m|Y\right)$;
        else
          $w^{(n,j)} = w^{(n,j)}\left(1 - P_D\left(Y\right)\right)$;
        for $k = 1 : |G|$ **do**
          $\Omega.X^{(n,j,t)} = \texttt{MeasUpdate}\left(\Phi.X^{(n,i,G(k))}, z_m\right)$;
          $\Omega.L^{(n,j,t)} = \Phi.L^{(n,i,G(k))}$;
          $t = t + 1$;
      $j = j + 1$;
$(\Omega.w, \Omega.\rho) = \texttt{Normalise}\left(w\right)$;
$P_A = \texttt{SumWeightSubsets}\left(\Omega.w, \chi\right)$;

Figure 7. Pseudo-code for `RelaxedUpdate` function. Input parameters; Φ (prior GLMB density), Z (measurement set), N (number of posterior components to generate), λ (clutter intensity), V (measurement space volume). Output parameters; Ω (posterior GLMB density), P_A (probabilities that each measurement is from a target).

Input: $\tau_X, \tau_L, \Omega, T$
Output: τ_X, τ_L
$n = \text{argmax}_n\left(\Omega.\rho^{(n)}\right)$;
$i = \text{argmax}_i\left(\Omega.w^{(n,i)}\right)$;
for $j = 1 : \left|\Omega.X^{(n,i)}\right|$ **do**
  $l = \Omega.L^{(n,i,j)}$;
  if $l \in \tau_L$ **then**
    $\tau_X\left(l\right) = \left[\tau_X\left(l\right); \left(T, \Omega.X^{(n,i,j)}\right)\right]$;
  else
    $\tau_L = [\tau_L; l]$;
    $\tau_X\left(l\right) = \left[\left(T, \Omega.X^{(n,i,j)}\right)\right]$;

Figure 8. Pseudo-code for `ExtractEstimates` function. Input parameters; τ_X (previous track estimates), τ_L (previous track labels), Ω (GLMB density), T (current time). Output parameters; τ_X (updated track estimates), τ_L (updated track labels).

VI. Simulation Results

In this section we test the performance of the two proposed versions of the GLMB-M filter on a simulated multi-target tracking scenario. The sensor generates noisy bearings-only measurements with false alarms and misdetections, and there are multiple crossing targets, each of which follows a white noise acceleration dynamic model. The sensor's motion comprises legs of constant velocity, interspersed with constant turn

Input: $\Phi, N_{tot}, N_{cmin}, R_{max}$
Output: Ω
$[C, I] = \texttt{Sort}(w, \rho)$;
$w_{min} = \Omega.w^{(C(1),I(1))}/R_{max}$;
$M_{tot} = 0$;
$M_{cdn} = \texttt{Zeros}(|\Phi.X|)$;
for $i = 1 : |C|$ **do**
  $c = C(i), s = I(i), m = M_{cdn}(c)$;
  $\texttt{keep} = (M_{tot} < N_{tot}) \times \left(\Phi.w^{(c,s)} > w_{min}\right)$
    $+ (m < N_{cmin})$;
  if keep **then**
    $\Omega.X^{(c,m)} = \Phi.X^{(c,s)}$;
    $\Omega.L^{(c,m)} = \Phi.L^{(c,s)}$;
    $\Omega.w^{(c,m)} = \Phi.w^{(c,s)}$;
    $M_{tot} = M_{tot} + 1$;
    $M_{cdn}(c) = m + 1$;
$(\Omega.w, \Omega.\rho) = \texttt{Normalise}(\Omega.w)$;

Figure 9. Pseudo-code for `Prune` function. Input parameters; Φ (GLMB density), N_{tot} (number of components to retain), N_{cmin} (minimum number to retain for any cardinality), R_{max} (maximum ratio of highest to lowest component weight). Output parameter; Ω (pruned GLMB density).

Input: Φ, Ψ
Output: Ω
$c = \texttt{Ones}(1, |\Phi.X| + |\Psi.X|)$;
for $m = 1 : |\Phi.X|$ **do**
  for $n = 1 : |\Psi.X|$ **do**
    $k = m + n$;
    for $i = 1 : \left|\Phi.X^{(m)}\right|$ **do**
      for $j = 1 : \left|\Psi.X^{(n)}\right|$ **do**
        $\Omega.X^{(k,c(k))} = \left[\Phi.X^{(m,i)}; \Psi.X^{(n,j)}\right]$;
        $\Omega.L^{(k,c(k))} = \left[\Phi.L^{(m,i)}; \Psi.L^{(n,j)}\right]$;
        $\Omega.w^{(k,c(k))} = \Phi.w^{(m,i)} \times \Psi.w^{(n,j)}$;
        $c(k) = c(k) + 1$;
$(\Omega.w, \Omega.\rho) = \texttt{Normalise}(\Omega.w)$;

Figure 10. Pseudo-code for `Multiply` function. Input parameters; Φ (GLMB density), Ψ (GLMB density). Output parameter; Ω (multiplied GLMB density).

rate manoeuvres. These manoeuvres improve the observability over the course of the simulation.

