Memory-Efficient Boundary Map for Large-Scale Aerial Maritime Inspection

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

***Abstract*—Performing tasks in maritime infrastructures poses significant safety risks to human operators. Autonomous aerial robots offer a safer alternative. To achieve such autonomy, an efficient and robust representation of the environment is fundamental. Occupancy grid maps provide an effective solution; however, in large-scale environments, existing methods impose high memory and computational demands, rendering them unsuitable for deployment on resource-constrained aerial platforms. To tackle this, we utilize a novel 2D boundary voxel representation instead of explicitly representing entire 3D volumes. We term our method as boundary map. This low-dimensional representation significantly reduces memory consumption by orders of magnitude, as demonstrated in our benchmark experiments. Furthermore, we validate our framework through long-range autonomous navigation experiments in a GNSS-denied, multi-floor real-world environment. Our results demonstrate that the proposed efficient mapping framework has high potential for supporting complex, large-scale maritime inspection tasks where onboard memory and computational resources are limited.**

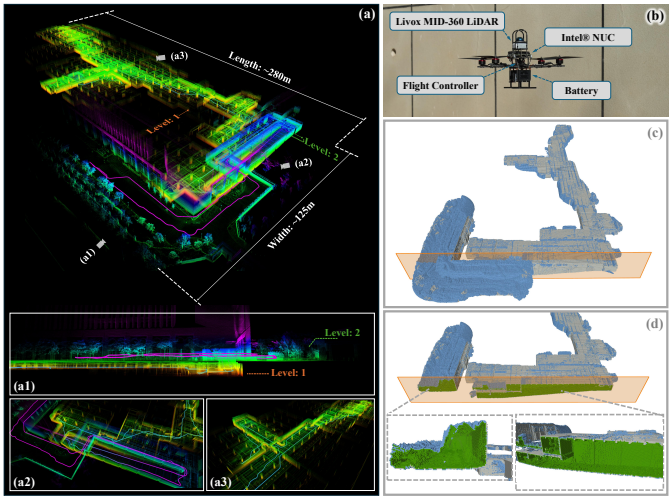

Fig. 1. MAV performing a long-range autonomous navigation task in an unknown multi-level building, utilizing the proposed mapping framework. (a) Accumulated point cloud and the MAV trajectory at the end of the task, also visualized from multiple viewpoints in (a1), (a2), and (a3). (b) Hardware configuration 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

Performing inspections in maritime infrastructures exposes human operators to substantial safety hazards, as they must often operate in confined, elevated, or structurally complex areas under harsh marine conditions. Autonomous aerial robots have emerged as a powerful alternative, offering safer, faster and more cost-effective inspection capabilities. However, these environments are typically large-scale and highly complex, posing significant challenges for the deployment of autonomous systems. Fundamental for collision avoidance, informed and reliable navigation and thorough inspection tasks is an accurate representation of the environment—whether locations are free, occupied, or unknown. Occupancy grid maps offer an efficient solution [1]–[4]. However, maritime infrastructures—such as large vessels, coastal logistics hubs, offshore oil rigs, and marine bridges—are typically vast in scale. In such settings, existing approaches can rapidly exhaust the limited memory resources available on aerial platforms. Therefore, a memory-efficient occupancy grid mapping framework is essential, serving as a critical foundation for large-scale autonomous operation and effective coverage of extensive maritime infrastructures.

Existing occupancy grid mapping methods can generally be divided into two categories: (i) grid-based methods [3, 5, 6] and (ii) octree-based methods [7, 8]. 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.

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 [5]), 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-

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

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. 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 **GNSS-denied**, **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.
- **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.

II. Methodology

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** $\mathbf{b}_{\text{int}}$ and **boundary exterior voxel** $\mathbf{b}_{\text{ext}}$, where the $\mathbf{b}_{\text{ext}}$ is further classified into **boundary exterior (unknown) voxel** $\mathbf{b}_{\text{ukn}}$ and **boundary exterior (occupied) voxel** $\mathbf{b}_{\text{occ}}$. The definition of boundary voxels is formulated as follows:

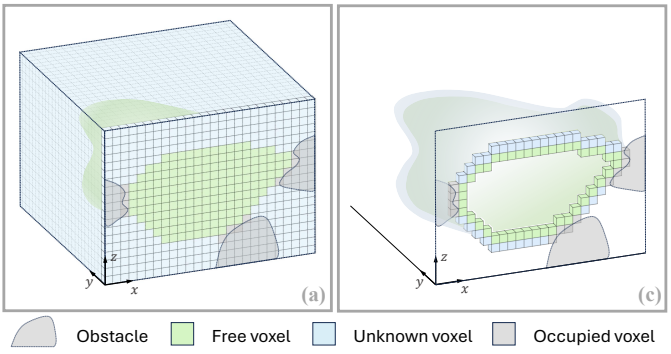

Fig. 2. (a) A uniform occupancy grid where each voxel is classified into one of three states: free, unknown, or occupied, represented by green, blue, and grey, respectively. For clarity, only the voxels along the cross-sectional plane are shown, with the remaining voxels represented by a simplified shape.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

Algorithm 1: Determine Occupancy State

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

$\mathbf{b}_{nn}, r_{\min} \leftarrow \texttt{FindNearestInDirection}(\mathbf{q}, \mathcal{E})$;
if $\mathbf{b}_{nn} \neq \texttt{null}$ **then**
  if $\mathbf{b}_{nn} = \mathbf{b}_{\text{int}}$ **then**
    return *free*;
  end
  else
    if $\mathbf{b}_{nn} = \mathbf{b}_{\text{occ}}$ ***and*** $r_{\min} = 0$ **then**
      return *occupied*;
    end
  end
end
return *unknown*;

