Memory-Efficient Boundary Map for Large-Scale Occupancy Grid Mapping

Benxu Tang, Yunfan Ren, Yixi Cai, Fanze Kong, Wenyi Liu, Fangcheng Zhu, Longji Yin, Liuyu Shi and Fu Zhang

Abstract*—Environmental representation is fundamental for autonomous robotic applications. Traditional occupancy grid mapping methods subdivide the environment into a grid of voxels, each associated with one of three occupancy states: free, occupied, or unknown. These methods explicitly maintain all voxels within the mapped volume and determine the occupancy state of a location by directly querying the corresponding voxel that the location falls within. However, maintaining all grid voxels in high-resolution and large-scale scenarios requires substantial memory resources. In this paper, we introduce a novel representation that only maintains the boundary of the mapped volume. Specifically, we explicitly represent the boundary voxels, such as the occupied voxels and frontier voxels, while free and unknown voxels are automatically represented by volumes within or outside the boundary, respectively. As our representation maintains only a closed surface in two-dimensional (2D) space, instead of the entire volume in three-dimensional (3D) space, it significantly reduces memory consumption. Then, based on this 2D representation, we propose a method to determine the occupancy state of arbitrary locations in the 3D environment. We term this method as boundary map. Besides, we design a novel data structure for maintaining the boundary map, supporting efficient occupancy state queries. Furthermore, to enable efficient construction and updates of the boundary map from the real-time sensor measurements, we propose a global-local mapping framework and corresponding update algorithms. Extensive benchmark experiments were conducted on various datasets, with results demonstrating that our method significantly reduces memory consumption while maintaining highly efficient map queries and updates. Moreover, we showcase a real-world application of the boundary map by deploying our mapping framework on a memory-constrained micro aerial vehicle (MAV) platform, enabling the robot to navigate in large and complex unknown environments.

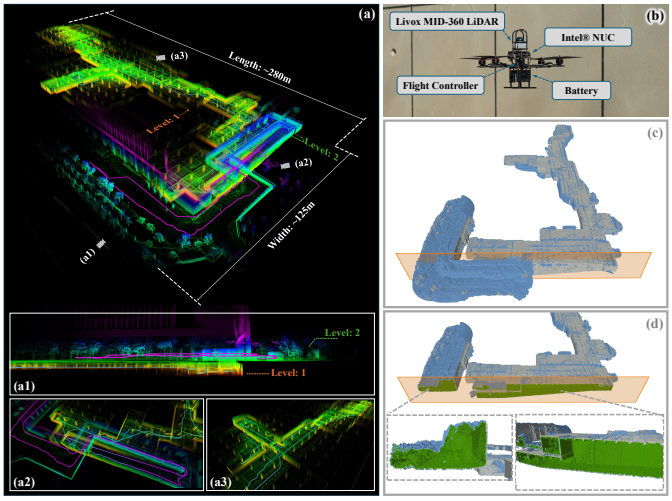

Fig. 1. MAV performing a long-range autonomous navigation task in an unknown multi-level building, utilizing the proposed mapping framework. (a) Accumulated point cloud and the MAV trajectory at the end of the task, also visualized from multiple viewpoints in (a1), (a2), and (a3). (b) Hardware configuration of the MAV platform used to perform the navigation task. (c) Overview of the resulting global boundary map, which forms a closed structure. As the viewpoint is external, the low-dimensional property of the map is not evident. To better illustrate its internal structure, the map is sliced along the orange plane. (d) Cross-sectional view revealing the hollow interior of the boundary map. Voxels stored in boundary map are classified into three types, visualized in distinct colors. The green voxels lie on the interior layer of the map, having an occupancy state of free. The grey and blue voxels are located on the exterior layer, with occupancy states of occupied and unknown, respectively.

I. Introduction

Autonomous robots have attracted significant attention in recent years and have been deployed across diverse domains, including search and rescue [1, 2], disaster response [3, 4], surveillance [5, 6], and 3D reconstruction [7, 8]. These applications not only enhance operational efficiency but also ensure the safety of human personnel. Despite notable advancements, deploying autonomous robots in complex, unknown, and large-scale environments remains a formidable challenge. A critical issue lies in the efficient and accurate determination of the occupancy status—free, occupied, or unknown—for locations in the environment. This capability is foundational for autonomous navigation tasks, such as obstacle avoidance [9]–[12], target search [13], active perception and exploration [14]–[18].

A solution for the problem is the occupancy grid mapping, which can generally be divided into two categories: (i) grid-based methods [19]–[21] and (ii) octree-based methods [22, 23]. Grid-based methods discretize the environment into a fixed grid. Each voxel in grid stores an occupancy state of free, occupied or unknown. These grid voxels are mapped to a specific memory location in an array, or a hash table. Determining the occupancy state of locations in the environment involves directly querying for the corresponding voxel that the location falls within. Octree-based methods, manage these grid voxels using a hierarchical tree structure. By merging voxels with identical occupancy status, octrees effectively reduce memory consumption compared to grid-based methods; however, this comes at the cost of significantly slower query and update speeds, which may compromise real-time performance in large-scale settings.

Mechatronics and Robotic Systems (MaRS) Laboratory, Department of Mechanical Engineering, The University of Hong Kong, Hong Kong SAR, China.

All existing occupancy grid maps explicitly represent both free and occupied volumes in the three-dimension (3D), and as the environment size and resolution increase, memory consumption can become prohibitive (e.g., exceeding 120GB [19]), restricting their application in such tasks. In addition to memory, large-scale environments also lead to substantial increases in map update time and query latency, which prohibit real-time capability. Consequently, there is a pressing need for novel techniques that facilitate efficient occupancy mapping in large-scale environments by minimizing memory overhead, and simultaneously enabling rapid map updates and queries.