The target state space is defined in terms of 2D Cartesian position and velocity vectors, i.e. for target t at time k, $x_k^{(t)} = \begin{bmatrix} p_{x,k}^{(t)} & p_{y,k}^{(t)} & \dot{p}_{x,k}^{(t)} & \dot{p}_{y,k}^{(t)} \end{bmatrix}^T$. All targets follow the dynamic model

$$x_{k+1} = Fx_k + \Gamma v_k \tag{44}$$

$$F = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \otimes I_2, \qquad \Gamma = \begin{bmatrix} T^2/2 \\ T \end{bmatrix} \otimes I_2 \tag{45}$$

where $T = 20s$ is the sensor sampling period, and $v_k \sim \mathcal{N}(0, Q)$ is a 2×1 independent and identically distributed Gaussian process noise vector with $Q = \sigma_v^2 I_2$ where the standard deviation of the target acceleration is $\sigma_v = 0.005m/s^2$.

We have simulated the measurements using two alternative methods; one based on a detection-level measurement model, and the other based on an image simulation followed by peak detection. The details of these models are given in the subsections to follow, and in both cases, the measurement

Input: $Z_{1:N}, N_{pred}, N_{up}, N_{ret}, N_{cmin}, R_{max}$
Output: τ_X, τ_L
$l_{next} = 1, \tau_X = \emptyset, \tau_L = \emptyset$;
$P_{A,1} = \texttt{Zeros}(1, |Z_1|)$;
for $k = 2 : N$ **do**
  $\Omega_b, l_{next} = \texttt{Birth}\left(Z_{k-1}, P_{A,k-1}, Z_k, l_{next}\right)$;
  if $k = 2$ **then**
    $\Omega_+ = \Omega_b$;
  else
    $\Omega_s = \texttt{PredictSurvivals}\left(\Omega_{k-1}, N_{pred}\right)$;
    $\Omega_+ = \texttt{Multiply}(\Omega_s, \Omega_b)$;
  if partitioned **then**
    $\Omega_k, P_{A,k} = \texttt{PartitionedUpdate}(\Omega_+, Z_k, N_{up})$;
  else
    $\Omega_k, P_{A,k} = \texttt{RelaxedUpdate}(\Omega_+, Z_k, N_{up})$;
  $(\tau_X, \tau_L) = \texttt{ExtractEstimates}(\tau_X, \tau_L, \Omega_k, T_k)$;
  $\Omega_k = \texttt{Prune}(\Omega_k, N_{ret}, N_{cmin}, R_{max})$;

Figure 11. GLMB-M filter top level recursion. Input parameters; $Z_{1:N}$ (measurements), N_{pred} (maximum predicted components), N_{up} (maximum updated components), N_{ret} (maximum retained components), N_{cmin} (minimum retained for any cardinality), R_{max} (maximum component weight ratio). Output parameters; τ_X (track estimates), τ_L (track labels).

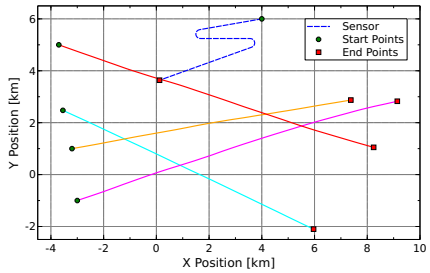
function is given by

$$h\left(x_k^{(t)}, x_k^{(s)}\right) = \arctan\left(\frac{p_{y,k}^{(t)} - p_{x,k}^{(s)}}{p_{y,k}^{(t)} - p_{x,k}^{(s)}}\right). \tag{46}$$

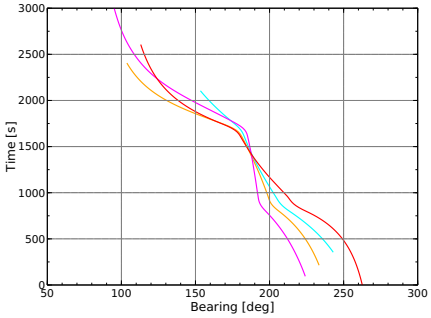
where $x_k^{(t)}$ is the target state vector, and $x_k^{(s)}$ is the sensor state vector. For this analysis, we model the pdf of each target using a single Gaussian, and the measurement updates are carried out using the extended Kalman filter (EKF). It is clearly possible to use more sophisticated non-linear filters, such as the unscented or cubature Kalman filter, or to model the single target pdfs using more accurate representations such as Gaussian mixtures or particle distributions. These techniques may lead to some performance improvement, but this analysis is beyond the scope of this paper.

The scenario consists of four targets, with geometry as shown in Figure 12a. One target is present at the beginning, with another three arriving during the first 400 seconds, and three terminating during the final 900 seconds. The bearings of all four targets cross one another in the middle of the scenario, which can be seen in Figure 12b. During this time there will be merged measurements present.

In all cases, the filter initialises the pdf of each newborn target as a Gaussian with mean range of 10km from the sensor, range standard deviation of 3km, velocity of 5m/s moving directly towards the sensor along the line of bearing, course standard deviation of 50°, and speed standard deviation of 2m/s. The filters generate a maximum of $N_{up} = 10000$ components in the GLMB density during the update, which is reduced to $N_{ret} = 1000$ (with $N_{cmin} = 10$, and $R_{max} = 10^8$) during the pruning step. All filters were implemented in C++, and executed on an Intel Core i7 2600 processor. We have not parallelised the generation of components in our implementation, hence it would be possible significantly speed up the execution time by doing so.



