Citation
Lee, J.Y. and Shim, C. and Nguyen, H.V. and Nguyen, T.T.D. and Choi, H. and Kim, Y. 2023. Label Space Partition Selection for Multi-Object Tracking Using Two-Layer Partitioning. In Proceedings of 2023 12th International Conference on Control, Automation and Information Sciences (ICCAIS), 27-29 Nov 2023, Hanoi, Vietnam. http://doi.org/10.1109/ICCAIS59597.2023.10382268

Label Space Partition Selection for Multi-Object Tracking Using Two-Layer Partitioning

Ji Youn Lee[1], Changbeom Shim[1*], Hoa Van Nguyen[1],
Tran Thien Dat Nguyen[1], Hyunjin Choi[2], and Youngho Kim[3]
[1]*School of Electrical Engineering, Computing and Mathematical Sciences*, Curtin University, Perth, Australia
[2]*AI R&D Center*, AIBIZ Co. Ltd., Seoul, Republic of Korea
[3]CMW Geosciences, Perth, Australia
jiyoun.lee2@postgrad.curtin.edu.au changbeom.shim@curtin.edu.au hoa.v.nguyen@curtin.edu.au
t.nguyen1@curtin.edu.au hjchoi@ai-biz.net younghok@cmwgeo.com

***Abstract*—Estimating the trajectories of multi-objects poses a significant challenge due to data association ambiguity, which leads to a substantial increase in computational requirements. To address such problems, a divide-and-conquer manner has been employed with parallel computation. In this strategy, distinguished objects that have unique labels are grouped based on their statistical dependencies, the intersection of predicted measurements. Several geometry approaches have been used for label grouping since finding all intersected label pairs is clearly infeasible for large-scale tracking problems. This paper proposes an efficient implementation of label grouping for label-partitioned generalized labeled multi-Bernoulli filter framework using a secondary partitioning technique. This allows for parallel computation in the label graph indexing step, avoiding generating and eliminating duplicate comparisons. Additionally, we compare the performance of the proposed technique with several efficient spatial searching algorithms. The results demonstrate the superior performance of the proposed approach on large-scale data sets, enabling scalable trajectory estimation.**

***Index Terms*—Multi-object tracking, Random finite set, Generalized labeled multi-Bernoulli, Label grouping**

I. Introduction

The objective of Multi-Object Tracking (MOT) is to estimate the trajectories of an unknown number of objects varying over time from a series of sensor measurements. It is challenging due to the imperfections of the sensor generate uncertainties such as noisy measurements, false alarms, missed detections, and unknown data associations [1]–[4]. In spite of these formidable challenges, MOT plays a pivotal role in diverse applications, including but not limited to surveillance [1], [2], computer vision [5], aerospace [6], cell biology [7], robotics [8]. The majority of MOT algorithms can be categorized into three main frameworks: Multiple Hypothesis Tracking (MHT) [1], [2], [6], Joint Probabilistic Data Association (JPDA) [2], and Random Finite Sets (RFS) [3].

The RFS-based framework, in particular, has gained popularity over the last two decades due to its rigorous mathematical foundations. It utilizes finite set-valued random variables to represent multi-object states and measurements and recursively propagates the multi-object density using the Bayes multi-object filter [3], [9]. Various RFS multi-object filters have been devised such as Probability Hypothesis Density (PHD) [9], Cardinalized Probability Hypothesis Density (CPHD) [10], Multi-Bernoulli (MB) [3], [11], and Poisson Multi-Bernoulli Mixture (PMBM) [12] filters. However, the main drawback of these filters is their inability to estimate identities or trajectories, necessitating an additional post-processing procedure to reconstruct object trajectories. Notably, trajectories serve as a crucial tool for capturing object behavior, while labels facilitate the differentiation of individual trajectories and enable communication of relevant object information to both human and machine users. The first mathematically principled and tractable RFS-based multi-object tracker is the Generalized Labeled Multi-Bernoulli (GLMB) filter [4], [13]. Its premise is the idea of labeled RFS which assigns a unique label to each object, thereby enabling the joint estimation of the object's states and trajectories.

The GLMB filter has demonstrated its performance and availability [14]–[22]. In particular, remarkable scalability was shown by tracking over one million concurrent objects [19]. This achievement was made through the use of functional approximation and efficient parallel computation, providing evidence of the robustness and efficiency of the GLMB filter. A crucial aspect of the functional approximation involves computing a product of manageable and nearly independent smaller GLMB densities rather than relying on the large GLMB density itself. Specifically, smaller GLMB densities comprise label groups that have statistical dependence on each other and each label group is approximately independent.