To address these challenges, we propose a novel map representation and a real-time mapping framework. Unlike existing methods, which explicitly represent the entire mapped volume in the three-dimension (3D), our method represents only the two-dimensional (2D) boundary voxels (see Section II-A). This low-dimensional representation significantly reduces memory consumption, achieving improvements by several times to orders of magnitude compared to the existing methods. A problem with using 2D boundary voxel representation lies in the determination of the occupancy state of an arbitrary location in the 3D environment. To address this issue, we propose an occupancy state determination method based on the boundary voxels. Besides, to support efficient occupancy state queries, we design a novel data structure to maintain the boundary map. Our approach achieves an average time complexity of $\mathcal{O}(1)$ for occupancy state queries, matching the efficiency of grid-based methods. Besides, we introduce a global-local mapping framework to facilitate efficient real-time map updates, with the map update time comparable to the most efficient baseline methods. Furthermore, we demonstrated the framework's robustness and practical capability through real-world deployment on a Micro Aerial Vehicle (MAV) within a challenging multi-floor environment. In particular, we conducted a long-range autonomous navigation experiment in this environment based on the proposed occupancy mapping framework.

In summary, our contributions can be listed as follows:

- **The Boundary Map:** We propose a novel occupancy grid map, which represents only the two-dimensional (2D) boundary voxels instead of the entire three-dimensional (3D) volume. This low-dimensional representation significantly improves memory efficiency. Based on the 2D boundary voxels, we propose a method for determining the occupancy state of arbitrary locations in the 3D environment. We term our method as boundary map. In addition, we introduced a novel data structure to maintain the boundary map, from which an efficient query algorithm is then proposed.
- **Real-Time Mapping Framework:** We developed a global-local mapping framework that supports efficient map updates from real-time sensor measurements. This framework leverages the boundary map as the global map and a robot-centric local map implemented as a uniform occupancy grid. Based on this map structure, we design a corresponding update method which incorporates a sliding mechanism and an incremental global map update method, achieving high map update efficiency. Such real-time performance is a prerequisite for the practical deployment of autonomous applications on resource-constrained aerial platforms.
- **Real-World Autonomous Navigation Experiment:** We conducted a long-range autonomous navigation experiment of a micro aerial vehicle (MAV) in a large-scale real-world scenario. Our mapping framework supports the MAV to navigate to several long-range goals across multiple floors of a building. This demonstrates the practical applicability of our framework to support large-scale autonomous navigation tasks.

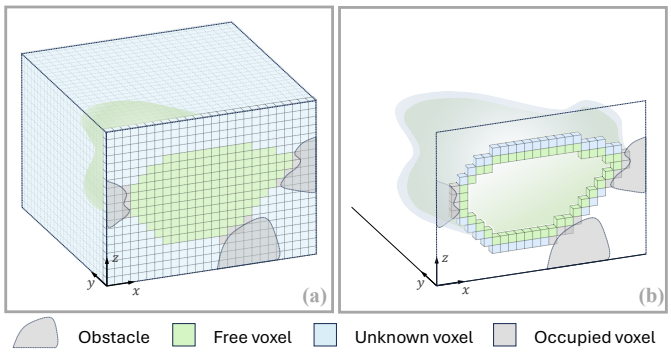

Fig. 2. (a) A uniform occupancy grid where each voxel is classified into one of three states: free, unknown, or occupied, represented by green, blue, and grey, respectively. For clarity, only the voxels along the cross-sectional plane are shown, with the remaining voxels represented by a simplified shape. (b) Our proposed boundary map, which consists of **boundary interior voxel**, **boundary exterior (unknown) voxel** and **boundary exterior (occupied) voxel**.

II. Methdology

A. The Boundary Map

We define the voxels that constitute the boundary map as **boundary voxels**. In the following, we present the categorization of boundary voxels based on their occupancy states and neighboring configurations. The boundary voxel is classified as **boundary interior voxel** $\mathbf{b}_{\mathrm{int}}$ and **boundary exterior voxel** $\mathbf{b}_{\mathrm{ext}}$, where the $\mathbf{b}_{\mathrm{ext}}$ is further classified into **boundary exterior (unknown) voxel** $\mathbf{b}_{\mathrm{ukn}}$ and **boundary exterior (occupied) voxel** $\mathbf{b}_{\mathrm{occ}}$. The definition of boundary voxels is formulated as follows:

$$
\begin{aligned}
\mathbf{b}_{\mathrm{int}} &\in \left\{ \mathbf{n} \;\middle|\; \begin{array}{l} \mathtt{occ}(\mathbf{n}) = \textit{free}, \\ \exists\, \mathtt{nbr}_6(\mathbf{n}) \in \{\textit{unknown}, \textit{occupied}\} \end{array} \right\}, \\
\mathbf{b}_{\mathrm{ukn}} &\in \left\{ \mathbf{n} \;\middle|\; \begin{array}{l} \mathtt{occ}(\mathbf{n}) = \textit{unknown}, \\ \exists\, \mathtt{nbr}_6(\mathbf{n}) = \textit{free} \end{array} \right\}, \\
\mathbf{b}_{\mathrm{occ}} &\in \{\mathbf{n} \mid \mathtt{occ}(\mathbf{n}) = \textit{occupied}\}\,.
\end{aligned}
\tag{1}
$$

where $\mathbf{n}$ denotes a voxel in the environment, $\mathtt{occ}(\mathbf{n})$ denotes the occupancy state of the voxel $\mathbf{n}$, and $\mathtt{nbr}_6(\mathbf{n})$ denotes occupancy states of the 6-connected neighbors of the voxel $\mathbf{n}$.