(a) Target-observer geometry



(b) True target bearings

Figure 12. Simulated multi-target scenario

A. Detection-level Simulation Model with Merging

Let $\mathbb{X}$ be the single-object state space, and $\mathbb{Z}$ the sensor measurement space. At time index k, consider a single sensor with known control input u_k, and let $\boldsymbol{X}_k$ represent the unknown multi-target state. Let $h(\cdot, u_k) : \mathbb{X} \times \mathbb{L} \to \mathbb{Z}$ be the measurement function, $C = \{c_1, c_2, ...c_{N_C}\}$ be a set of disjoint cells covering $\mathbb{Z}$ such that $\bigcup_{i=1}^{N_C} c_i = \mathbb{Z}$ and $c_i \cap c_j = \emptyset, \forall i \neq j$, and let $\boldsymbol{T}_k^{(i)}$ be the set of targets whose true state falls in cell i at time k,

$$\boldsymbol{T}_k^{(i)}(\boldsymbol{X}) = \{\boldsymbol{x} \in \boldsymbol{X} : h(\boldsymbol{x}, u_k) \in c_i\}. \tag{47}$$

Then cell i produces the following target measurement

$$Z_k^{(i)} = \begin{cases} \frac{1}{\left|\boldsymbol{T}_k^{(i)}(\boldsymbol{X})\right|} \sum\limits_{\boldsymbol{x} \in \boldsymbol{T}_k^{(i)}} h(\boldsymbol{x}, u_k) + w_k, & \left|\boldsymbol{T}_k^{(i)}(\boldsymbol{X})\right| > 0 \\ \emptyset, & \left|\boldsymbol{T}_k^{(i)}(\boldsymbol{X})\right| = 0 \end{cases} \tag{48}$$

with probability $p_D\left(\boldsymbol{T}_k^{(i)}(\boldsymbol{X})\right)$, and $Z_k^{(i)} = \emptyset$ with probability $q_D\left(\boldsymbol{T}_k^{(i)}(\boldsymbol{X})\right) = 1 - p_D\left(\boldsymbol{T}_k^{(i)}(\boldsymbol{X})\right)$. In the above, w_k is an independent measurement noise vector. The set of target generated measurements at time k is thus $D_k = \bigcup_{i=1}^{N_C} Z_k^{(i)}$. In addition, at each time k, the sensor generates an RFS of false measurements denoted K_k, with a Poisson cardinality distribution of known mean. The spatial locations of the points in K_k are simulated by repeatedly generating a random point within a randomly chosen unoccupied cell, such that any given cell cannot contain more than one measurement. The overall measurement set is thus given by $Z_k = D_k \cup K_k$.

This model allows us to explicitly control the measurement noise, clutter rate, and resolution, such that the filter parameters can be matched to their true values. We have tested the algorithms under three bearing cell widths; 1°, 2° and 4°. The measurement noise is dependent on the number of unresolved targets in the group, and is set according to Table I.

Table I
MEASUREMENT NOISE SETTINGS

Cell Width	σ_w		
	1 Target	2 Targets	>2 Targets
1°	0.25°	0.375°	0.5°
2°	0.5°	0.75°	1°
4°	1°	1.5°	2°

In these simulations, the detection probability is 0.98, and the mean number of clutter measurements is 30. To analyse the relative performance of the algorithms, we use the Optimal Sub-pattern Assignment (OSPA) distance, with a cutoff of 1km, and order parameter of 2. Figures 13, 14 and 15 show the average OSPA distance and algorithm execution times (over 200 Monte Carlo runs) for the standard GLMB, GLMB-MP, and GLMB-MR filters for the three different cell widths. Also shown are the results of running the standard GLMB filter on measurements with perfect resolution and the same noise level as fully resolved targets in each case.

As one would expect, the standard GLMB filter performs poorly when processing the finite resolution measurements, since it cannot account for measurement merging. When the target bearings begin to cross each other, we see a significant jump in OSPA for the standard GLMB. The reason for this is clear, the filter can only explain the missing measurements as misdetections or target deaths. Since the probability of detection is high, the probability of having several misdetections in a row is very small. Hence the filter believes that targets have died and terminates the tracks, which is also the reason for the associated drop in execution time. When the targets become resolved again, the filter must initiate new tracks (with default prior state), since it has no way of continuing a track that has been previously terminated. For this reason, the standard filter will take a long time to recover from dropped tracks, especially in situations when observability is a factor.

Also as expected, the GLMB-MP filter performs best in all three cases, because by enumerating the partitions of the target set in advance, it is able to obtain better posterior components than the GLMB-MR filter. The drawback of this algorithm is that when the targets get close to each other, the execution time rises dramatically due to the increasing number of feasible partitions. As the cell width increases, the width of the peak in the execution time also increases, because the targets are unresolved for a longer period.

