# Real-time Bandwidth-efficient Occupancy Grid Map Synchronization for Multi-Robot Systems

Liuyu Shi*, Longji Yin*, Fanze Kong*, Yunfan Ren, Fangcheng Zhu, Benxu Tang, and Fu Zhang

***Abstract*— Robot swarms are increasingly being applied in various domains. However, due to the inherent limitation imposed by low real-time communication bandwidth, the synchronization of environmental information among multiple robots remains a persistent and challenging problem in practical applications. In response to this challenge, we introduce a comprehensive framework for synchronizing occupancy grid maps (OGMs) in practical multi-robot systems that operate under communication bandwidth constraints. In our research, we elaborately design the data structure of transmitted local OGMs and employ the Hilbert space-filling curve for voxel sorting. By adopting this approach, data redundancy is effectively increased, resulting in lower information entropy for compression and significantly reducing the volume of communication data. Finally, our framework outperforms the benchmark method by reducing the average and maximum bandwidth usage by more than 10 times in high-resolution scenarios. Moreover, our method has been successfully applied in the multi-UAV autonomous navigation application, demonstrating its real-time and bandwidth-efficient nature, as well as its practical value.**

## I. Introduction

The application of multi-robot systems has remarkably pushed beyond the limits of individual robot capabilities, embracing a spectrum of both complex and innovative operations [1]–[3]. However, the development of multi-robot systems also brings about a range of challenges. A critical and persistent issue is the constraint of limited communication bandwidth, which makes it hard for each robot to have real-time synchronization of environment information measured by other robots in the system. This limitation becomes particularly pronounced in operations requiring high-level coordination and optimal decision-making.

Among various possible forms of environmental information for synchronization, such as point cloud maps and depth images, the Occupancy Grid Map (OGM) stands out as a robust and effective structure for representing the environment. It has become an essential component in robotic systems due to its crucial role in the planning and decision-making process. Synchronization of OGMs is a common requirement in the application of multi-robot systems. For instance, in multi-robot autonomous exploration, robots generate frontiers based on the jointly perceived and synchronized OGMs from individual robots [1], [4]. These frontiers often act as critical targets in the decision-making process for allocating tasks among the robots. Furthermore, OGM synchronization can be utilized for real-time navigation in multi-robot systems, enabling each robot to synchronize with teammates' observations of the environment. This allows for a more comprehensive understanding of the surroundings and facilitates the formulation of superior navigation strategies [5], [6], as shown in Fig. 1.

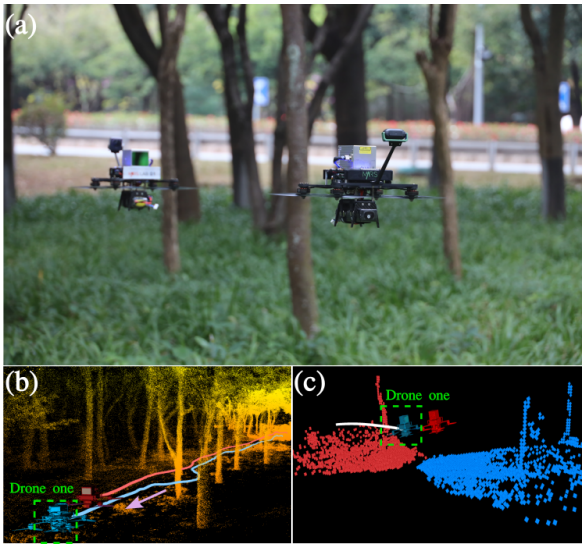


Fig. 1: A demo showcasing OGM synchronization among two autonomously navigating drones equipped with Livox Avia LiDAR. (a) A drone is capable of avoiding obstacles in areas not observed by its onboard LiDAR, by leveraging the OGM shared by its teammate. (b) Drone one, accompanied by its historical trajectory (blue line), reaches the goal after flying in an opposite direction from its LiDAR FoV. (c) Obstacle avoidance by drone one based on the map (red part) synchronized from its teammate. More details can be found in the attached video at https://youtu.be/ZQXo5RkOfS4.

However, OGMs have a large data volume due to their fine-grained representation, requiring a large number of high-resolution grids with additional attributes for each grid. It implies that high communication bandwidth is required for transmission. However, considering the restricted bandwidth capacity of existing communication networks like Wi-Fi or 5G in real-world settings, synchronizing high-resolution OGMs across multi-robot systems becomes a daunting task.

Certain research has acknowledged the previous issue and tends to offer solutions that leverage lightweight environment representations to lessen the data needed for communication. Methods like using star-convex polytopes to depict known free space [7], and the Gaussian Mixture Model (GMM) as a global spatial representation of the environment [8], are examples of such solutions. Nonetheless, these strategies can introduce extraneous computation and potential inaccuracies, possibly leading to the omission of essential map information. A direct and bandwidth-efficient synchronization

L. Shi, L. Yin, F. Kong, Y. Ren, F. Zhu, B. Tang, and F. Zhang are with the Department of Mechanical Engineering, University of Hong Kong. {liuyushi, ljyin, kongfz, renyf, zhufc, tangbenx}@connect.hku.hk, {fuzhang}@hku.hk.

* **Equal contribution.**

method that does not lose OGM information needs to be considered.