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

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

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

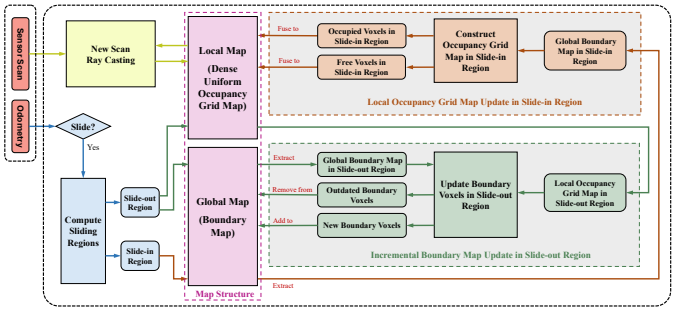

Fig. 3. Overview of the proposed global-local mapping framework.

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). 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.

B. 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 [9]. 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 the global map and fused into the local map. An overview this mapping framework is provided in Figure 3.

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) [5], Hash Grid (HG) [6], Octomap (Octo) [7], UFOMap (UFO) [8], and D-Map [10]. Five large-scale sequences *ford_1*, *ford_2*, *ford_3*, *kitti_00* and *kitti_02* from two public datasets [11, 12] were selected for evaluation. The memory consumption results are presented in Figure 4. The update time results are reported in I, with the best and second-best results in each setting highlighted using distinct tint colors.

Fig. 4. 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.

B. Real-World Autonomous Navigation

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 [13]. Additional system components are integrated as follows. For MAV localization, we utilize a modified version [14] of FAST-LIO2 [15], 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 [16]. Several long-range goals were assigned during the mission. More details are presented in Fig. 5.

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. 6, 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

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**

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

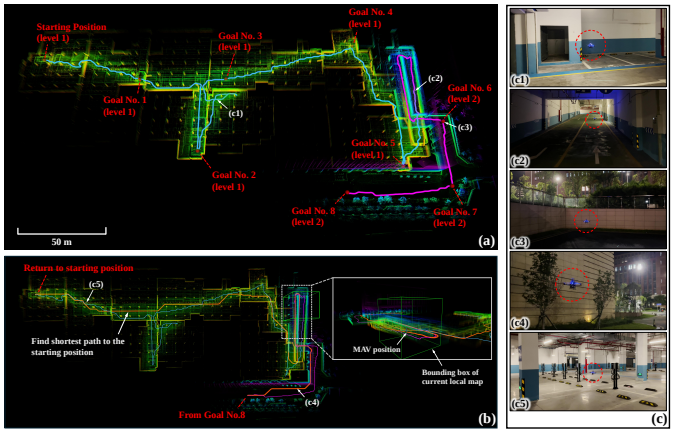

Fig. 5. Illustration of a long-range autonomous navigation task conducted in a multi-level building. Starting from an initial position, the MAV sequentially navigated to a series of manually defined goals, each specified after the previous one was reached. All goals were intentionally placed beyond the 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).

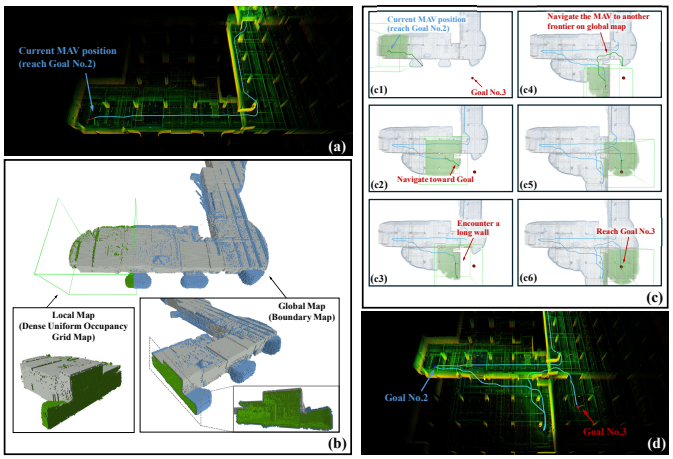

Fig. 6. Illustration of how the MAV navigates itself to a long-range goal, using the segment from Goal No.2 to Goal No.3 as an example. (a) The accumulated point cloud and trajectory when reaching Goal No.2. (b) Visualization of the corresponding mapping framework at this moment, including the global 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.

maintaining a global occupancy map for reliable long-range autonomy in such complex environment.

IV. DISCUSSION AND FUTURE WORK

In this work, we present a memory-efficient, real-time occupancy grid mapping framework for large-scale environments. This representation provides a foundational component upon which advanced, task-specific autonomous algorithms can be built, facilitating complex operations such as high-resolution inspection and active perception in large-scale maritime structures.

Furthermore, the proposed framework holds significant potential for extension to multi-agent swarm systems. Deploying collaborative swarms is an efficient solution for covering vast maritime infrastructures. By distributing tasks across a swarm of MAVs, operational efficiency can be significantly enhanced while mitigating the endurance limitations of individual platforms. To enable efficient collaboration, the synchronization of occupancy information across multiple agents is essential. However, due to limited real-time communication bandwidth, transmitting large volumetric grids between agents is inefficient. In contrast, the compactness of our boundary map makes it particularly well-suited for this task. We plan to exploit this property in future work by developing efficient methods to merge boundary maps from multiple agents, thereby enabling collaborative large-scale tasks.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.