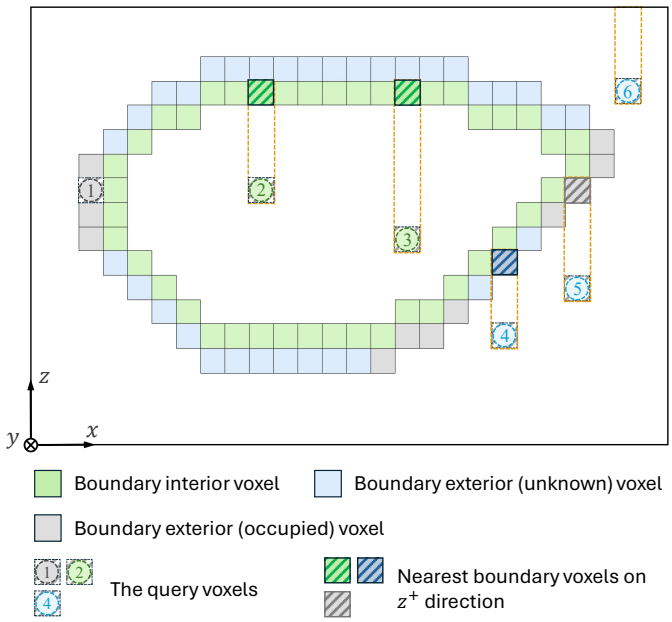

Fig. 3. Occupancy state determination based on our boundary map. The diagram shows the determination process for several example query voxels (nos. 1–5). If the query voxel is located exactly on the boundary map, the query voxel state can be determined by type of the boundary voxel directly (query voxel no. 1). Otherwise, we search along the z^+ direction (i.e., a representative direction for illustration propose) to find the nearest boundary voxel on the boundary map, denoted as $\mathbf{b}_{nn}$. If $\mathbf{b}_{nn}$ is a boundary interior voxel $\mathbf{b}_{\text{int}}$, the query voxel is determined as free (query voxels no. 2 and 3). If the $\mathbf{b}_{nn}$ is a boundary exterior voxel $\mathbf{b}_{\text{ext}}$(i.e., either unknown $\mathbf{b}_{\text{ukn}}$ or occupied $\mathbf{b}_{\text{occ}}$), the query voxel is determined as unknown (query voxels no. 4 and 5). If $\mathbf{b}_{nn}$ does not exist, the query voxel is determined as unknown too (query voxel no. 6).

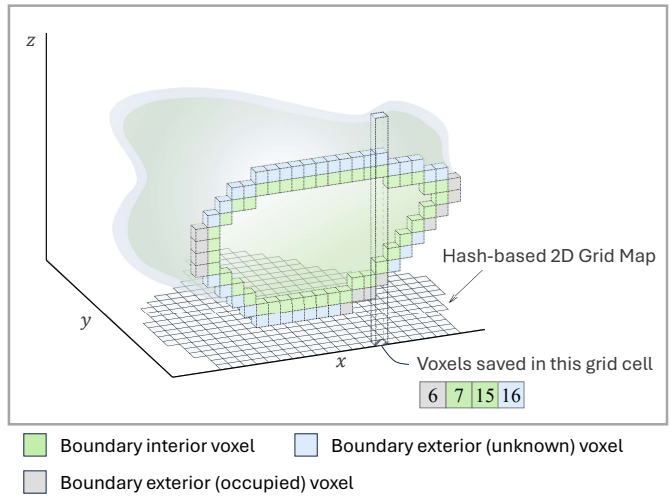

Fig. 4. The boundary map, which consists of boundary voxels in the three-dimension (3D), is maintained in a two-dimensional (2D) data structure. Without loss of generality, z-axis is selected as the projection axis for illustration purpose in the manuscript. The boundary map is projected along the z-axis onto the (x, y)-plane. Then a hash-based 2D grid map is maintained on the (x, y)-plane where each grid cell stores boundary voxels projected onto that cell. Each boundary voxel in the grid cell is represented by its z-coordinate and a type indicator indicating its type (i.e., $\mathbf{b}_{\text{int}}$, $\mathbf{b}_{\text{ukn}}$, and $\mathbf{b}_{\text{occ}}$). The numbers in the figure correspond to the z-coordinates of the boundary voxels.

B. Occupancy State Determination

To determine the occupancy state of a query voxel $\mathbf{q}$, we begin by searching for its nearest boundary voxel till the spatial extent of the environment along one of these six directions: $\{x^+, x^-, y^+, y^-, z^+, z^-\}$. The selected direction is referred to as the search direction $\mathcal{E}$, and it serves as an input to Algorithm 1. For illustrative purposes, we select z^+ as the search direction in the following discussion. We denote the nearest boundary voxel along the search direction $\mathcal{E}$ (e.g., z^+) as $\mathbf{b}_{nn} = (b^x_{nn}, b^y_{nn}, b^z_{nn}) \in \mathbb{Z}^3$, and define the corresponding nearest distance to the query voxel $\mathbf{q}$ as r_{min}, which is computed as: $r_{\text{min}} = |b^z_{nn} - q^z|$. This process is named as `FindNearestInDirection` function (Line 1). Then, the occupancy state of the query voxel $\mathbf{q}$ is determined based on the type of $\mathbf{b}_{nn}$ and the nearest distance r_{min}, as presented in Algorithm 1. An illustrative example is provided in Fig. 3.