To address the aforementioned challenges, we introduce a unique data structure for OGM compression and transmission, along with a comprehensive framework for occupancy grid map synchronization, tailored for practical multi-robot systems operating under the constraints of limited communication bandwidth. Summarizing our contributions as follows:

1) We propose a bandwidth-efficient data structure specifically designed for OGM compression and transmission, which enables real-time sharing of map packages without loss of environmental information.
2) We design a comprehensive OGM synchronization framework that can support various applications in multi-robot systems.
3) We verify the effectiveness of our framework by applying it to real-world multi-UAV autonomous navigation experiments. Our framework can synchronize 0.1-m resolution with 7.5-m sensing range OGMs among two drones with an average bandwidth usage of 25 KB/s.

## II. Related Works

The effective sharing of environmental information under communication bandwidth constraints is a frequently considered and referenced issue in the design of multi-robot coordination systems. Taking into account the mentioned bandwidth limitations, some works have adopted the approach of maintaining and transmitting a lightweight environment representation to avoid the transmission of commonly used but data-heavy voxel maps or point clouds, which proves to be convenient for the planning and decision-making process. For instance, certain studies in [8], [9] employ the Gaussian mixture model (GMM) as a global spatial representation of the environment to ensure communication-efficient multi-robot exploration. This model effectively compresses a vast amount of data into several parameters. In a similar application context, Gao *et al.* [7] employ star-convex polytopes to represent the explored free space, which enables the acquisition of essential environment information for exploration while keeping the bandwidth cost low. In order to achieve efficient communication with the ground station for collaborative SLAM in the forest scene, Tian *et al.* [10] adopted object-based representations, specifically employing a tree-based representation. This approach involves compressing each submap into a sparse collection of trees with several estimated tree parameters before transmission. Although the aforementioned methods make sense in specific applications, they lack accurate and comprehensive environmental map information and introduce unnecessary computation. Consequently, they do not possess the generality for planning and decision-making akin to the occupancy grid map. Some collaborative multi-robot systems employ map-sharing modules that transmit the information of voxels in OGMs with specific states, aiming to reduce bandwidth consumption. For instance, Racer [1] only transmits the addresses and occupancy states of voxels that have newly observed. In ColAG [6], only the addresses of occupied and unknown grids in a specific region around a robot are transmitted. However, these map-sharing methods offer limited bandwidth reduction and are not suitable for general cases with larger update ranges and higher resolution. As far as we are aware, there is no framework available in the community that compresses OGMs to meet transmission bandwidth requirements and accurately synchronize environmental information.

## III. Occupancy Grid Map Data Compression

In this section, we present an effective method to compress OGM data for transmission. We begin by explaining the motivation behind the design of this method based on the data compression mechanism (Sec. III-A). We then proceed to present the specific process of compressing OGMs, including the design of the data structure (Sec. III-B), the utilization of the space-filling curve for sorting (Sec. III-C), and the compression of the data stream using the zlib library (Sec. III-D).

### A. Data Compression Mechanism

The process of data compression is to seek a shorter encoding for a source sequence. Essentially, it aims to identify the probability distribution $p(x)$ of the content to be compressed and replaces those parts with a high probability of occurrence with shorter forms based on a specific encoding algorithm. Information entropy, denoted as $H(X)$, is a pivotal concept in source coding. It characterizes the average amount of information, excluding redundancy. The definition is given by Eq. 1, which varies continuously with the probability $p(x)$. It determines the optimal expected encoding length $L^*$ for each symbol in a source sequence, as represented by Eq. 2 [11]. There are many existing methods and libraries to implement optimal coding, such as Huffman coding [12]–[14]. To improve compression performance, we need to reduce the inherent information entropy $H(X)$ of the source, that is, reorganizing the form of the original data to change its probability distribution $p(x)$ such that the optimal expected encoding length $L^*$ is as small as possible.

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x) \tag{1}$$

$$H(X) \leq L^* < H(X) + 1 \tag{2}$$

- $X$: A discrete random variable with alphabet $\mathcal{X}$ and probability mass function $p(x) = \Pr(X = x), x \in \mathcal{X}$.
- $H(X)$: Information entropy of $X$.
- $L^*$: The associated expected length of an optimal code and the unit is bits.

During the process of synchronizing OGMs among multiple robots, it is common to transmit specific types of voxels, such as state-changed voxels [1] or voxels with specific occupancy states [6]. Since the voxels for transmission cannot be determined in advance but depend on the actual update of the OGM, it becomes necessary to transmit the voxels' addresses (often represented by a 32-bit integer to ensure a sufficient address space) along with their occupancy states. Transmitting the addresses of all these voxels contributes

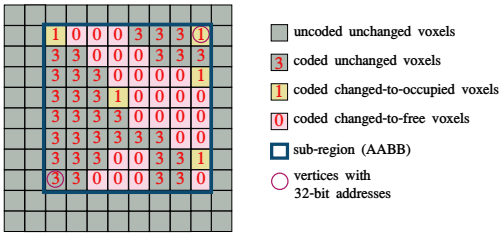


Fig. 2: A demonstration to show data structure.