Although selecting a partition of the label space is feasible via *label partitioning* [19], further practical implementation of scalable MOT is still hindered by computationally inefficient procedures. Object groups well-separated in the measurement space exhibit low statistical dependence, as they seldom share the same measurements. Especially in the measurement update stage, it is notably time-consuming within the scalable GLMB filter, as it necessitates the examination of all potential label pairs to ascertain whether their gating regions intersect. This bottleneck becomes particularly pronounced in large-scale and/or high-density multi-object scenarios. Therefore, the need

This work was partially supported by the Australian Research Council via the Linkage project LP200301507.
∗ Corresponding author

for an efficient label partitioning implementation is imperative to enable the practical use of scalable GLMB filters in real-world scenarios.

To improve computational efficiency in label partitioning, spatial indexing techniques can be leveraged. One frequently employed method is data-driven partitioning, where objects are approximated in each different level of segment-tree like R-tree [23] and its variants [24]–[26]. In [19], R-tree was utilized for indexing labels and examining whether their gating regions intersect spatially. On the other hand, space-driven partitioning involves dividing the space into spatially disjoint partitions such as kd-tree [27], quad-tree [28], and grid [29]. Among these, exploiting the grid structure supports straightforward implementation and parallelization, making it suitable for in-memory processing [30]–[33], which was also studied for label partitioning [34], [35]. However, a potential drawback is to get rid of excessive duplicates when retrieving intersecting objects in a dense area due to objects being present in multiple grid cells [36]. Numerous approaches have been introduced to address the issue of duplicate results in space-driven partitioning [37]–[41].

In this paper, we propose an efficient implementation for label space partition selection in label-partitioned GLMB filtering. The computational bottleneck for parallel MOT is mitigated by using the latest space-oriented technique. Specifically, we employ a two-layer partitioning technique [42] when searching statistically dependent objects, which is important for grouping multiple objects and processing parallel GLMB filtering. To reduce the number of intersection tests, every object within each spatial partition is further divided into secondary classifying. In our implementation, redundant steps in checking overlapping labels are pruned so that a feasible label space partition can be found efficiently. A numerical study is conducted to evaluate the performance of the proposed method.

The remainder of this paper is organized as follows. Section II gives essential background on the GLMB filters, and Section III presents our proposed approach. The performance of our method is evaluated in section IV through various experiments. Finally, Section V provides concluding remarks.

II. Preliminaries

This section provides a brief background for this paper. Initially, in Section II-A, we introduce the standard GLMB filter. Section II-B delves into the scalable GLMB filter and label partitioning. Additionally, to provide insight from a spatial join perspective, we introduce prominent spatial join techniques in Section II-C. For a more comprehensive understanding, further details can be found in references [4], [13], [19], [36].

A. Generalized Labeled Multi-Bernoulli Tracker

The GLMB filter is designed to estimate object trajectories by modeling the multi-object state with labeled RFS. In particular, a multi-object state $\mathbf{X}$ is a collection of the single-object state $\mathbf{x} = (x, \ell)$ where $x \in \mathbb{X}$ is its kinematic state, and ℓ is a distinct label in some discrete label space $\mathbb{L}$. We define a projection $\mathcal{L} : \mathbb{X} \times \mathbb{L} \to \mathbb{L}$ by $\mathcal{L}((x, \ell)) = \ell$. A set $\mathbf{X}$ of labeled objects has no duplicate labels if $\mathbf{X}$ and its labels $\mathcal{L}(\mathbf{X}) = \{\mathcal{L}(\mathbf{x}) : \mathbf{x} \in \mathbf{X}\}$ have the same cardinality, i.e., $|\mathcal{L}(\mathbf{X})| = |\mathbf{X}|$. In other words, if the labels are distinct we have $\Delta(\mathbf{X}) \triangleq \delta_{|\mathbf{X}|}(|\mathcal{L}(\mathbf{X})|) = 1$, where the generalized Kronecker delta function is defined as

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

(with arbitrary argument, i.e., set, vector or scalar). Besides, the integral of a function $\mathbf{f} : \mathcal{F}(\mathbb{X} \times \mathbb{L}) \to \mathbb{R}$ is given by $\int \mathbf{f}(\mathbf{x}) d\mathbf{x} = \sum_{\ell \in \mathbb{L}} \int_{\mathbb{X}} \mathbf{f}((x, \ell)) dx$.

For compact expression, following [13], we define the multi-object exponential as

$$\vartheta^X \triangleq \prod_{x \in X} \vartheta(x),$$

and the set inclusion function follows as

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

The GLMB filter propagates the GLMB (multi-object) density of the form

$$\boldsymbol{\pi}(\mathbf{X}) = \Delta(\mathbf{X}) \sum_{\xi \in \Xi} w^{(\xi)}(\mathcal{L}(\mathbf{X})) \left[p^{(\xi)}\right]^{\mathbf{X}}, \tag{1}$$

where: Ξ is a discrete index set; $p^{(\xi)}(\cdot, \ell)$ is a probability density; and $w^{(\xi)}(I)$ is non-negative weight that satisfies $\sum_{\xi \in \Xi} \sum_{L \in \mathcal{F}(\mathbb{L})} w^{(\xi)}(L) = 1$ with $\mathcal{F}(\mathbb{L})$ is all finite subsets of $\mathbb{L}$ including the empty set. The GLMB filter is a closed-form multi-object tracking filter since the prior and the filtering densities are both GLMB families. At each time step, the number of terms in the filtering GLMB density grows exponentially, hence only significant components (ones with high weights) are retained to meet the computational constraint.