Algorithm 1: Determine Occupancy State

Input: Query voxel $\mathbf{q}$, Search direction $\mathcal{E}$
Output: Occupancy state of $\mathbf{q}$: *free*, *occupied*, or *unknown*

1 $\mathbf{b}_{nn}, r_{\text{min}} \leftarrow$ `FindNearestInDirection`$(\mathbf{q}, \mathcal{E})$;
2 **if** $\mathbf{b}_{nn} \neq$ `null` **then**
3 **if** $\mathbf{b}_{nn} = \mathbf{b}_{\text{int}}$ **then**
4 **return** *free*;
5 **end**
6 **else**
7 **if** $\mathbf{b}_{nn} = \mathbf{b}_{\text{occ}}$ ***and*** $r_{\text{min}} = 0$ **then**
8 **return** *occupied*;
9 **end**
10 **end**
11 **end**
12 **return** *unknown*;

C. Data Structure

In this section, we first introduce a novel data structure designed to maintain the boundary map. Based on the proposed data structure, we describe the algorithmic implementation for efficiently determining the occupancy state of an arbitrary location in the environment (i.e., performing a map query).

We begin by introducing the concepts of the projection axis and the projection plane. One of the x, y, or z axes is designated as the projection axis. The plane orthogonal to this axis is referred to as the projection plane, defined by the remaining two axes in their natural sequential order. To facilitate explanation, we use z as the projection axis (the same as in II-B), and accordingly, the projection plane corresponds to the (x, y)-plane.

We project all boundary voxels along the z-axis onto the (x, y)-plane. Then we maintain a hash-based 2D grid map on this plane, where each grid cell stores boundary voxels projected onto it. The boundary voxels stored in a 2D grid cell are organized by an array and are sorted based on their z-coordinates. In other words, a grid cell $\mathbf{c} = (c^x, c^y) \in \mathbb{Z}^2$ of the 2D grid map maintains all boundary voxels that share the same (x, y)-coordinates as (c^x, c^y).

For the boundary voxel saved in the 2D grid cell, it suffices to store its z-coordinate and an indicator of its type. Each boundary voxel belongs to one of three types: boundary

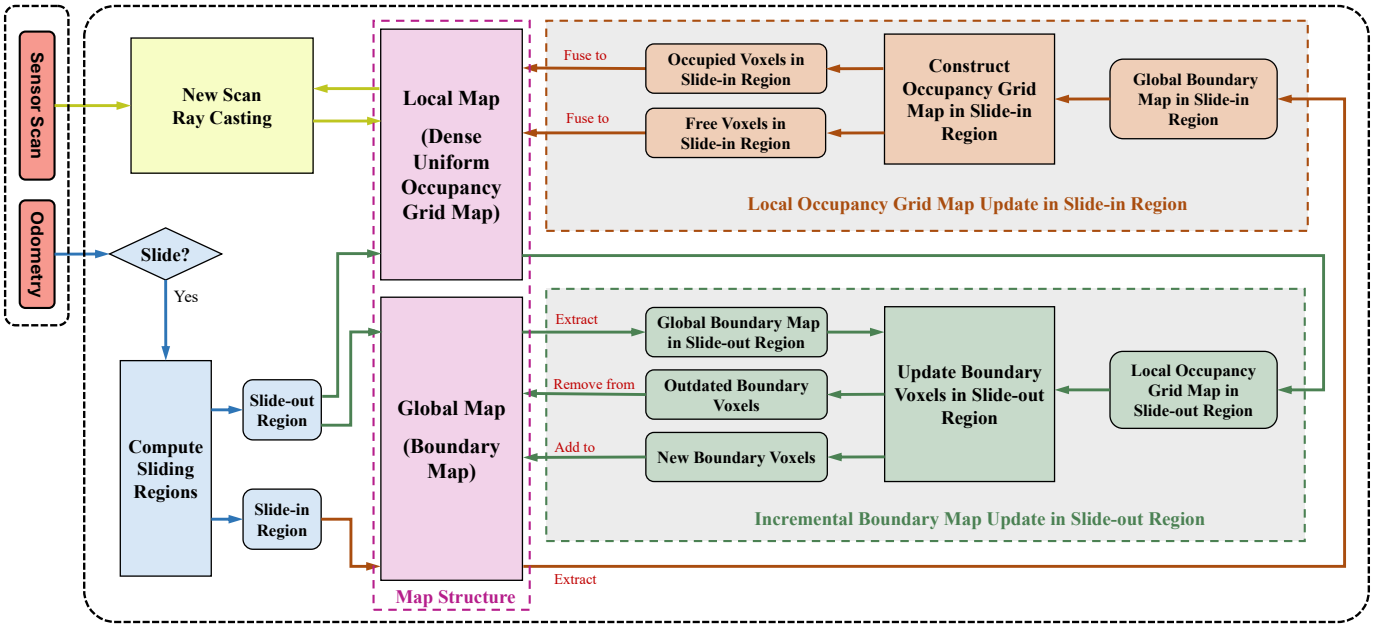

Fig. 5. Overview of the proposed global-local mapping framework. The system takes odometry and sensor scans as inputs. The pink block illustrates the map structure, consisting of a global map (the low-dimensional boundary map) and a local map (a dense uniform occupancy grid map). The mapping framework is updated once every new odometry and sensor data arrive. For each point in the new scan, the **New Scan Ray Casting** performs ray casting and probabilistic occupancy update on the local occupancy grid map. The new odometry is used to determine if the local map needs to slide and, if so, compute the slide-in region and slide-out region of the local map. If a slide occurs, the occupancy state within the slide-out region of the local map is updated to the global map by the **Incremental Boundary Map Update** (green block), while the occupancy state within the slide-in region of the local map is loaded from the global map via the **Local Occupancy Grid Map Update** (brown block).

interior voxel $\mathbf{b}_{\text{int}}$, boundary exterior (unknown) voxel $\mathbf{b}_{\text{ukn}}$ or boundary exterior (occupied) voxel $\mathbf{b}_{\text{occ}}$. We represent each boundary voxel using a 32-bit integer (`Int32`): the lower 30 bits encode the z-coordinate, and the upper 2 bits are sufficient to encode the boundary voxel type.