The GLMB-MR filter clearly has higher error, however, it does not suffer from spikes in execution time. In all three cases, the peak execution time of the GLMB-MR is less than 20% of that for the GLMB-MP. This is due to the fact that the latter does a more expensive ranked assignment algorithm, and needs to carry out many group measurement updates for different partitions of the target set, each of which involves higher dimensional matrices. In contrast, the GLMB-MR only performs a single measurement update for each target, and a cheaper assignment based on k-shortest paths. Hence the

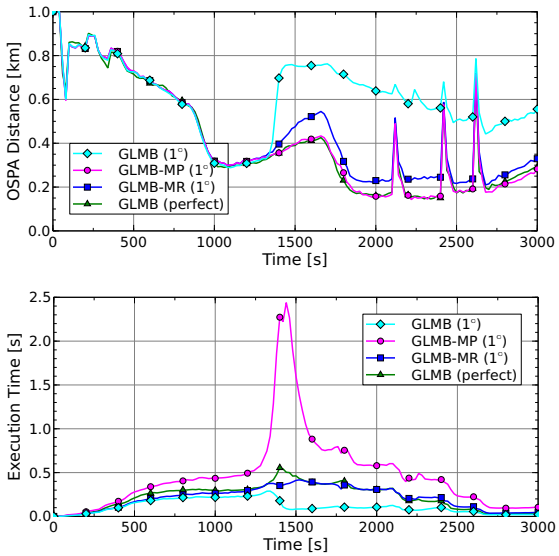


Figure 13. OSPA distance and execution time: 1 degree resolution

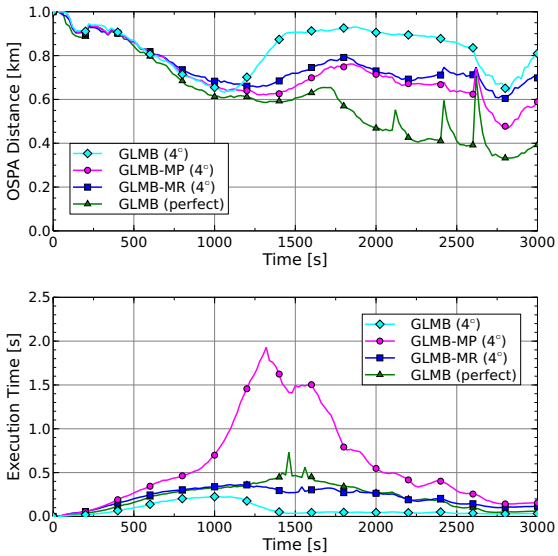


Figure 15. OSPA distance and execution time: 4 degree resolution

GLMB-MR has a relatively consistent run time, similar to that of the standard GLMB filter.

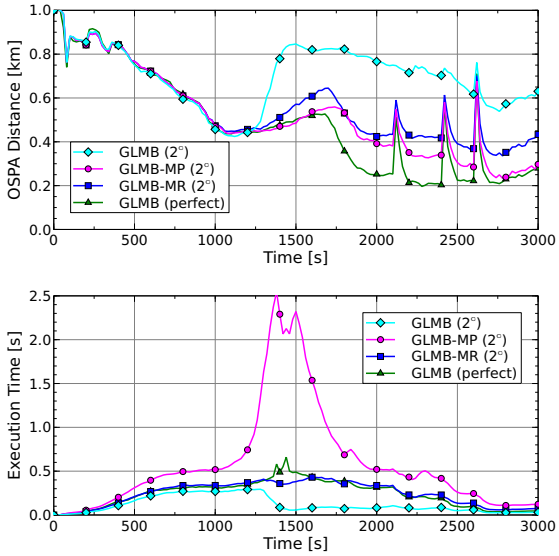


Figure 14. OSPA distance and execution time: 2 degree resolution

B. Image-based Simulation Model

The simulation model in the previous section allowed us to explicitly control the clutter rate and measurement noise parameters via simulation at the detection level. In this section we apply the merged measurement filters to data generated through a more realistic simulation model, in which we generate a sensor image, and use a simple peak detector to extract the point measurements. The image is simulated by first generating Rayleigh distributed background noise in each cell. A point spread function is applied to each target to obtain the mean signal amplitude in each cell, and a Ricean distributed signal amplitude is then generated for every cell. This is repeated for all targets, and the signal amplitudes added together in each cell. To generate detections, the vector of cell amplitudes is interpolated using a spline function, and the locations of all peaks that exceed a threshold are extracted. Under this model, the combined effect of the cell width and the point spread function control the distance at which targets become unresolvable. Figure 16 shows an example of the image generated from this model, with cell width of 1°, average signal to noise ratio of 5dB, and a Gaussian point spread function with standard deviation of 0.5°.

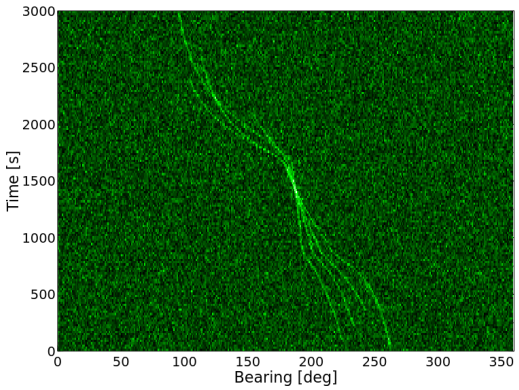


Figure 16. Simulated time-bearing image with 1° cell width

Figures 17 and 18 show the average OSPA distance over 200 runs for the standard GLMB, GLMB-MP and GLMB-MR filters, for the case of 1° and 2° cell widths. These results follow the same trend that was observed under the detection-level simulation model. The standard GLMB filter fails due to dropped tracks, and the GLMB-MP generally outperforms the GLMB-MR, albeit with a higher computational cost.