B. Label-partitioned GLMB Filtering

In the standard GLMB filter implementation, the computational demand grows significantly when the number of objects increases since the filter complexity is at least in the square number of objects [43]. The primary computational bottleneck occurs during the measurement update stage, with statistical dependence mainly arising from uncertainty in the associations between measurements and objects. This challenge is especially pronounced in dense scenarios where objects are close to each other.

Nevertheless, in practice, objects often travel in groups (i.e., pedestrians, flock of birds), where each group can be assumed to be well-separated from the other. Exploiting this fact, a scalable implementation of the GLMB filter [19] approximates the GLMB density by a product of GLMBs for relatively independent groups of objects. It reduces the complexity to the logarithmic magnitude in the number of objects from the quadratic magnitude in the standard implementation.

Let $\mathfrak{L}$ be some partition of the label space $\mathbb{L}$. A labeled RFS density on $\mathcal{F}(\mathbb{X} \times \mathbb{L})$ is considered as $\mathfrak{L}$ *-partitioned GLMB*

density $\boldsymbol{\pi}_{\mathfrak{L}}$ if it can be expressed as a product of GLMBs (each defined on $\mathcal{F}(\mathbb{X} \times L)$, for $L \in \mathfrak{L}$), as follows

$$\boldsymbol{\pi}_{\mathfrak{L}}(\boldsymbol{X}) = \prod_{L \in \mathfrak{L}} \boldsymbol{\pi}_{\mathfrak{L}}^{(L)}(\boldsymbol{X} \cap (\mathbb{X} \times L)). \tag{2}$$

If the current filtering density is an $\mathfrak{L}$-partitioned GLMB, the new filtering density $\boldsymbol{\pi}_+$ at the next time step is a standard GLMB [19], not suitable for scalable implementation. To maintain the scalability, the new filtering density can be approximated by an $\mathfrak{L}$-partitioned GLMB. Nevertheless, since the current partition $\mathfrak{L}$ might not be a suitable partition for approximating the new filtering density, the goal is to find the optimal partition $\mathfrak{L}_+$ of the next time step label space $\mathbb{L}_+$ (in some statistical sense) in the space of all possible partitions of $\mathbb{L}_+$ (denoted as $\mathcal{P}(\mathbb{L}_+)$) for the new filtering density.

C. Label Partitioning

Label partitioning in the scalable GLMB filter [19] aims to find a partition of the labels to minimize the amount of potential measurement sharing between objects in different groups. Naturally, labels that are well-separated in the measurement space exhibit low statistical dependence because the likelihood of these labels producing closely spaced measurements is minimal. Here, "well-separated" means the distance between trajectories of labels is significantly larger than measurement noise and predicted object position uncertainty. Assume each factor $\boldsymbol{\pi}_{\mathfrak{L}}^{(L)}$ has the following form:

$$\boldsymbol{\pi}_{\mathfrak{L}}^{(L)} = \left\{ \left(w_{\mathfrak{L},L}^{(I,\xi)}, p_{\mathfrak{L},L}^{(\xi)} \right) \right\}_{(I,\xi) \in \mathcal{F}(L) \times \Xi^{(L)}}. \tag{3}$$

Then at the next time step, the distribution of the predicted measurement of each label $\ell \in \mathbb{L}_+$ is given by

$$\tilde{p}_+(z, \ell) = \int g(z|x, \ell) p_+(x, \ell) dx \tag{4}$$

where $p_+(x, \ell)$ is the predicted state and $g(z|x, \ell)$ is the single-object measurement likelihood function.

For each label $\ell \in \mathbb{L}_+$, one can use the distribution $\tilde{p}_+(\cdot, \ell)$ to establish a gating region, $B(\ell) \subseteq \mathbb{Z}$, which holds most of the probability mass for the predicted measurement. These gating regions form the foundation for dividing the new label space $\mathbb{L}_+$ into a partition $\mathfrak{L}_+ = \{L_1, \ldots, L_{|\mathfrak{L}_+|}\}$. Furthermore, to find feasible label partitions, the maximum number of labels within a single partition is constrained by $L_{\max}$, i.e., $|L| \leq L_{\max}, \forall L \in \mathfrak{L}_+$. In summary, $\mathfrak{L}_+$ must meet the following conditions [19]:

1) For all $L \in \mathfrak{L}_+$ and for any $\ell_i, \ell_j \in L$, either $B(\ell_i) \cap B(\ell_j) \neq \emptyset$ or $B(\ell_i) \cap B(\ell_1) \neq \emptyset$, $B(\ell_1) \cap B(\ell_2) \neq \emptyset$, ..., $B(\ell_n) \cap B(\ell_j) \neq \emptyset$ for $\{\ell_1, ..., \ell_n\} \subseteq L$;
2) For all $i, j \in \{1, ..., |\mathfrak{L}_+|\}$ and $i \neq j$, $[\bigcup_{\ell \in L_i} B(\ell)] \cap [\bigcup_{\ell \in L_j} B(\ell)] = \emptyset$; and
3) For all $L \in \mathfrak{L}_+$, $|L| \leq L_{max}$.