To determine the occupancy state of a query voxel $\mathbf{q} = (q^x, q^y, q^z) \in \mathbb{Z}^3$, we first lookup the hash table for the 2D grid cell $\mathbf{c}_q = (q^x, q^y) \in \mathbb{Z}^2$, which having the same (x, y)-coordinates with the query voxel $\mathbf{q}$. If the 2D grid cell $\mathbf{c}_q$ does not exist in the hash table, it implies that this grid cell $\mathbf{c}_q$ contains no boundary voxels, which indicates $\mathbf{b}_{nn}$ does not exist. In this case, we assign a `null` flag to $\mathbf{b}_{nn}$. On the other hand, if the grid cell $\mathbf{c}_q$ is found in the hash table, we proceed by retrieving the array stored in the grid cell $\mathbf{c}_q$. This array contains all boundary voxels in the environment whose (x, y)-coordinates equal (q^x, q^y). Given that the z-axis is selected as the projection axis, the search direction $\mathcal{E}$ can be chosen as either z^+ or z^-. Since the voxels in each array are pre-sorted along the z-axis, the $\mathbf{b}_{nn}$ can be located efficiently using binary search. Specifically, if the search direction is z^+, $\mathbf{b}_{nn}$ is the first voxel in the array whose z-coordinate is greater than or equal to q^z. Conversely, if the search direction is z^-, $\mathbf{b}_{nn}$ is the first voxel whose z-coordinate is less than or equal to q^z. When search direction is z^+, the $\mathbf{b}_{nn}$ does not exist if the z-coordinates of all boundary voxels saved in this grid cell $\mathbf{c}_q$ are less than q^z. On the other hand, when the search direction is z^-, this is the case when the z-coordinates of all boundary voxels saved in $\mathbf{c}_q$ are greater than q^z. In both cases, we assign a `null` flag to $\mathbf{b}_{nn}$. This procedure corresponds to the implementation of the `FindNearestInDirection` function in Algorithm 1 (Line 1). Once $\mathbf{b}_{nn}$ is identified, the occupancy state of the query voxel $\mathbf{q}$ can be determined by following the remaining steps outlined in Algorithm 1 (Line 2–12).

In occupancy mapping, determining the occupancy state of a given location in the environment is commonly referred to as a query operation. Accordingly, we refer to the above algorithm as the query algorithm in our method.

D. Global-local Mapping Framework

We propose a global-local mapping framework designed to support updates of the boundary map from the real-time sensor measurements. The framework integrates a robot-centric local map with the global map. Specifically, the local map is a fixed-size occupancy grid map centered on the vehicle's current position, maintaining occupancy information around the robot. The global map is a boundary map, maintaining occupancy information outside the local map region.

When a new sensor scan is received, along with the vehicle's current position, the framework is updated accordingly. The occupancy information in the local map is updated by the new scan via the ray casting technique [24]. The received vehicle's position determines whether the local map needs to slide. If the vehicle moves beyond a certain threshold distance from the current local map center, a local map sliding is triggered. For the region slides out from the local map, the occupancy information is represented by boundary voxels and incrementally updated into the global map. For the region slides into the local map, the occupancy states are loaded from

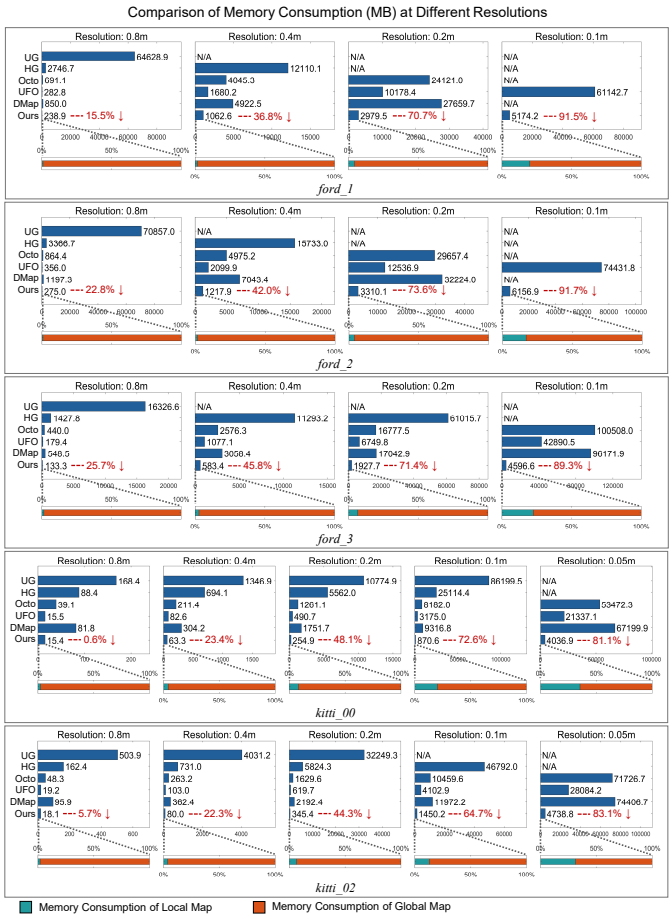

Fig. 6. Memory consumption (in MB) of UG (Uniform Grid), HG (Hash Grid), Octo (Octomap), UFO (UFOMap), D-Map, and our method. The percentage shown in red indicates the memory reduction of our method relative to the best of the rest methods in comparison. For our method, we also show the memory breakdown of the local and global maps, respectively. N/A indicates that the method failed due to memory consumption exceeding the limit.

the global map and fused into the local map. An overview this mapping framework is provided in Fig. 5.