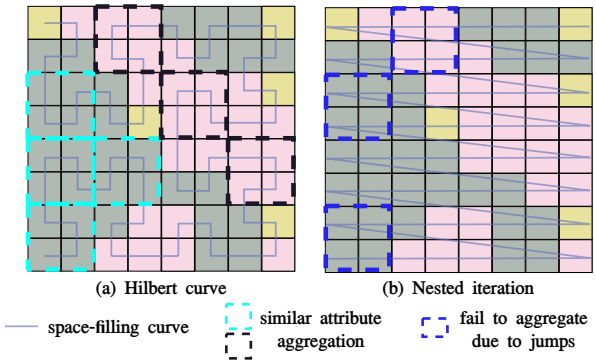


Fig. 3: The figure gives an example of a comparison between the Hilbert curve and nested iteration, using the same data structure as depicted in Fig. 2. The blue box in (b) shows some of the voxels whose attributes are the same but are not neighboring in the nested method. In contrast, voxels with similar attributes are mostly neighboring in the Hilbert curve (bright blue box and black box in (a)).

to a large amount of data, and the unique value of each address for transmission leads to a uniform distribution of the content (i.e. the addresses) that maximizes entropy and hence the encoding length $L^*$. This characteristic makes them unsuitable for processing by compression algorithms. Therefore, it is necessary to consider restructuring the data format to avoid transmitting a large number of addresses and also to increase data redundancy which is more suitable for compression.

Unlike existing methods [1], [6] transmitting certain discontinuous voxels in the OGM, we transmit the states of all the voxels in a local OGM box (see Sec. III-B), eliminating the requirement to transmit each voxel's address individually since the complete set of voxels contained in the box can be arranged in a pre-determined order (see Sec. III-C). The only need is to transmit the boundary of the OGM box which can be accomplished using only two 32-bit addresses, and the states of all voxels within the boundary. To lower the data volume, the state of each voxel is represented by a 2-bit data entry representing KnownFree ("00"), Occupied ("01"), Unknown ("10"), and Unchanged ("11"). The introduction of the "Unchanged" state is to enhance data redundancy, as there are often a vast number of consecutive voxels whose occupancy states do not change during map updates. To further optimize the probability distribution, we utilize Hilbert curves to enhance the aggregation of similar attributes among neighboring voxels, which will further increase the redundancy of the data stream for zlib compression. The detailed method will be introduced in the following sections.

### B. Data Structure

During the map update process, it is crucial to identify and record the voxels whose occupancy status has changed since we can leverage these state-changed voxels to calculate a minimum sub-region of OGM to transmit, which could further reduce data volume. To achieve this, an axis-aligned bounding box (AABB) of all state-changed voxels is computed and generated as the local OGM box (i.e., the blue box in Fig. 2). Two vertex coordinates of the AABB are converted into two 32-bit integer addresses using a hash function mentioned in [15] to represent the box boundary.

To represent the state of each voxel in the AABB, a 2-bit attribute value is utilized to represent four states: Unchanged ("11") denotes voxels whose state remains unchanged in the current update cycle. If the state of the voxel changes, then KnownFree ("00"), Occupied ("01"), and Unknown ("10") correspond to the post-change occupancy states of the voxel. It is noteworthy that we treat the Unknown("10") as Unchanged ("11") since the Unknown("10") does not provide any information that can contribute to map fusion. The introduction of the "Unchanged" category serves the purpose of enhancing data redundancy by assigning all voxels, regardless of their current occupancy status, the same attribute value as long as their occupancy status remains unchanged. Conversely, if the occupancy status of each unchanged voxel is transmitted distinctively, it can lead to a reduction in data redundancy since the occupancy states of these voxels may vary.

### C. Space-Filling Curves

The essence of compression lies in the identification and elimination of redundancies within data. To achieve this, we begin by discerning these redundancies through an examination of the characteristics of OGM voxel models.

Voxels with the same attribute tend to cluster together. In many instances, 3D models display discernible patterns, with a considerable portion consisting of geometry and substantial empty regions commonly referred to as air or void. This concept can be likened to what artists describe as negative space. Effective compression methods must efficiently organize these extensive areas of both positive and negative space. Similar to the color attribute, the occupancy status of a voxel is closely correlated with its geometry. The changes in voxels also exhibit a high level of continuity as the map updates. This implies that the attribute distances between neighboring voxels are considerably smaller than the distances between randomly scattered points in this attribute space.

Hilbert curves are a specific category of space-filling curves characterized by their ability to form a one-dimensional line that effectively occupies a three-dimensional space. Their distinctive trait lies in their ability to preserve a high spatial locality, ensuring that points closely positioned along the curve also maintain proximity within the three-dimensional spatial context. Compared to

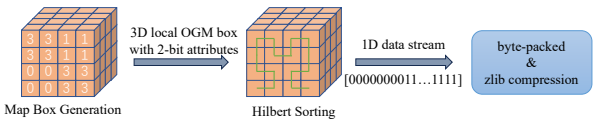


Fig. 4: Work flow diagram of OGM compression for transmission.

the conventional nested iteration method where traversal jumps occur each time a fixed axis is completed, Hilbert curves are continuous curves without any jumps. This unique characteristic ensures that all points along the curve are direct neighbors. The advantages of the Hilbert curve are depicted as shown in Fig. 3 in a comparative manner. Therefore, by utilizing the Hilbert curve indexing order to sort each voxel within the local OGM box, we can effectively group spatially adjacent voxels. Since adjacent voxels tend to have similar attribute values, we essentially grouped similar attribute values together, further increasing the information redundancy in the data stream. Subsequently, every four 2-bit voxel attributes are packed into a byte and are processed by compression.