The process involves updating entire prediction gates to decrease the gating probability P_G until the specified limit of $L_{\max}$ is reached. Algorithm 1 illustrates the proposed label partitioning process.

Finding $\mathfrak{L}_+$ poses challenges regarding time complexity due to the necessity of examining all possible label pairs to determine if their predicted gating regions $B(\ell)$, referred to as Gaussian-mixed Minimum Bounding Rectangles (GMBRs), intersect. Additionally, this step must be reiterated until each partition's maximum label count is no greater than $L_{\max}$. In [19], an R-tree is employed to alleviate the computational complexity of the partitioning, making it feasible for such problems. However, using an R-tree structure for indexing becomes increasingly burdensome as the number of procedure iterations grows [42].

Algorithm 1: Label Space Partitioning [19]

Input : $\mathcal{L}(\mathbf{X}), P_G, L_{\max}$
Output : $\mathfrak{L}_+$

do
  $M \leftarrow$ `FindGMBR(`$\mathcal{L}(\mathbf{X}), P_G$`)`;
  $\mathfrak{P} \leftarrow$ `SelectLabelPartition(`M`)`;
  $\mathfrak{L}_+ \leftarrow$ `GroupLabel(`$\mathfrak{P}$`)`;
  Reduce P_G;
while $\max|L| \leq L_{max}$;

To address these computational challenges, an alternative approach is to employ a grid structure for object indexing and retrieval. Indexing the object with its associated grid can be accomplished with linear time complexity, through algebraic operations [44], [45]. While indexing objects offers computational advantages, the issue of duplicate results arises when objects are assigned to multiple grid-based tiles, necessitating additional processing.

A straightforward approach to eliminate duplicates is by hashing the search results and checking for duplicates within each candidate set [37]. While this method is simple, it can be costly, particularly when dealing with a large number of results. Another widely used method involves using a reference point within the intersection area between each result and the search range. If the reference point falls inside the partition, the result is reported; otherwise, it is discarded [40]. While this method eliminates the need for hashing, we must account for the cost associated with identifying duplicate results and computing the reference point for each of them [42].

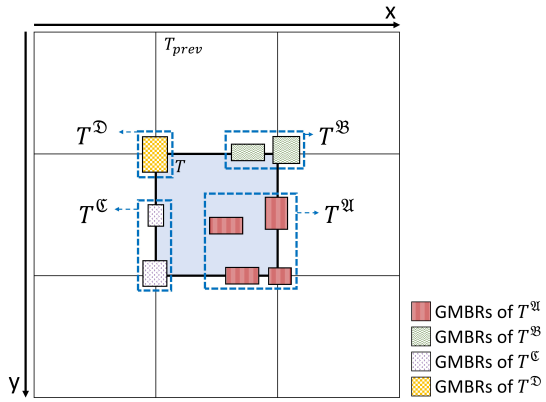
III. Efficient Label-Partitioning for GLMB Filters

A secondary partitioning approach refers to a spatial indexing technique used in *space-driven partitioning* spatial index, such as a grid. It involves categorizing the indexed objects as rectangles within each spatial partition into four secondary classes, as introduced in [42]. This approach reduces the number of comparisons needed during intersection assessments and eliminates the generation of duplicate results on the grid.

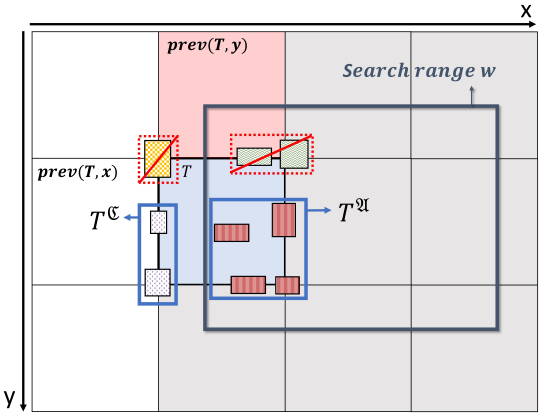
Section III-A provides an overview of the secondary partitioning technique proposed in [42]. Section III-B describes the approach for label partitioning using this secondary partitioning technique.