III. Experiments and Applications

A. Benchmark Experiments

Extensive benchmark experiments were conducted to evaluate the performance of our mapping framework against several state-of-the-art methods, including Uniform Grid (UG) [19], Hash Grid (HG) [20], Octomap (Octo) [22], UFOMap (UFO) [23], and D-Map [25]. Five large-scale sequences *ford_1*, *ford_2*, *ford_3*, *kitti_00* and *kitti_02* from two public datasets [26, 27] were selected for evaluation. The memory consumption results are presented in Fig. 6. The update time and query time results are reported in Table I and in Table II, respectively, with the best and second-best results in each setting highlighted using distinct tint colors.

B. Real-World Application

Experiment Setup: Our MAV platform is equipped with a semi-solid state LiDAR, the Livox MID-360, a flight control unit (FCU) running PX4 Autopilot [28]. Additional system components are integrated as follows. For MAV localization, we utilize a modified version [29] of FAST-LIO2 [30], which provides high-accuracy state estimation at a frequency of 100 Hz. For trajectory tracking and control, we employ the on-manifold model predictive controller proposed in [31]. Several long-range goals were assigned during the mission. More

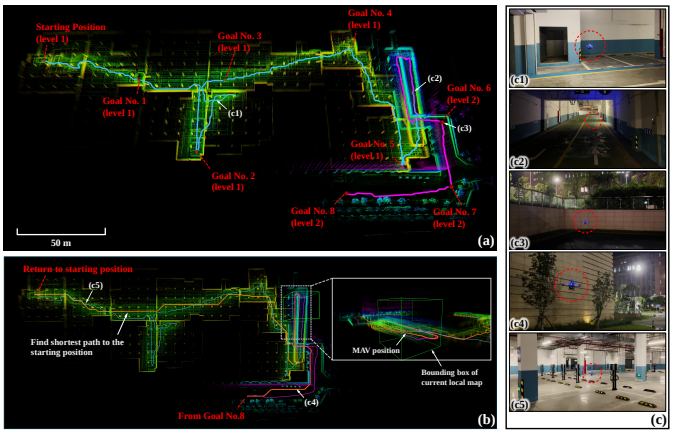

Fig. 7. Illustration of a long-range autonomous navigation task conducted in a multi-level building. Starting from an initial position, the MAV sequentially navigated to a series of manually defined goals, each specified after the previous one was reached. All goals were intentionally placed beyond the MAV's current local map to enforce long-range navigation. In total, eight goals were assigned and visited. After reaching the final goal (Goal No.8), the MAV was instructed to autonomously plan and execute a return trajectory to the starting position. (a) Locations of the eight goals. Goals No.1–5 are located on level 1, while Goals No.6–8 are on level 2. The trajectory is color-coded by altitude. (b) Return trajectory from Goal No.8 to the starting position, highlighted in orange. The MAV followed the shortest path computed using the A* search algorithm based on the proposed mapping framework. (c) Sample images of the MAV captured during the mission, with their corresponding locations on the trajectory marked in (a) and (b).

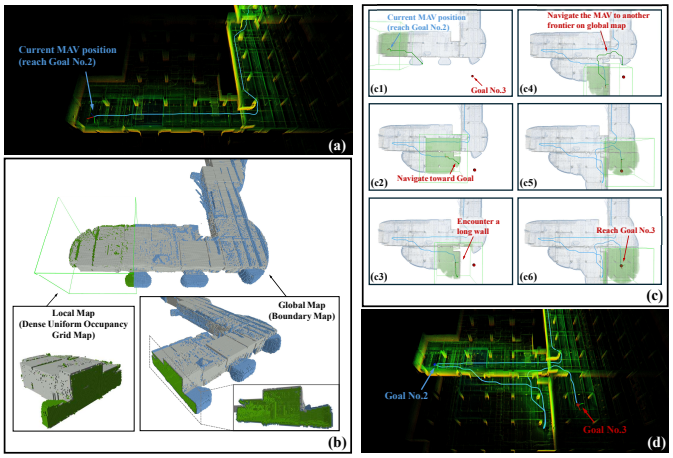

Fig. 8. Illustration of how the MAV navigates itself to a long-range goal, using the segment from Goal No.2 to Goal No.3 as an example. (a) The accumulated point cloud and trajectory when reaching Goal No.2. (b) Visualization of the corresponding mapping framework at this moment, including the global map (i.e., boundary map) and the local map. The free, unknown and occupied voxels are colored by green, blue and grey, respectively. (c) Autonomous navigation process toward Goal No.3. The red dot marks the location of Goal No.3. The green dot indicates the selected target on the frontier. The green line shows the computed A* path to the frontier target. For clarity, both the global and local maps are rendered semi-transparent. (d) Accumulated point cloud and trajectory upon reaching Goal No.3.