VII. Conclusion

We have proposed an algorithm for multi-object tracking in the presence of merged measurements, using the framework of labelled random finite sets. The algorithm is a generalisation of

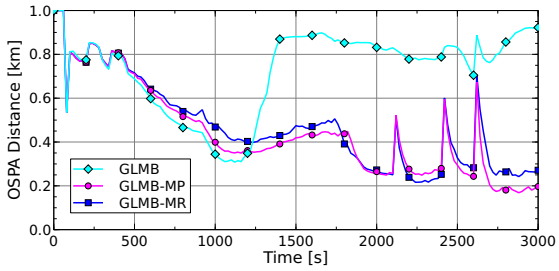


Figure 17. OSPA distance for image-based simulation with 1° cell width

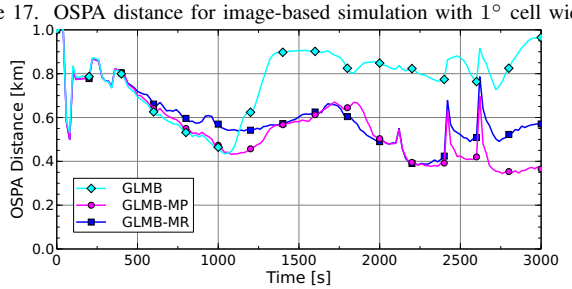


Figure 18. OSPA distance for image-based simulation with 2° cell width

the GLMB filter, originally presented in [17] for the standard sensor model without measurement merging. The exact form of our proposed merged measurement tracker is intractable, so we have also proposed an approximation. Two alternative implementations are presented, one based on partitioning of the target set, and a cheaper method based on relaxed measurement assignment. Simulations on a bearings-only tracking scenario confirm that the partitioning approach does indeed outperform the relaxed assignment approach, and both of these methods offer significant improvements over the GLMB filter with standard likelihood, as the latter has no mechanism for maintaining tracks in the presence of merged measurements.

APPENDIX

Lemma 7. *Let $\mathbb{X}$ be a state space and $\mathbb{L}$ a discrete label space. Then, for two functions $h:\mathcal{F}(\mathbb{L})\to\mathbb{R}$ and $g:\mathcal{F}(\mathbb{X}\times\mathbb{L})\to\mathbb{R}$, where g is integrable on $\mathcal{F}(\mathbb{X}\times\mathbb{L})$,*

$$\int \Delta(\boldsymbol{X})\, h(\mathcal{L}_{\boldsymbol{X}})\, g(\boldsymbol{X})\, \delta\boldsymbol{X} = \sum_{L\subseteq\mathbb{L}} h(L)\, g_L \qquad (49)$$

where

$$g_{\{l_{1:n}\}} = \int g(\{(x_1,l_1),...,(x_n,l_n)\})\, dx_{1:n}.$$

The full proof of this lemma is omitted due to space limitations. However, it is a straightforward generalisation of Lemma 3 in [17], to the case where g is non-separable.

Proof of Proposition 5: The density of surviving targets is

$$\begin{aligned}\pi_S(\boldsymbol{W}) &= \int f_S(\boldsymbol{W}|\boldsymbol{X})\,\pi(\boldsymbol{X})\,\delta\boldsymbol{X}\\ &= \int \Delta(\boldsymbol{W})\,\Delta(\boldsymbol{X})\,1_{\mathcal{L}_{\boldsymbol{X}}}(\mathcal{L}_{\boldsymbol{W}})\,[\Phi(\boldsymbol{W};\cdot)]^{\boldsymbol{X}}\\ &\quad\times\Delta(\boldsymbol{X})\sum_{c\in\mathbb{C}} w^{(c)}(\mathcal{L}_{\boldsymbol{X}})\,p^{(c)}(\boldsymbol{X})\,\delta\boldsymbol{X}\\ &= \Delta(\boldsymbol{W})\sum_{c\in\mathbb{C}}\sum_{L\subseteq\mathbb{L}} 1_L(\mathcal{L}_{\boldsymbol{W}})\,w^{(c)}(L)\,\eta_S^{(c)}(L)\,p_{S,L}^{(c)}(\boldsymbol{W})\end{aligned}$$

where the last line is obtained by swapping the order of summation and integration, and applying Lemma 7 with $h(L)=1_L(\mathcal{L}_W)\,w^{(c)}(L)$ and $g(\boldsymbol{X})=[\Phi(\boldsymbol{W};\cdot)]^{\boldsymbol{X}}\,p^{(c)}(\boldsymbol{X})$. Hence the predicted density, which is the product of the birth and survival densities, is given by