A. Secondary Classifying for Label Partitioning

1) Secondary Classifying: Consider a regular grid that divides the space into $N \times M$ disjoint spatial partitions,



(a) Secondary Classes



(b) Selecting Relevant Classes

Fig. 1: Examples of Secondary Classifying

referred to as tiles. Given a set of GMBRs, each GMBR r is indexed into tiles that spatially intersect with r. Assume a 2D dimension, i.e., dimension $d \in \{x, y\}$, the coordinates of the GMBR are denoted as intervals; $r.x = [r.x_l, r.x_u]$ and $r.y = [r.y_l, r.y_u]$ on the respective x and y axes. Here, $r.d_l$ and $r.d_u$ denote the minimum and maximum coordinates of the GMBR r on each dimension, respectively. Subsequently, each GMBR r assigned to each tile T is further classified into a secondary class, $\mathfrak{A}$, $\mathfrak{B}$, $\mathfrak{C}$ and $\mathfrak{D}$, as follows:

- r is categorized into $\mathfrak{A}$, if r originates within tile T for both dimensions, i.e.,

  if, $T.x_l \leq r.x_l$ and $T.y_l \leq r.y_l$;
- r is categorized into $\mathfrak{B}$, if $r.x$ starts within $T.x$ and $r.y$ starts before $T.y$, i.e.,

  if, $T.x_l \leq r.x_u$ and $T.y_l > r.y_l$;
- r is categorized into $\mathfrak{C}$, if $r.x$ starts before $T.x$ and $r.y$ starts within $T.y$, i.e.,

  if, $T.x_l > r.x_l$ and $T.y_l \leq r.y_l$; and
- r is categorized into $\mathfrak{D}$, if r originates before tile T for both dimensions, i.e.,

  if, $T.x_l > r.x_l$ and $T.y_l > r.y_l$.

Fig. 1 (a) provides visual examples of GMBRs classifying into the four secondary classes within a tile. For convenience, the secondary classes of tile T are denoted as $T^{\mathfrak{A}}$, $T^{\mathfrak{B}}$, $T^{\mathfrak{C}}$ and $T^{\mathfrak{D}}$ respectively.

2) Selecting Valid Classes: Label partitioning utilizing a grid structure can yield duplicated results when attempting to find groups of intersecting GMBRs. For instance, in Fig. 1 (a), rectangles within both $T^{\mathfrak{B}}$ and $T^{\mathfrak{D}}$ are compared twice: once when inspecting the tile T_{prev} and then again when inspecting tile T. Consequently, results may be duplicated if intersecting GMBRs are present. Given a search range w, secondary classes prevent the generation of duplicate results based on the following criteria:

- If w intersects with tiles T and starts before tile T in the x dimension, then it is necessary to exclude $T^{\mathfrak{C}}$ and $T^{\mathfrak{D}}$;
- If w intersects tile T and w starts before T in dimension y, then $T^{\mathfrak{B}}$ and $T^{\mathfrak{D}}$ should be disregarded; and
- If w intersects tile T and w starts before T in both dimensions, then only $T^{\mathfrak{A}}$ should be considered.

Note that the proof of these criteria can be found in [42].

Fig. 1 (b) provides an illustrative example of selecting relevant classes. Let $prev(T, d)$ denote the tile that precedes T in the specified dimension d. Given a search range w, only the tiles that intersect with w are sequentially inspected. In this scenario, w starts before tile T only in the y dimension. Therefore, we can selectively check the rectangles in the $T^{\mathfrak{A}}$ and $T^{\mathfrak{C}}$ because rectangles in the $T^{\mathfrak{B}}$ and $T^{\mathfrak{D}}$ have been examined when $prev(T, y)$ was inspected. Note that more stages for reducing the number of comparisons have been proposed in [42]. However, this work does not consider these additional criteria because they may not be suitable here.

B. Implementation of Label Partitioning

We proposed a computationally efficient method for label space partitioning combined with a secondary classifying approach. It possesses a notable advantage in terms of ease of implementation and seamless integration into diverse, complex research projects and large-scale spatial data management systems, facilitating parallel computation. Moreover, it effectively lowers time while consistently delivering robust performance. While this method is applicable to any *space-driven partitioning* spatial index, we have specifically tailored it for use within a two-dimensional grid structure to optimize label partitioning efficiency.