### D. Compression

After packing the 2-bit voxel attribute list into a byte vector, we utilize zlib for compression. zlib is a powerful and widely-used open-source software library that internally employs a combination of a variation of LZ77 and Huffman coding [12]. The core principle of the Huffman tree is to assign shorter binary codes to more frequently occurring symbols, reducing the overall length of the encoded data while ensuring that each symbol can still be uniquely decoded. This property makes Huffman coding an efficient method for data compression. Similarly, the LZ77 compression algorithm effectively reduces the size of input data by replacing redundant information with metadata. As a result, zlib achieves superior compression performance in scenarios with a higher level of data redundancy, which aligns with the motivation behind our earlier design. The entire workflow of OGM compression is depicted in Fig. 4.

## IV. Occupancy Grid Map Synchronization

In this section, we outline the comprehensive workflow of OGM synchronization, encompassing map data packing (Sec. IV-A), sharing (Sec. IV-B), and map fusion after receiving map data packages (Sec. IV-C).

### A. Map Data Packing

The overall packing process of map data is shown in Alg. 1, which runs repetitively with the reception of new point cloud data and sensor pose information. First, the system performs ray casting (Line 6) and accumulates state-changed voxels (Line 8-12). If the number of state-changed voxels exceeds a predefined threshold $d$ or the duration since the last packing surpasses a certain limit $t_l$, as shown in Line 13, the process advances to the packing stage. The introduction of these two thresholds is aimed at preventing the data volume of a map package from becoming excessively large at a transmission while ensuring the synchronization speed of map transmission (i.e., flow control).

Upon satisfying the mentioned condition, the algorithm confirms the boundary of the local OGM box based on the principle of AABB (Line 15). Subsequently, the map package box is traversed voxel by voxel in a Hilbert curve's order to represent the attribute values of voxels in an array according to changes and occupancy states (Line 16). Following this, the algorithm packs the attribute list into a byte vector and employs zlib to compress the data (Line 17). At this point, a compressed local map package is generated, numbered (by subscript $i$), and stored in the corresponding container $\mathcal{H}$ (Line 18). This container $\mathcal{H}$ serves as a collection that facilitates the efficient management of map data (see Section IV-B). Finally, the algorithm resets the change status list $\mathcal{S}$ to "Unchanged" states (Line 19), concluding one cycle of the compression process. The system is then ready to receive new input data and repeat the cycle, ensuring the map remains current with changes while updating in the environment.

**Algorithm 1** Map Data Packing Algorithm

1: **Notation:** Updating candidates queue $\mathcal{C}$; Change status list $\mathcal{S}$; Voxels with changed states quantity threshold $d$; The time interval between two successive packs threshold $t_l$; Map data $\mathcal{D}_i$; Compressed map package $\mathcal{M}_i$; Compressed map package container $\mathcal{H}$.
2: **Input:** The sensor's position $x_k$; Input point cloud $\mathcal{P}_k$.
3: **Algorithm**
4: Initialize $\mathcal{S}$ with "11" (Unchanged) states;
5: **while** True **do**
6: $\quad\mathcal{C} =$ **RayCasting**$(x_k, \mathcal{P}_k)$;
7: $\quad$**if** $\mathcal{C}$.IsNotEmpty **then**
8: $\quad\quad$**for** each $voxel \in \mathcal{C}$ **do**
9: $\quad\quad\quad$**if** $voxel$.IsStateChanged **then**
10: $\quad\quad\quad\quad\mathcal{S}[voxel.\text{idx}] \leftarrow voxel$.OccupancyState;
11: $\quad\quad\quad$**end if**
12: $\quad\quad$**end for**
13: $\quad\quad$**if** ChangesInList$(\mathcal{S}) \geq d$ **or** Durtion $\geq t_l$ **then**
14: $\quad\quad\quad$/* == Packing the $i$-th Map Package == */
15: $\quad\quad\quad\mathcal{D}_i =$ **GeneratePackageBox**$(\mathcal{S})$ (Sec. III-B)
16: $\quad\quad\quad\mathcal{D}_i =$ **HilbertSorting**$(\mathcal{D}_i)$ (Sec. III-C)
17: $\quad\quad\quad\mathcal{M}_i =$ **CompressData**$(\mathcal{D}_i)$ (Sec. III-D)
18: $\quad\quad\quad\mathcal{H} \leftarrow$ **AppendCompressedPackage**$(\mathcal{M}_i)$
19: $\quad\quad\quad$**ResetChangeStatusList**$(\mathcal{S})$
20: $\quad\quad$**end if**
21: $\quad$**end if**
22: **end while**
23: **End Algorithm**

### B. Request-Response Mechanism

Drawing inspiration from the request-response mechanism used in Racer [1], we have implemented a more thoughtfully designed request-response mechanism that enables the efficient and robust transmission of compressed map packages.

Assume that there are $n$ robots, denoted as $R_1, ..., R_n$. Each robot, $R_j$, maintains $n$ lists of storage map packages ($\mathcal{H}_1, ..., \mathcal{H}_n$), where $H_j$ is used to store its own compressed map packages ($\mathcal{M}_1$,...,$\mathcal{M}_i$) generated according to Alg. 1 up to the current step $i$. The remaining lists are used to store compressed map packages received from other robots. The working principle of the request-response mechanism is as follows. $R_k$, as the requester for example, publishes its index lists that encompass the range of indices for each of the $n$ storage map package lists, denoted as $\{\mathcal{L}_{k1}, ..., \mathcal{L}_{kn}\}$. Each $\mathcal{L}_{kx}$ contains several sequentially ordered and discontinuous intervals that cover the indices it possesses. Upon receiving this request, $R_j$, acting as the responder, compares these external index lists $\{\mathcal{L}_{k1}, ..., \mathcal{L}_{kn}\}$ with its own internal index lists $\{\mathcal{L}_{j1}, ..., \mathcal{L}_{jn}\}$ correspondingly. If $R_j$ identifies any missing indices from $R_k$, it sends the corresponding missing compressed map packages as a response to $R_k$, thereby completing one cycle of the request-response process.