TABLE I
COMPARISON OF MAP UPDATE TIME (MS) AT DIFFERENT RESOLUTIONS.

	ford_1				*ford_2*				*ford_3*			
Resolution (m)	0.8	0.4	0.2	0.1	0.8	0.4	0.2	0.1	0.8	0.4	0.2	0.1
Uniform Grid	11.56	×	×	×	11.89	×	×	×	**9.74**	×	×	×
Hash Grid	12.08	33.16	×	×	11.38	30.46	×	×	12.36	36.04	196.23	×
Octomap	80.01	176.94	498.39	×	77.14	163.72	461.71	×	80.80	173.28	466.49	1353.29
UFOMap	61.69	119.20	246.68	511.86	58.39	113.10	278.57	480.39	62.60	122.40	249.38	528.85
D-Map	13.60	53.83	109.16	×	13.38	49.55	92.71	×	14.18	57.60	117.86	493.75
Ours	**10.98**	**24.89**	**71.34**	**219.59**	**10.58**	**23.61**	**67.69**	**207.78**	12.04	**26.59**	**75.50**	**209.37**

	kitti_00					*kitti_02*				
Resolution (m)	0.8	0.4	0.2	0.1	0.05	0.8	0.4	0.2	0.1	0.05
Uniform Grid	20.42	35.41	67.39	**209.19**	×	19.92	35.80	71.61	×	×
Hash Grid	21.97	40.37	96.56	342.80	×	21.70	42.56	89.32	274.34	×
Octomap	133.20	251.77	509.20	1157.82	3355.46	139.04	261.07	541.39	1240.35	3706.38
UFOMap	120.81	228.12	438.72	867.68	1701.42	124.23	236.93	457.62	906.31	1771.99
D-Map	**11.86**	**25.61**	**48.89**	226.12	1166.79	**11.78**	**28.56**	**55.68**	**222.26**	938.94
Ours	21.09	36.54	79.47	245.07	**794.19**	20.72	38.13	83.76	263.03	**860.78**

TABLE II
COMPARISON OF AVERAGE MAP QUERY TIME (NS) AT DIFFERENT RESOLUTIONS.

	ford_1				*ford_2*				*ford_3*			
Resolution (m)	0.8	0.4	0.2	0.1	0.8	0.4	0.2	0.1	0.8	0.4	0.2	0.1
Uniform Grid	10510.9	×	×	×	25133.8	×	×	×	**41.18**	×	×	×
Hash Grid	86.28	123.12	×	×	84.11	132.83	×	×	101.73	288.19	173169.0	×
Octomap	373.56	512.90	606.18	×	370.64	490.14	634.95	×	425.05	487.36	636.77	137039.0
UFOMap	174.16	227.98	314.85	39516.2	182.43	234.03	313.96	82606.9	155.77	211.11	295.76	567.18
D-Map	654.38	784.49	903.17	×	680.72	820.07	899.19	×	616.85	758.78	859.91	142607.3
Ours	**78.86**	**83.20**	**81.32**	**45.13**	**80.36**	**79.83**	**77.39**	**45.85**	79.07	**88.78**	**98.03**	**56.86**

	kitti_00					*kitti_02*				
Resolution (m)	0.8	0.4	0.2	0.1	0.05	0.8	0.4	0.2	0.1	0.05
Uniform Grid	**30.87**	**30.43**	**36.30**	28773.4	×	**32.58**	**41.82**	**46.95**	×	×
Hash Grid	65.81	75.88	87.83	139.59	×	63.97	65.10	90.04	151.89	×
Octomap	203.19	253.33	318.83	431.84	469013.6	183.22	227.12	304.98	380.92	638444.8
UFOMap	86.53	128.51	159.48	203.38	264.28	91.67	127.28	149.64	205.21	254.30
D-Map	496.90	477.45	546.25	666.67	84924.8	372.27	423.10	479.59	597.92	683656.5
Ours	32.67	36.04	36.42	**44.28**	**48.23**	55.42	62.96	62.62	**56.78**	**58.80**

Note: × indicates that the method failed due to memory consumption exceeding the limit.

details are presented in Fig. 7.

Autonomous Navigation: Leveraging the occupancy information provided by the proposed mapping framework, the MAV computes an A* path in free space toward a frontier that is close to the target. A representative example of this navigation process is illustrated in Fig. 8, which depicts the MAV exploring the unknown space to reach the Goal No.3. In this scenario, the MAV was directed toward a frontier on the global map upon encountering a long wall, then eventually reached Goal No.3. Such a mechanism prevents the vehicle from falling into local traps, and exemplifies the necessity of maintaining a global occupancy map for reliable long-range autonomy in such complex environment.

IV. DISCUSSION AND FUTURE WORK

In this paper, we introduced a novel method for occupancy grid mapping. Our method maintains only the two-dimensional (2D) boundary voxels rather than all grid voxels in the three-dimensional (3D) space. This low-dimensional representation significantly reduces memory consumption, particularly in high-resolution and large-scale mapping scenarios. We then propose a dedicated data structure and an algorithm that enable fast and constant-time occupancy state queries. Moreover, we developed a global-local mapping framework and a corresponding update method, achieving highly efficient map updates from real-time sensor measurements. Extensive experimental results demonstrated that our method outperforms state-of-the-art occupancy mapping techniques by achieving substantial memory reduction while simultaneously preserving high query and update efficiency.

In the future, the compactness of the boundary map can be further exploited to address a critical challenge in multi-agent robot systems: the synchronization of occupancy information across multiple agents. Due to limited real-time communication bandwidth, transmitting large occupancy grid maps between agents is inefficient. In contrast, the compactness of our boundary map makes it particularly well-suited for this task. To enable synchronization of occupancy information by transmitting the boundary map, a method for merging different boundary maps from other agents is necessary. This will be explored as part of our future work.

References

[1] T. Rouček, M. Pecka, P. Čížek, T. Petříček, J. Bayer, V. Šalanskỳ, D. Heřt, M. Petrlík, T. Báča, V. Spurnỳ, *et al.*, "Darpa subterranean challenge: Multi-robotic exploration of underground environments," in *Modelling and Simulation for Autonomous Systems: 6th International Conference, MESAS 2019, Palermo, Italy, October 29–31, 2019, Revised Selected Papers 6*. Springer, 2020, pp. 274–290.