Algorithm 2 describes the label space partition selection of label partitioning for scalable GLMB filters. Given a set of GMBRs M and grid configuration $\mathcal{G}$, all GMBRs are classified into secondary classes for each tile t in the `SecondaryClassifying` stage based on III-A1. Following this, the intersection search process begins to find spatially intersecting GMBRs within a specified search range. Here, each GMBR from the set M is used as a search range w. During each iteration, relevant tiles intersecting w are selected using algebraic operations in the `GetRelevantTiles` stage. Within the relevant tiles containing information about the secondary classes, the `GetSecondaryClass` stage is executed to select the most relevant classes for each tile based on III-A2. By selecting the relevant secondary classes,

Algorithm 2: Label Space Partition Selection

Input: Grid $\mathcal{G}$, GMBRs M
Output: Label partitions $\mathfrak{P}$

Function `TwoLayerLabelPartition`$(\mathcal{G}, M)$**:**
  $\mathfrak{P} \leftarrow \emptyset$;
  $\mathcal{S}_{\mathcal{G}} \leftarrow$ `SecondaryClassifying`$(\mathcal{G}, M)$;
  foreach *GMBR* $w \in M$ **do**
    $T_m \leftarrow$ `GetRelevantTiles`$(\mathcal{G}, w)$;
    $\mathcal{S}_t \leftarrow$ `GetRelevantClasses`$(w, T_m, \mathcal{S}_{\mathcal{G}})$;
    $\mathcal{I}_w \leftarrow$ `IntersectingTest`$(w, T_w, \mathcal{S}_t^X)$;
    $\mathfrak{P} \leftarrow \mathfrak{P} \cup (w, \mathcal{I}_w)$;
  return $\mathfrak{P}$;

`IntersectingTest` stage finds intersected GMBRs with w among a set of GMBRs from these classes. As a result, the `TwoLayerLabelPartition` function returns label partitions that group GMBRs intersecting with each other. In summary, the secondary classifying technique reduces redundant comparisons, leading to computational efficiency in the label partitioning function.

IV. Numerical Study

In this section, we discuss the experimental performance of our method. The effectiveness and computational efficiency are evaluated by changing the number of labels. All experiments were conducted on a machine with 64GB of RAM and an Intel(R) Xeon(R) CPU E5-2696 v3 @ 2.30GHz running Ubuntu 22.04 using Python 3.10.12.

Baseline. We evaluate the simulation results by comparing the proposed method with secondary partitioning (so-called Two-layer) to two existing approaches: R-tree [19] and *inclusion-checking grid* (IG) [34]. IG utilizes grid-based label partitioning along with a pruning mechanism to decrease the number of intersection checks and time complexity. Our approach adopts the principles of IG but goes a step further by combining secondary classification to eliminate duplicated results. Note that all filtering processes are the same across the three methods, with only the label partitioning stage being adjusted. The results of label partitioning in all three methods are identical, meaning they correctly determine intersected GMBRs for each GMBR, although with variations in processing time.

Data sets. We employed diverse datasets encompassing varying object cardinalities, ranging from 10K to 100K. These objects were randomly generated with a uniform distribution and represented by GMBRs with a maximum width or height of 20m. Furthermore, our analysis focused on a 2 km $\times$ 2 km surveillance area and the increase of cardinalities indicates that the test environment is going to be denser. By default, we configured the tests with 30K objects and 100 tiles on each axis of the grid for other experiments.

The initial experiments involve comparing the performance of the proposed approach with that of R-tree and IG across datasets of varying sizes. In Fig. 2, we present the total CPU time results for different numbers of labels. It is evident that the two grid-based structures consistently outperform the R-tree in terms of CPU times. Furthermore, the proposed algorithm performs similarly to IG until the dataset size reaches 50K objects. However, it surpasses IG thereafter, and the performance gap widens as the dataset size increases. The experimental results emphasize the effective reduction in time complexity achieved by the proposed method, particularly in dense environments.

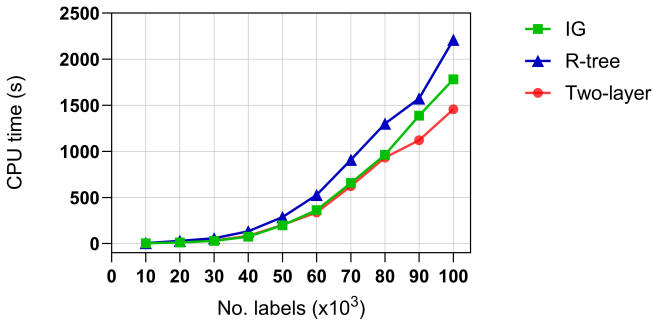


Fig. 2: Comparison of the total processing time

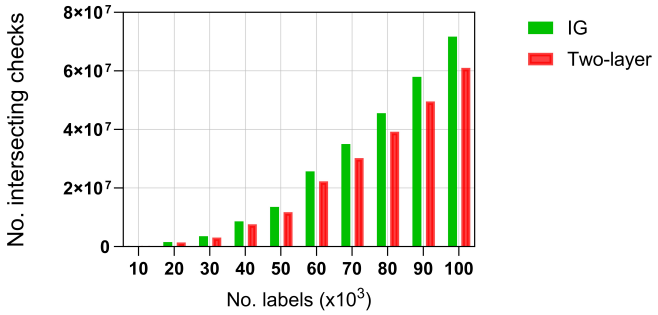


Fig. 3: Comparison of the number of intersecting checks

The superiority of the proposed approach is attributed to its reduction in the computation of intersecting tests by minimizing duplicate results compared to IG. Figure 3 compares the number of intersecting tests between the two grid-based methods. As shown in the graph, the gap between these two methods widens as the dataset size increases, resulting in reduced processing time.