Since each robot stores all the compressed map packages, it can address common data loss issues during communication by publishing a request to its teammates at intervals. Subsequently, its teammates can identify the missed map packages and initiate retransmission (response) to effectively fill the loss. To prevent transmitting a large number of missing map packages during prolonged disconnection, the responder adopts a different strategy if the range of missing indices exceeds a certain threshold. In this case, the responder encodes the occupancy status of the current OGM using "00"(KnownFree), "01"(Occupied), and "10"(Unknown) codes (discarding "11"(Unchanged)) and shares the encoded OGM, which represents the current OGM as it is, to teammates. Sharing the current OGM directly can eliminate the need for transferring the full history of OGM encoded with "Unchanged" states.

It's worth noting that the responder keeps verifying which packages had been transmitted last time but was not successfully merged into other robots' OGMs yet due to communication delay by comparing the requester's current missing indices in the last interval with previous ones. This step is taken to prevent the redundant sending of the same map package that was not actually lost but just experienced a delay in communication. At the same time, it actively monitors the bandwidth usage to avoid overloading the communication channel. The responder achieves this by calculating the total volume of data it has processed during this time. If the total volume per second exceeds a defined threshold, the responder will split the data packets and send them with a time delay in between.

### C. Map Fusion Mechanism

After receiving the map package transmitted by teammates, it is crucial to integrate it effectively into global OGM. To achieve this, we employ the following strategies. Each robot maintains a timestamp list of all voxels in its global OGM. Upon receiving and decoding a map package, the receiver proceeds to compare the timestamp associated with the state-changed voxels in the received package with the timestamp of corresponding voxels in the receiver's global OGM. The occupancy state is eventually determined by the voxel with the latest timestamp to ensure the preservation of the most up-to-date voxel state.

## V. Experiments

In this section, we compare the proposed method with the map-sharing module in Racer [1] in terms of bandwidth usage (Sec. V-A) and apply it to the multi-UAV autonomous navigation system in real-world experiments (Sec. V-B) in order to demonstrate the superiority and practicality of the proposed method.

### A. Benchmark Comparison

In this section, we compare our proposed method with the map-sharing module in Racer[1] [1], which is an open-source multi-UAV exploration algorithm. Racer transmits the list of 32-bit unsigned integer addresses of voxels with state changes only, along with a corresponding occupancy status for each voxel. In terms of communication protocol, Racer employs a similar request-response mechanism based on index lists. However, Racer lacks mechanisms to prevent unnecessary retransmissions caused by reception delay during normal communication processes and traffic monitoring mentioned in Sec. IV-B.

We first analyze the factors that affect the data volume and conduct a quantitative comparison between the two methods. The data volume of the proposed method and the benchmark method primarily depends on two aspects: the changed voxel ratio (i.e., the ratio of the number of state-changed voxels to the total number of voxels) and the size of the OGM. Different changed voxel ratios lead to different amount of data to transmit in the benchmark method. Meanwhile, it can influence the probability distribution of the content (i.e., occupancy status) to be transmitted by the proposed method, thereby impacting the compression rate. On the other hand, the size of the OGM can affect the minimum number of bytes required for each address in the benchmark method. It is worth noting that, to ensure fairness, we do not consider computing the sub-region for our method, as the benchmark method can also do this (although it does not do so) by transmitting relative addresses within the sub-region.

We evaluate the two methods under different OGM sizes (measured by the number of voxels) and changed voxel ratio, we calculate the number of state-changed voxels and sample, randomly, 10 times the exact state-changed voxel location in the OGM. Among these state-changed voxels, 50% are randomly set as KnownFree ("00"), while the remaining 50% are set as Occupied ("01"). The remaining are set as Unchanged ("11"). We calculate the average data volume over all 10 sampled cases of a configuration. For each case, the transmission data volume required by Racer is determined by the number of state-changed voxels multiplied by the minimum number of bytes required for representing the full address range of the corresponding OGM of that

[1]https://github.com/SYSU-STAR/RACER

configuration (e.g. in Fig. 5, two bytes in (a), three bytes in (b) and (c), and four bytes in (d)). On the other hand, the transmission data volume for our method is determined by the size of the map package after compression, which may vary over the 10 sampled cases due to the change of distribution of state-changed and state-unchanged voxels.

The comparison in Fig. 5 is highly evident and intuitive. The proposed method only has a slightly larger data volume than the benchmark method at the beginning where the changed voxel ratio is extremely small (i.e., below 0.01 in (a), below 0.001 in (b), and 0.0005 in (c) and (d)). However, at this stage, the data volume is well within the bandwidth capacity of practical communication networks. As the changed voxel ratio increases, data volume in Racer linearly increases, while the proposed method plateaus and even exhibits a decreasing trend due to the high data redundancy in the data distribution. The reduction in data volume compared to the benchmark method becomes significantly noticeable (about one order of magnitude at a high changed voxel ratio).