$$\begin{aligned}&\pi_+(\boldsymbol{X}_+) = f_B(\boldsymbol{X}_B)\,\pi_S(\boldsymbol{X}_S)\\ &= \Delta(\boldsymbol{X}_B)\,\Delta(\boldsymbol{X}_S)\sum_{c\in\mathbb{C}}\sum_{L\subseteq\mathbb{L}} w_B(\mathcal{L}_{\boldsymbol{X}_B})\,1_L(\mathcal{L}_{\boldsymbol{X}_S})\\ &\quad\times w^{(c)}(L)\,\eta_S^{(c)}(L)\,[p_B(\cdot)]^{\boldsymbol{X}_B}\,p_{S,L}^{(c)}(\boldsymbol{X}_S)\\ &= \Delta(\boldsymbol{X}_+)\sum_{c\in\mathbb{C}}\sum_{L\subseteq\mathbb{L}} w_B(\mathcal{L}_{\boldsymbol{X}_+}-\mathbb{L})\,1_L(\mathcal{L}_{\boldsymbol{X}_+}\cap\mathbb{L})\\ &\quad\times w^{(c)}(L)\,\eta_S^{(c)}(L)\,[p_B(\cdot)]^{\boldsymbol{X}_+-\mathbb{X}\times\mathbb{L}}\,p_{S,L}^{(c)}(\boldsymbol{X}_+\cap\mathbb{X}\times\mathbb{L})\\ &= \Delta(\boldsymbol{X}_+)\sum_{c\in\mathbb{C}}\sum_{L\subseteq\mathbb{L}} w_{+,L}^{(c)}(\mathcal{L}_{\boldsymbol{X}_+})\,p_{+,L}^{(c)}(\boldsymbol{X}_+) \qquad \square\end{aligned}$$

Proof of Proposition 6: The product of the prior distribution and the likelihood is

$$\begin{aligned}\pi(\boldsymbol{X})\,g(Z|\boldsymbol{X}) &= \Delta(\boldsymbol{X})\,\lambda\sum_{c\in\mathbb{C}}\sum_{\substack{\mathcal{U}(\boldsymbol{X})\in\mathcal{P}(\boldsymbol{X})\\ \theta\in\Theta(\mathcal{U}(\mathcal{L}_{\boldsymbol{X}}))}} w^{(c)}(\mathcal{L}_{\boldsymbol{X}})\\ &\quad\times\left[\check{\psi}_Z(\cdot;\theta)\right]^{\mathcal{U}(\boldsymbol{X})}\,p^{(c)}(\boldsymbol{X})\\ &= \Delta(\boldsymbol{X})\,\lambda\sum_{c\in\mathbb{C}}\sum_{\substack{\mathcal{U}(\boldsymbol{X})\in\mathcal{P}(\boldsymbol{X})\\ \theta\in\Theta(\mathcal{U}(\mathcal{L}_{\boldsymbol{X}}))}} w^{(c)}(\mathcal{L}_{\boldsymbol{X}})\,\eta_Z^{(c,\theta)}(\mathcal{U}(\mathcal{L}_{\boldsymbol{X}}))\\ &\quad\times p^{(c,\theta)}(\mathcal{U}(\boldsymbol{X})|Z)\end{aligned} \qquad (50)$$

and its integral with respect to the multi-object state is

$$\begin{aligned}&\int \pi(\boldsymbol{X})\,g(Z|\boldsymbol{X})\,\delta\boldsymbol{X}\\ &= \lambda\sum_{c\in\mathbb{C}}\sum_{n=0}^{\infty}\frac{1}{n!}\sum_{(l_{1:n})\in\mathbb{L}^n}\sum_{\substack{\mathcal{U}(\{l_{1:n}\})\in\mathcal{P}(\{l_{1:n}\})\\ \theta\in\Theta(\mathcal{U}(\{l_{1:n}\}))}}\\ &\quad\times w^{(c)}(\{l_{1:n}\})\,\eta_Z^{(c,\theta)}(\mathcal{U}(\{l_{1:n}\}))\\ &\quad\times\int p^{(c,\theta)}(\mathcal{U}(\{(x_1,l_1),\ldots,(x_n,l_n)\})|Z)\,dx_{1:n}\\ &= \lambda\sum_{c\in\mathbb{C}}\sum_{n=0}^{\infty}\sum_{L\in\mathcal{F}_n(\mathbb{L})}\sum_{\substack{\mathcal{U}(L)\in\mathcal{P}(L)\\ \theta\in\Theta(\mathcal{U}(L))}} w^{(c)}(L)\,\eta_Z^{(c,\theta)}(\mathcal{U}(L))\\ &= \lambda\sum_{c\in\mathbb{C}}\sum_{L\subseteq\mathbb{L}}\sum_{\substack{\mathcal{U}(L)\in\mathcal{P}(L)\\ \theta\in\Theta(\mathcal{U}(L))}} w^{(c)}(L)\,\eta_Z^{(c,\theta)}(\mathcal{U}(L))\end{aligned} \qquad (51)$$

The result (27) follows by substituting (50) and (51) into the Bayes update equation (3). $\square$