V. Conclusion

This paper has proposed an efficient implementation for label partitioning in the scalable GLMB filter. By employing a secondary classifying technique, our approach eliminates unnecessary comparisons, resulting in improved computational efficiency. To assess its effectiveness, we conducted simulation experiments, which demonstrated its superior performance compared to other methods, especially as the cardinality of objects increases. Future work includes efficient parallel GLMB filtering on various real-world datasets.

References

[1] S. Blackman and R. Popoli, "Design and analysis of modern tracking systems(book)," *Norwood, MA: Artech House, 1999.*, 1999.

[2] Y. Bar-Shalom, P. K. Willett, and X. Tian, *Tracking and data fusion*. YBS publishing Storrs, CT, USA:, 2011, vol. 11.

[3] R. Mahler, *Statistical multisource-multitarget information fusion*. Artech House, Inc., 2007.

[4] B.-N. Vo, B.-T. Vo, and D. Phung, "Labeled random finite sets and the bayes multi-target tracking filter," *IEEE Transactions on Signal Processing*, vol. 62, no. 24, pp. 6554–6567, 2014.

[5] I. J. Cox and S. L. Hingorani, "An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking," *IEEE Trans. on Patt. Analysis and Machine Intel.*, vol. 18, no. 2, pp. 138–150, Feb 1996.

[6] D. Reid, "An algorithm for tracking multiple targets," *IEEE Trans. on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979.

[7] T. T. D. Nguyen, B.-N. Vo, B.-T. Vo, D. Y. Kim, and Y. S. Choi, "Tracking cells and their lineages via labeled random finite sets," *IEEE Transactions on Signal Processing*, vol. 69, pp. 5611–5626, 2021.

[8] H. V. Nguyen, M. Chesser, L. P. Koh, H. Rezatofighi, and D. C. Ranasinghe, "Trackerbots: Autonomous unmanned aerial vehicle for real-time localization and tracking of multiple radio-tagged animals," *J. of Field Robotics*, vol. 36, no. 3, pp. 617–635, 2019.

[9] R. P. Mahler, "Multitarget bayes filtering via first-order multitarget moments," *IEEE Transactions on Aerospace and Electronic systems*, vol. 39, no. 4, pp. 1152–1178, 2003.

[10] R. Mahler, "PHD filters of higher order in target number," *IEEE Transactions on Aerospace and Electronic systems*, vol. 43, no. 4, pp. 1523–1543, 2007.

[11] B.-T. Vo, B.-N. Vo, and A. Cantoni, "The cardinality balanced multi-target multi-Bernoulli filter and its implementations," *IEEE Transactions on Signal Processing*, vol. 57, no. 2, pp. 409–423, 2008.

[12] J. L. Williams, "Marginal multi-Bernoulli filters: RFS derivation of MHT, JIPDA, and association-based MeMBer," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 3, pp. 1664–1687, 2015.

[13] B.-T. Vo and B.-N. Vo, "Labeled random finite sets and multi-object conjugate priors," *IEEE Transactions on Signal Processing*, vol. 61, no. 13, pp. 3460–3475, 2013.

[14] B.-N. Vo, B.-T. Vo, and H. G. Hoang, "An efficient implementation of the generalized labeled multi-bernoulli filter," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 1975–1987, 2016.

[15] B.-N. Vo, B.-T. Vo, and M. Beard, "Multi-sensor multi-object tracking with the generalized labeled multi-Bernoulli filter," *IEEE Transactions on Signal Processing*, vol. 67, no. 23, pp. 5952–5967, 2019.

[16] B.-N. Vo and B.-T. Vo, "A multi-scan labeled random finite set model for multi-object state estimation," *IEEE Transactions on Signal Processing*, vol. 67, no. 19, pp. 4948–4963, 2019.

[17] T. T. D. Nguyen, B.-N. Vo, B.-T. Vo, D. Y. Kim, and Y. S. Choi, "Tracking cells and their lineages via labeled random finite sets," *IEEE Transactions on Signal Processing*, vol. 69, pp. 5611–5626, 2021.

[18] H. Van Nguyen, H. Rezatofighi, B.-N. Vo, and D. C. Ranasinghe, "Distributed multi-object tracking under limited field of view sensors," *IEEE Transactions on Signal Processing*, vol. 69, pp. 5329–5344, 2021.

[19] M. Beard, B. T. Vo, and B.-N. Vo, "A solution for large-scale multi-object tracking," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2754–2769, 2020.

[20] A. Trezza, D. J. Bucci, and P. K. Varshney, "Multi-sensor joint adaptive birth sampler for labeled random finite set tracking," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1010–1025, 2022.