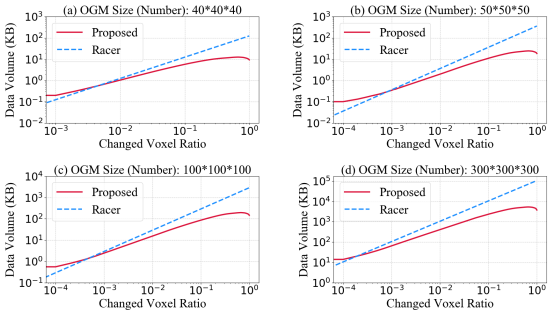


Fig. 5: The data volume comparison between the proposed method and the benchmark as the changed voxel ratio varies; (a)-(d) show the results across different OGM sizes.

Besides the numerical study, we further evaluate the two methods on data from multi-UAV autonomous exploration tasks, which are the aimed application of Racer. We have migrated Racer's exploration algorithm to the Marsim simulation environment [16] and generated multi-robot autonomous exploration data records (point clouds and LiDAR pose) in various scenarios within this environment. Using these records as input, we conducted tests in the simulation environment to compare and evaluate the transmission bandwidth usage. To ensure fairness, we conducted comparative tests using a single exploration record data package for each scene, which is the point cloud map provided by the simulator. The first scene involved two unmanned aerial vehicles (UAVs) autonomously exploring a forest environment measuring 60 m × 25 m × 10 m. The second scene consisted of five UAVs autonomously exploring an indoor environment measuring 35 m × 30 m × 4 m. For each scene, we considered two parameter settings: a resolution of 0.1 m with a sensing range of 7 m, and a resolution of 0.2 m with a sensing range of 20 m. In the proposed method, we set the threshold for the number of voxels with changed states ($d$) to 50,000, and the threshold for the time interval between two successive packets ($t_l$) to 0.5s, aiming to balance both synchronization speed and bandwidth usage. Racer does not implement any such flow control.

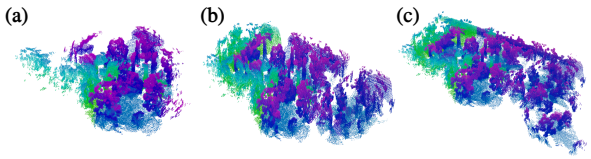


Fig. 6: (a)-(c) shows the process of OGM synchronization between two drones over time during the exploration of a forest. The blue-purple area represents the self-updated map, and the green area means the synchronized map received from the teammate.

The comparative analysis between our proposed method and Racer's focused on the average and maximum transmission bandwidth usage, as shown in Table I. The bandwidth usage is determined by calculating the total number of bytes occupied by the messages transmitted per second. For each test, the average or maximum transmission bandwidth usage was calculated by averaging or maximizing the values across all UAVs. As observed from the table, the proposed method significantly outperforms Racer in terms of both the average and maximum bandwidth usage for transmission across all scenarios. Most notably, in the forest scenario with a resolution of 0.1 m, the average bandwidth usage of the proposed method is 12 times lower than Racer, while the maximum bandwidth usage is 24 times lower. This significant reduction is mainly due to efficient data organization and compression. These results illustrate the superior efficiency and effectiveness of the proposed method in handling bandwidth limitation, implying that we can transmit a larger scope and higher resolution OGM under bandwidth constraints.

It is worth noting that, to validate the role of the Hilbert space-filling curve in the proposed method, we compared the bandwidth usage between traversing with Nest and with Hilbert sorting. Throughout this process, we maintained the other structures of the framework unchanged. We conducted tests in the same forest scene with a resolution of 0.1 m and a sensing range of 5 m and used the same record data. The results are presented in Table II. In this context, compared to nested traversal, the average and maximum bandwidth usage after Hilbert traversal both decreased by around 20%. The process of OGM synchronization between two drones during the exploration of a forest is illustrated in Fig. 6.

### *B. Real-world Experiments*

To assess the functionality and effectiveness of our new map-sharing module, we integrated it into multiple LiDAR-based quadrotor platforms, forming a robust multi-UAV system. This integration was conducted in the context of the potential specific application, allowing us to evaluate the module's performance in real-world scenarios.

*1) Application:* Occupancy Grid Map synchronization plays a crucial role in decision-making and planning for widespread applications in multi-robot systems. One of the main reasons is that OGM synchronization enables mutual supplementation of the inherent Field of View (FoV) among multiple robots, known as FoV extension. This capability

TABLE I: Bandwidth Usage Comparison

| No. | Scene | Size (m) | Drones | Resolution (m) | Sensing Radius (m) | Ave. TX BW (KB/s) | | Max TX BW (KB/s) | | Tot. TX Vol. (MB) | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Ours | Racer [1] | Ours | Racer [1] | Ours | Racer [1] |
| 1 | Forest | 60x25x10 | 2 | 0.1 | 7 | **35.84** | 430.71 | **107.42** | 2570.13 | **4.54** | 43.62 |
| 2 | Forest | 60x25x10 | 2 | 0.2 | 20 | **18.77** | 59.44 | **85.25** | 409.80 | **2.28** | 5.73 |
| 3 | Indoor | 35x30x4 | 5 | 0.1 | 7 | **9.25** | 55.70 | **56.20** | 883.53 | **2.01** | 7.93 |
| 4 | Indoor | 35x30x4 | 5 | 0.2 | 20 | **1.26** | 7.66 | **8.74** | 147.18 | **0.26** | 0.97 |

TABLE II: 3D Hilbert vs Nest

| Metric \ Method | Hilbert curve | Nested iteration |
|---|---|---|
| Ave. TX BW (KB/s) | **17.60** | 21.59 |
| Max TX BW (KB/s) | **43.97** | 59.11 |

empowers robots to navigate efficiently even within areas not observed by its own onboard sensor. We implemented our new map-sharing method in this specific application context to demonstrate its excellent performance and efficacy, which holds practical significance.