Derivation of (31): The product of the prior distribution and the likelihood is

$$\begin{aligned}\pi(\boldsymbol{X})\,g(Z|\boldsymbol{X}) &= \Delta(\boldsymbol{X})\,\lambda\sum_{c\in\mathbb{C}}\sum_{\substack{\mathcal{U}(\boldsymbol{X})\in\mathcal{P}(\boldsymbol{X})\\ \theta\in\Theta(\mathcal{U}(\mathcal{L}_{\boldsymbol{X}}))}} w^{(c)}(\mathcal{L}_{\boldsymbol{X}})\\ &\quad\times\left[\check{\psi}_Z(\cdot;\theta)\right]^{\mathcal{U}(\boldsymbol{X})}\left[p^{(c)}(\cdot)\right]^{\boldsymbol{X}}.\end{aligned}$$

Using equation (34),

$$\left[\tilde{p}^{(c)}(\cdot)\right]^{\mathcal{U}(\boldsymbol{X})} = \prod_{\boldsymbol{Y}\in\mathcal{U}(\boldsymbol{X})} \tilde{p}^{(c)}(\boldsymbol{Y}) = \prod_{\boldsymbol{Y}\in\mathcal{U}(\boldsymbol{X})} \left[p^{(c)}(\cdot)\right]^{\boldsymbol{Y}} = \left[p^{(c)}(\cdot)\right]^{\boldsymbol{X}}$$

hence we can write

$$\begin{aligned}\pi(\boldsymbol{X})\, g(Z|\boldsymbol{X}) &= \Delta(\boldsymbol{X})\,\lambda \sum_{c\in\mathbb{C}} \sum_{\substack{\mathcal{U}(\boldsymbol{X})\in\mathcal{P}(\boldsymbol{X}) \\ \theta\in\Theta(\mathcal{U}(\mathcal{L}_{\boldsymbol{X}}))}} w^{(c)}(\mathcal{L}_{\boldsymbol{X}}) \\ &\quad \times \left[p^{(c)}(\cdot)\,\breve{\psi}_Z(\cdot;\theta)\right]^{\mathcal{U}(\boldsymbol{X})} \\ &= \Delta(\boldsymbol{X})\,\lambda \sum_{c\in\mathbb{C}} \sum_{\substack{\mathcal{U}(\boldsymbol{X})\in\mathcal{P}(\boldsymbol{X}) \\ \theta\in\Theta(\mathcal{U}(\mathcal{L}_{\boldsymbol{X}}))}} w^{(c)}(\mathcal{L}_{\boldsymbol{X}}) \left[\eta_Z^{(c,\theta)}(\cdot)\right]^{\mathcal{U}(\mathcal{L}_{\boldsymbol{X}})} \\ &\quad \times \left[p^{(c,\theta)}(\cdot|Z)\right]^{\mathcal{U}(\boldsymbol{X})} \end{aligned} \tag{52}$$

and its integral

$$\begin{aligned}&\int \pi(\boldsymbol{X})\, g(Z|\boldsymbol{X})\,\delta\boldsymbol{X} \\ &= \lambda \sum_{c\in\mathbb{C}} \sum_{\substack{\mathcal{U}(\boldsymbol{X})\in\mathcal{P}(\boldsymbol{X}) \\ \theta\in\Theta(\mathcal{U}(\mathcal{L}_X))}} \int \Delta(\boldsymbol{X})\, w^{(c)}(\mathcal{L}_{\boldsymbol{X}}) \left[\eta_Z^{(c,\theta)}(\cdot)\right]^{\mathcal{U}(\mathcal{L}_{\boldsymbol{X}})} \\ &\quad \times \left[p^{(c,\theta)}(\cdot|Z)\right]^{\mathcal{U}(\boldsymbol{X})} \delta\boldsymbol{X} \\ &= \lambda \sum_{c\in\mathbb{C}} \sum_{\substack{\mathcal{U}(\boldsymbol{X})\in\mathcal{P}(\boldsymbol{X}) \\ \theta\in\Theta(\mathcal{U}(\mathcal{L}_{\boldsymbol{X}}))}} \sum_{L\subseteq\mathbb{L}} w^{(c)}(L) \left[\eta_Z^{(c,\theta)}(\cdot)\right]^{\mathcal{U}(L)}. \end{aligned} \tag{53}$$

The posterior (31)-(35) follows by substitution of (52) and (53) into the Bayes update equation (3). □

REFERENCES

[1] F. E. Daum and R. J. Fitzgerald, "The importance of resolution in multiple target tracking," *Proc. of SPIE 2235, Proc. SPIE Signal & Data Processing of Small Targets* , vol. 329, 1994.

[2] K.-C. Chang, Y. Bar-Shalom, "Joint probabilistic data association for multitarget tracking with possibly unresolved measurements and maneuvers", *IEEE Trans. Autom. Cont*rol, vol. 29, no. 7, pp. 585-594, July 1984.

[3] D. Svensson, M. Ulmke, and L. Danielsson, "Joint probabilistic data association filter for partially unresolved target groups", *Proc. 13th Int. Conf. Inform. Fusion*, Edinburgh, UK, July 2010.

[4] D. Svensson, M. Ulmke and L. Hammarstrand, "Multitarget Sensor Resolution Model and Joint Probabilistic Data Association", *IEEE Trans. Aerosp. Elecron. Syst.*, vol. 48, no. 4, pp. 3418-3434, Oct. 2012.

[5] W. Koch, G. van Keuk, "Multiple hypothesis track maintenance with possibly unresolved measurements", *IEEE Trans. Aerosp. Elecron. Syst.* Vol. 33, No. 3, pp. 883-892, July 1997.