[21] D. Moratuwage, B.-N. Vo, B.-T. Vo, and C. Shim, "Multi-scan multi-sensor multi-object state estimation," *IEEE Trans. Signal Process.*, vol. 70, pp. 5429–5442, 2022.

[22] C. Shim, B.-T. Vo, B.-N. Vo, J. Ong, and D. Moratuwage, "Linear complexity Gibbs sampling for generalized labeled multi-Bernoulli filtering," *IEEE Transactions on Signal Processing*, 2023.

[23] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, 1984, pp. 47–57.

[24] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The r*-tree: An efficient and robust access method for points and rectangles," in *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, 1990, pp. 322–331.

[25] T. Sellis, N. Roussopoulos, and C. Faloutsos, "The r+-tree: A dynamic index for multi-dimensional objects." 1987.

[26] K. Kim, S. K. Cha, and K. Kwon, "Optimizing multidimensional index trees for main memory access," *ACM SIGMOD Record*, vol. 30, no. 2, pp. 139–150, 2001.

[27] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[28] R. A. Finkel and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta informatica*, vol. 4, no. 1, pp. 1–9, 1974.

[29] J. L. Bentley and J. H. Friedman, "Data structures for range searching," *ACM Computing Surveys (CSUR)*, vol. 11, no. 4, pp. 397–409, 1979.

[30] K. Mouratidis, D. Papadias, and M. Hadjieleftheriou, "Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, 2005, pp. 634–645.

[31] U. Demiryurek, F. Banaei-Kashani, and C. Shahabi, "Efficient continuous nearest neighbor query in spatial networks using euclidean restriction," in *International Symposium on Spatial and Temporal Databases*. Springer, 2009, pp. 25–43.

[32] W. Kim, C. Shim, W. Heo, S. Yi, and Y. D. Chung, "Moving view field nearest neighbor queries," *Data & Knowledge Engineering*, vol. 119, pp. 58–70, 2019.

[33] J. Du, B. Shen, S. Zhao, M. A. Cheema, and A. N. Toosi, "Efficient object search in game maps," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, 8 2023, pp. 5567–5576.

[34] C. Shim and D. Y. Kim, "Space-oriented label partitioning for multi-object tracking," in *2021 International Conference on Control, Automation and Information Sciences (ICCAIS)*. IEEE, 2021, pp. 425–431.

[35] C. Shim, J. Y. Lee, D. Moratuwage, D. Y. Kim, and Y. D. Chung, "Generalized label grouping for scalable trajectory estimation," in *2022 11th International Conference on Control, Automation and Information Sciences (ICCAIS)*. IEEE, 2022, pp. 30–35.

[36] E. H. Jacox and H. Samet, "Spatial join techniques," *ACM Transactions on Database Systems (TODS)*, vol. 32, no. 1, pp. 7–es, 2007.

[37] W. G. Aref and H. Samet, "Hashing by proximity to process duplicates in spatial databases," in *Proceedings of the third international conference on Information and knowledge management*, 1994, pp. 347–354.

[38] J. M. Patel and D. J. DeWitt, "Partition based spatial-merge join," *ACM Sigmod Record*, vol. 25, no. 2, pp. 259–270, 1996.

[39] X. Zhou, D. J. Abel, and D. Truffet, "Data partitioning for parallel spatial join processing," *Geoinformatica*, vol. 2, pp. 175–204, 1998.

[40] J.-P. Dittrich and B. Seeger, "Data redundancy and duplicate detection in spatial join processing," in *Proceedings of 16th International Conference on Data Engineering (Cat. No. 00CB37073)*. IEEE, 2000, pp. 535–546.

[41] J. Yu, Z. Zhang, and M. Sarwat, "Spatial data management in apache spark: the geospark perspective and beyond," *GeoInformatica*, vol. 23, pp. 37–78, 2019.

[42] D. Tsitsigkos, K. Lampropoulos, P. Bouros, N. Mamoulis, and M. Terrovitis, "A two-layer partitioning for non-point spatial data." in *ICDE*, 2021, pp. 1787–1798.

[43] B.-N. Vo, B.-T. Vo, and H. G. Hoang, "An efficient implementation of the generalized labeled multi-Bernoulli filter," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 1975–1987, 2016.

[44] D. Šidlauskas, S. Šaltenis, C. W. Christiansen, J. M. Johansen, and D. Šaulys, "Trees or grids? indexing moving objects in main memory," in *Proceedings of the 17th ACM SIGSPATIAL international conference on Advances in Geographic Information Systems*, 2009, pp. 236–245.

[45] S. Ray, R. Blanco, and A. K. Goel, "Supporting location-based services in a main-memory database," in *2014 IEEE 15th International Conference on Mobile Data Management*, vol. 1. IEEE, 2014, pp. 3–12.