[2] M. Tranzatto, T. Miki, M. Dharmadhikari, L. Bernreiter, M. Kulkarni, F. Mascarich, O. Andersson, S. Khattak, M. Hutter, R. Siegwart, *et al.*, "Cerberus in the darpa subterranean challenge," *Science Robotics*, vol. 7, no. 66, p. eabp9742, 2022.

[3] S. Kawatsuma, M. Fukushima, and T. Okada, "Emergency response by robots to fukushima-daiichi accident: summary and lessons learned," *Industrial Robot: An International Journal*, vol. 39, no. 5, pp. 428–435, 2012.

[4] O. SeungSub, H. Jehun, J. Hyunjung, L. Soyeon, and S. Jinho, "A study on the disaster response scenarios using robot technology," in *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. IEEE, 2017, pp. 520–523.

[5] L. Yoder and S. Scherer, "Autonomous exploration for infrastructure modeling with a micro aerial vehicle," in *Field and Service Robotics: Results of the 10th International Conference*. Springer, 2016, pp. 427–440.

[6] W. Tabib, K. Goel, J. Yao, C. Boirum, and N. Michael, "Autonomous cave surveying with an aerial robot," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 1016–1032, 2021.

[7] S. Isler, R. Sabzevari, J. Delmerico, and D. Scaramuzza, "An information gain formulation for active volumetric 3d reconstruction," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 3477–3484.

[8] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, and J. Nieto, "An efficient sampling-based method for online informative path planning in unknown environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1500–1507, 2020.

[9] B. T. Lopez and J. P. How, "Aggressive 3-d collision avoidance for high-speed navigation." in *ICRA*, 2017, pp. 5759–5765.

[10] F. Kong, W. Xu, Y. Cai, and F. Zhang, "Avoiding dynamic small obstacles with onboard sensing and computation on aerial robots," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7869–7876, 2021.

[11] Y. Ren, F. Zhu, W. Liu, Z. Wang, Y. Lin, F. Gao, and F. Zhang, "Bubble planner: Planning high-speed smooth quadrotor trajectories using receding corridors," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 6332–6339.

[12] Y. Ren, F. Zhu, G. Lu, Y. Cai, L. Yin, F. Kong, J. Lin, N. Chen, and F. Zhang, "Safety-assured high-speed navigation for mavs," *Science Robotics*, vol. 10, no. 98, p. eado6187, 2025.

[13] F. Yang, C. Cao, H. Zhu, J. Oh, and J. Zhang, "Far planner: Fast, attemptable route planner using dynamic visibility update," in *2022 ieee/rsj international conference on intelligent robots and systems (iros)*. IEEE, 2022, pp. 9–16.

[14] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon" next-best-view" planner for 3d exploration," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1462–1468.

[15] T. Dang, F. Mascarich, S. Khattak, C. Papachristos, and K. Alexis, "Graph-based path planning for autonomous robotic exploration in subterranean environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 3105–3112.

[16] B. Zhou, Y. Zhang, X. Chen, and S. Shen, "Fuel: Fast uav exploration using incremental frontier structure and hierarchical planning," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 779–786, 2021.

[17] C. Cao, H. Zhu, H. Choset, and J. Zhang, "Tare: A hierarchical framework for efficiently exploring complex 3d environments." in *Robotics: Science and Systems*, vol. 5, 2021, p. 2.

[18] B. Tang, Y. Ren, F. Zhu, R. He, S. Liang, F. Kong, and F. Zhang, "Bubble explorer: Fast uav exploration in large-scale and cluttered 3d-environments using occlusion-free spheres," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 1118–1125.

[19] M. C. M. H. P. Moravec, "Robot evidence grids," *CMU Robotics Institute Technical Report CMU-RI-TR-96-06*, 1996.

[20] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3d reconstruction at scale using voxel hashing," *ACM Transactions on Graphics (ToG)*, vol. 32, no. 6, pp. 1–11, 2013.

[21] 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.

[22] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, pp. 189–206, 2013.

[23] D. Duberg and P. Jensfelt, "Ufomap: An efficient probabilistic 3d mapping framework that embraces the unknown," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6411–6418, 2020.

[24] J. Amanatides, A. Woo, *et al.*, "A fast voxel traversal algorithm for ray tracing." in *Eurographics*, vol. 87, no. 3. Citeseer, 1987, pp. 3–10.

[25] Y. Cai, F. Kong, Y. Ren, F. Zhu, J. Lin, and F. Zhang, "Occupancy grid mapping without ray-casting for high-resolution lidar sensors," *IEEE Transactions on Robotics*, 2023.

[26] S. Agarwal, A. Vora, G. Pandey, W. Williams, H. Kourous, and J. McBride, "Ford multi-av seasonal dataset," *The International Journal of Robotics Research*, vol. 39, no. 12, pp. 1367–1376, 2020.

[27] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[28] P. Liu, C. Feng, Y. Xu, Y. Ning, H. Xu, and S. Shen, "Omninxt: A fully open-source and compact aerial robot with omnidirectional visual perception," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 10 605–10 612.

[29] F. Zhu, Y. Ren, F. Kong, H. Wu, S. Liang, N. Chen, W. Xu, and F. Zhang, "Decentralized lidar-inertial swarm odometry," *arXiv preprint arXiv:2209.06628*, 2022.

[30] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.

[31] G. Lu, W. Xu, and F. Zhang, "On-manifold model predictive control for trajectory tracking on robotic systems," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 9, pp. 9192–9202, 2022.