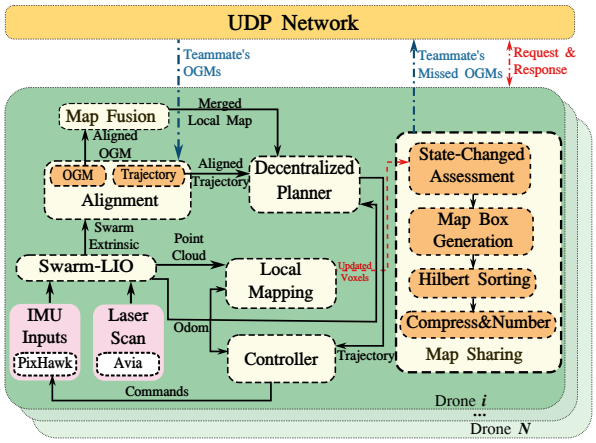


Fig. 7: The diagram of system architecture.

*2) System Architecture:* The swarm system consists of two autonomous drones equipped with Livox Avia LiDAR, enabling collaborative flight and obstacle avoidance. Localization of the swarm is achieved through a decentralized swarm LiDAR-inertial odometry (Swarm-LIO) in [17]. The decentralized planner has been meticulously designed, taking inspiration from the work referenced in [18]. We flexibly modify its constraints, including yaw and distance, to generate optimized trajectories for the swarm in this task. The local mapping on each drone is an OGM implemented based on ROG-Map [15], which is a computation-efficient occupancy grid map package that operates on uniform grids. After the raycasting in the local mapping module, the updated voxels are collected and subsequently transferred to the map-sharing module, where they undergo state-changed assessment, OGM box generation, Hilbert sorting, and compression. The process of map sharing and synchronization is facilitated through a request-response mechanism via a

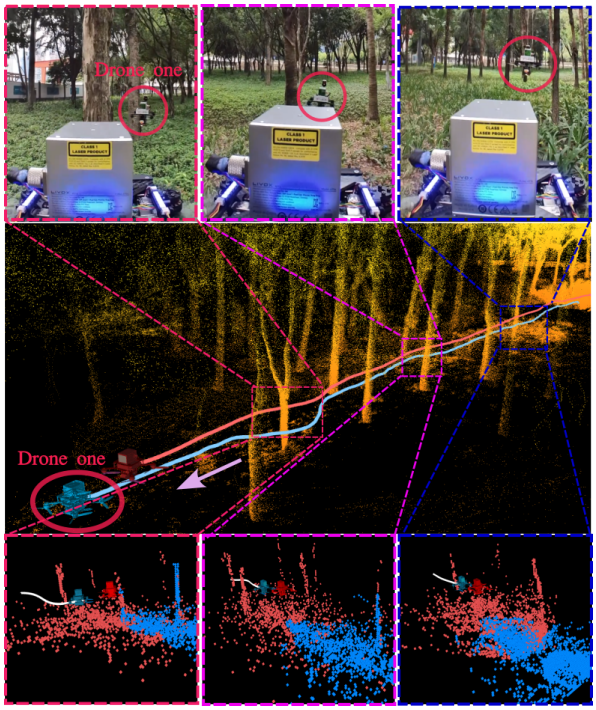


Fig. 8: First line: Drone 1 (red circle) navigating in areas not observed by its onboard sensor from teammate's view; Second line: Lio mapping and historical trajectory after reaching the goal in a small dense forest; Third line: Drone 1 (blue) and teammate's (red) local map for obstacle avoidance. The white curve represents the local trajectory of Drone 1.

UDP wireless network. The teammate's missed OGMs are transmitted by the responder. Upon receiving map packages from a teammate, the drone synchronizes the timestamps using the time offset and proceeds to align OGM data from the teammate's frame to its global frame using the swarm extrinsic provided by Swarm-LIO, which is known as OGM alignment. Subsequently, the drone performs map fusion by evaluating and merging the aligned voxels into its own map. The merged local map serves as the input for the planner. The complete system architecture is depicted in Fig. 7.

*3) Result:* We test the swarm system in an unknown real-world cluttered forest. The velocity of the drones is set to 1.5 m/s. The radius of the update scope for local mapping corresponds to 7.5 m along the x and y axes and 1.5 m along the z-axis, with the OGM resolution set to 0.1 m. We set the threshold for the map-packing time interval ($t_l$) to 0.2s to rapidly provide the map information required for navigation. In the conducted experiment, the swarm achieved successful autonomous navigation through a forest multiple times, benefiting from real-time OGM

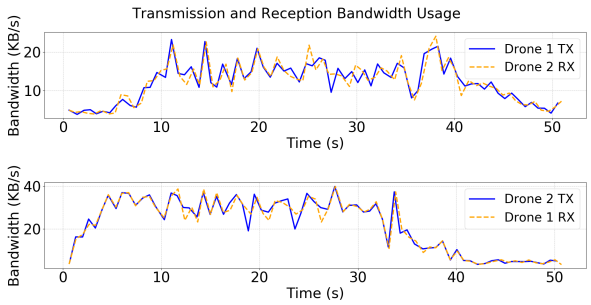


Fig. 9: Transmission and Reception Bandwidth Usage.