[6] H. A. P. Blom, E. A. Bloem, "Bayesian tracking of two possibly unresolved maneuvering targets", *IEEE Trans. Aerosp. Elecron. Syst.,* vol. 43, no. 2, pp. 612-627, Apr. 2007.

[7] S. Jeong, J. K. Tugnait, "Tracking of two targets in clutter with possibly unresolved measurements", *IEEE Trans. Aerosp. Elecron. Syst.*, vol. 44, no. 2, pp. 748-765, April 2008.

[8] T. Kirubarajan, Y. Bar-Shalom, K. R. Pattipati, "Multiassignment for tracking a large number of overlapping objects", *IEEE Trans. Aerosp. Electron. Syst.*, vol. 37, no. 1, pp. 2-21, Jan. 2001.

[9] H. Chen, K. Pattipati, T. Kirubarajan, and Y. Bar-Shalom, "General data association with possibly unresolved measurements using linear programming," *Proc. Conf. Comput. Vis. Pattern Recog. Workshop (CVPRW)*, Madison, WI, USA, June 2003.

[10] Z. Khan, T. Balch, and F. Dellaert, "Multitarget tracking with split and merged measurements", *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, pp. 605-610, June 2005.

[11] S. J. Davey, "Tracking possibly unresolved targets with PMHT", *Information, Decision and Control (IDC) 2007*, Feb. 2007.

[12] D. Musicki, M. Morelande, "Finite Resolution Multitarget Tracking", *Proc. SPIE Signal and Data Processing of Small Targets*, vol. 5913, 2005.

[13] D. Musicki, W. Koch, "Multi scan target tracking with finite resolution sensors", *Proc. 11th Int. Conf. Inform. Fusion*, Cologne, Germany, July 2008.

[14] R. P. S. Mahler, *Statistical Multisource-Multitarget Information Fusion*, Artech House, 2007.

[15] R. Mahler, "Multitarget Bayes filtering via first-order multitarget moments", *IEEE Trans. Aerosp. Elecron. Syst.*, vol. 39, no. 4, pp. 1152-1178, Oct. 2003.

[16] R. Mahler, "PHD filters for nonstandard targets, II: Unresolved targets", *Proc. 12th Int. Conf. Inform. Fusion*, pp. 922-929, Seattle, Washington, USA, July 2009.

[17] B.-T. Vo, B.-N. Vo, "Labeled random finite sets and multi-object conjugate priors", *IEEE Trans. Signal Process.*, vol. 61, no. 13, pp. 3460-3475, July 2013.

[18] B.-N. Vo, B.-T. Vo, and D. Phung, "Labeled Random Finite Sets and the Bayes Multi-Target Tracking Filter", *IEEE Trans. Signal Process.*, vol. 62, no. 29, pp. 6554-6567, 2014.

[19] B.-N. Vo, S. Singh, and A. Doucet, "Sequential Monte Carlo methods for Bayesian Multi-target filtering with Random Finite Sets", *IEEE Trans. Aerosp. Electron. Syst.*, vol. 41, no. 4, pp. 1224-1245, 2005.

[20] M. Beard, B.-T. Vo, B.-N. Vo, "Multi-target Tracking with Merged Measurements Using Labelled Random Finite Sets", *Proc. 17th Int. Conf. Inform. Fusion*, Salamanca, Spain, July 2014.

[21] F. Papi, B.-N. Vo, B.-T. Vo, C. Fantacci, M. Beard. "Generalized Labeled Multi-Bernoulli Approximation of Multi-Object Densities", arXiv preprint arXiv:1412.5294, December 2014.

[22] R. Mahler, "PHD filters of higher order in target number", *IEEE Trans. Aerosp. and Elec. Syst.*, vol. 43, no. 4, pp.1523-1543, Oct 2007.

[23] B.-T. Vo, B.-N. Vo, A. Cantoni. "The cardinality balanced multi-target multi-Bernoulli filter and its implementations", *IEEE Trans. Signal Process.*, vol. 57, no. 2, pp. 409-423, Feb. 2009.

[24] D. B. Reid, "An Algorithm for Tracking Multiple Targets", *IEEE Trans. Autom. Control*, vol. AC-24, pp. 843-854, Dec. 1979.

[25] T. Kurien, "Issues in the Design of Practical Multitarget Tracking Algorithms", *Multisensor-Multitarget Tracking: Advanced Applications Vol. 1*, pp. 43-83, Artech House, 1990.

[26] K. G. Murty, "An algorithm for ranking all the assignments in increasing order of cost", *Operations Research*, vol. 16, no. 3, pp. 682-678, 1968.

[27] D. P. Bertsekas, "The auction algorithm: A distributed relaxation method for the assignment problem", *Annals of Operations Research*, vol. 14, no. 1, pp. 105-123, 1988.

[28] J. Y. Yen, "Finding the K-shortest loopless paths in a network", *Management Science*, vol. 17, pp. 712-716, 1971.

[29] D. Schuhmacher, B.-T. Vo, B.-N. Vo, "A consistent metric for performance evaluation of multi-object filters", *IEEE Trans. Signal Process.*, vol. 56, no. 8, pp. 3447-3457, Aug. 2008.