synchronization. The specific implementation is illustrated in Fig. 8. In experiments, two drones move towards the target point, with one positioned in front of the other one. It is important to note that the drone in the front (i.e., Drone 1 in Fig. 8) flies in an opposite direction w.r.t. its LiDAR FoV. Therefore, it relies on the OGMs transmitted by its teammate for obstacle avoidance. The onboard transmission and reception bandwidth consumed is depicted in Fig. 9, and its corresponding statistical data is presented in Table III, indicating an average bandwidth usage of around 25KB/s and a maximum bandwidth usage below 40KB/s. What is interesting to note is that the bandwidth usage between two drones in mutual communication shows some variation. This is partly due to packet loss phenomena in the UDP network and the corresponding retransmission mechanism in our method. Another contributing factor is the influence of network latency and stability. The onboard computation time, including the packing and compression process time, as well as the time from receiving to insertion into the map, is displayed in Table IV. The packing process, as well as the time it takes from receiving a map package to completing map fusion, is all completed within 10 ms, ensuring real-time transmission and processing of the map data.

TABLE III: Drone 1 Onboard Bandwidth Usage (KB/s)

| $TX_{ave}$ | $RX_{ave}$ | $TX_{max}$ | $RX_{max}$ |
|---|---|---|---|
| 25.4 | 25.3 | 39.9 | 39.9 |

TABLE IV: Onboard Computation Time (ms)

| $t_{pack}$ | $t_{recv2insert}$ |
|---|---|
| 5.30 | 7.08 |

## VI. Conclusion

In this paper, we presented a real-time and bandwidth-efficient framework for occupancy grid map synchronization, specifically designed to apply in multi-robot systems under communication bandwidth limitations. We implemented a local OGM box for transmission through using two 32-bit integer addresses to represent the boundary of the OGM box, along with 2-bit data entries that denote the state of each voxel within it. To further optimize the probability distribution, we utilized the Hilbert curves to enhance the aggregation of similar attributes among neighboring voxels. This method significantly enhanced the redundancy of the data stream, making it highly suitable for compression and effectively reducing the volume of communication data. Experiments in both simulation and real-world confirmed the efficiency of the proposed framework.

## Acknowledgement

This work is funded by the RGC General Research Fund (Grant No. 17204523) and DJI. The authors gratefully acknowledge the funding provided by DJI and the equipment support provided by Livox Technology during this project.

## References

[1] B. Zhou, H. Xu, and S. Shen, "Racer: Rapid collaborative exploration with a decentralized multi-uav system," *IEEE Transactions on Robotics*, 2023.

[2] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[3] L. Yin, F. Zhu, Y. Ren, F. Kong, and F. Zhang, "Decentralized swarm trajectory generation for lidar-based aerial tracking in cluttered environments," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 9285–9292.

[4] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proceedings of the second international conference on Autonomous agents*, 1998, pp. 47–53.

[5] J. Hou, X. Zhou, Z. Gan, and F. Gao, "Enhanced decentralized autonomous aerial swarm with group planning," 2022.

[6] Z. Li, R. Mao, N. Chen, C. Xu, F. Gao, and Y. Cao, "Colag: A collaborative air-ground framework for perception-limited ugvs' navigation," *arXiv preprint arXiv:2310.13324*, 2023.

[7] Y. Gao, Y. Wang, X. Zhong, T. Yang, M. Wang, Z. Xu, Y. Wang, Y. Lin, C. Xu, and F. Gao, "Meeting-merging-mission: A multi-robot coordinate framework for large-scale communication-limited exploration," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 13 700–13 707.

[8] M. Corah, C. O'Meadhra, K. Goel, and N. Michael, "Communication-efficient planning and mapping for multi-robot exploration in large environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1715–1721, 2019.

[9] C. O'Meadhra, W. Tabib, and N. Michael, "Variable resolution occupancy mapping using gaussian mixture models," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2015–2022, 2018.

[10] Y. Tian, K. Liu, K. Ok, L. Tran, D. Allen, N. Roy, and J. P. How, "Search and rescue under the forest canopy using multiple uavs," *The International Journal of Robotics Research*, vol. 39, no. 10-11, pp. 1201–1221, 2020.

[11] T. M. Cover, *Elements of information theory*. John Wiley & Sons, 1999.

[12] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[13] D. E. Knuth, "Dynamic huffman coding," *Journal of algorithms*, vol. 6, no. 2, pp. 163–180, 1985.

[14] A. Moffat, "Huffman coding," *ACM Computing Surveys (CSUR)*, vol. 52, no. 4, pp. 1–35, 2019.

[15] Y. Ren, Y. Cai, F. Zhu, S. Liang, and F. Zhang, "Rog-map: An efficient robocentric occupancy grid map for large-scene and high-resolution lidar-based motion planning," *arXiv preprint arXiv:2302.14819*, 2023.

[16] F. Kong, X. Liu, B. Tang, J. Lin, Y. Ren, Y. Cai, F. Zhu, N. Chen, and F. Zhang, "Marsim: A light-weight point-realistic simulator for lidar-based uavs," *IEEE Robotics and Automation Letters*, vol. 8, no. 5, pp. 2954–2961, 2023.

[17] F. Zhu, Y. Ren, F. Kong, H. Wu, S. Liang, N. Chen, W. Xu, and F. Zhang, "Swarm-lio: Decentralized swarm lidar-inertial odometry," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 3254–3260.

[18] L. Quan, L. Yin, C. Xu, and F. Gao, "Distributed swarm trajectory optimization for formation flight in dense environments," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 4979–4985.