Cooperative and Autonomous Mapping for Heterogeneous NAVs

Ruiwen Xu†, Yongtao Ou†, Hanjie Yu, Ziyi Zhang, Feng Shan, Weiwei Wu, and Junzhou Luo
School of Computer Science and Engineering, Southeast University, Nanjing, China
Emails: {raven, yongtao, yuhanjie, ziyi, shanfeng, weiweiwu, jluo}@seu.edu.cn

Abstract*—Nano unmanned aerial vehicles (NAVs) have been increasingly used for mapping due to their advantages in weight and size, such as search, rescue and reconnaissance in confined areas. However, because the resources carried by NAVs are extremely limited, it is a challenge to achieve efficient autonomous mapping in 3D space. To address this challenge, this paper proposes a new autonomous mapping model and system architecture with heterogeneous NAVs. Firstly, this paper introduces an enhanced OctoMap-based map model designed for resource-constrained NAVs. In comparison to the conventional OctoMap map model, the proposed model exhibits a significant reduction of 85.94% in memory usage while maintaining the same number of voxel blocks. In addition, We propose an active exploration algorithm based on Next Best View (NBV) theory, which can achieve agile and efficient autonomous mapping. Furthermore, a merging rays strategy and a dynamic adjustment of Edge-NAV positions strategy are proposed to improve the communication quality between heterogeneous NAVs. The map model and exploration algorithms are deployed on real-world NAVs, and the correctness and effectiveness of the proposed algorithm are validated through extensive simulations and real-world experiments, accomplishing more than the exploration completion rate of 80% with less than memory pool usage of 60%.

***Index Terms*—Cooperative NAVs, Autonomous Mapping, Computing Offloading, OctoMap**

I. Introduction

Three-Dimensional (3D) Mapping by Unmanned Aerial Vehicle (UAV) is a technique that is used to create useful 3D maps of hard-to-reach areas. Due to the diverse range of applications scenarios, it attracts a lot attentions recently [1]–[7]. Autonomous 3D mapping has been widely employed in open areas to do topographic mapping, geological surveys, and disaster response planning [8]–[11].

In compact and obstacle-rich environments, 3D mapping is also needed, for example, rescue teams need to map a culvert or a collapsed building immediately to make a rescue plan, managers need to periodically map a factory workshop to ensure moving parts of the machines are in a good position, commanders need to map a hostile build quickly to support fast military action. There is currently very few research [12] partly because flying in such an environment is challenging.

Nano unmanned aerial vehicles (NAVs) are UAVs with diameters of only a few centimeters and weighing only a few dozen grams. NAV is suitable to carry out 3D mapping tasks in the compact space or the obstacle-rich environment due to their size and weight advantages [2]. However, it has a tight resource budget, NAV 3D mapping faces three limitations: low computation capability, limited onboard memory, and low payload capacity [2], [13]. Hence, it is important to deploy multiple NAVs that can cooperate with each other to complete the 3D mapping task. Fig. 1 shows the system model of the heterogeneous NAVs cooperating with each other to autonomously mapping in this paper.

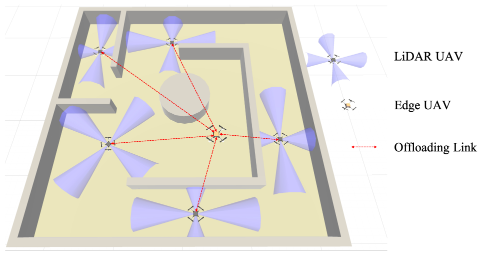


Fig. 1: Cooperatively and autonomously mapping for heterogeneous NAVs

Previous research works on UAV 3D mapping generally study active exploration [14]–[16], octree optimization [17], [18], image identification [19], deep learning [20]. However, we summarize that there are still some important challenges for NAV 3D mapping.

Challenge 1: although the OctoMap [21] saves memory in mapping, however it is still too memory-intense to directly deploy onboard on NAVs. OctoMap is an efficient probabilistic mapping framework, it is a significant improvement over the raster map. In recent years, OctoMap has undergone numerous improvements in terms of memory and computational cost [22], [23]. The data structure used by Octomap to store the nodes is octree, requiring an amount of memory space. So, directly deploying the octree data structure is unsuitable for more resource-constrained NAVs.

Challenge 2: current active exploration strategies for NAVs focus more on information profit and exploration efficiency but pay less attention to the impact of memory constraints on map storage. The octree requires modification to ensure quick collection of environmental information in unfamiliar 3D environments and adapt to resource-constrained NAVs. However, a dilemma arises where, on the one hand, exploring unfamiliar areas quickly requires the use of more nodes, leading to memory overruns in resource-constrained NAVs. On the other hand, if NAV memory is aggressively managed, it may result in overly conservative exploration, limiting coverage to a local area. Therefore, finding a trade-off between the stricter memory usage management and the faster updates of octree maps is a crucial question that warrants careful consideration.

Challenge 3: it is hard to coordinate multiple heterogeneous NAVs to cooperatively explore their surroundings and construct a combined 3D maps. It is important to note that the energy sources of NAVs are limited, making them less possible to explore for long periods of time and cover a large area. Thus, cooperative efforts among NAVs are necessary to construct 3D maps of larger areas. Moreover, the payload capacity of a NAV is normally small, heterogeneous NAVs cooperation must be considered. Tasks should be shared among the heterogeneous NAVs to reduce the load of a single NAV and prevent rapid energy consumption. To achieve this, not only an effective communication among NAVs is essential, but also a unified cooperative framework for heterogeneous NAV teams is crucial.

To cope with the above three challenges, improvements are made to the classic OctoMap; a new active exploration algorithm is proposed; a generic edge-assisted computational offloading framework is designed for heterogeneous NAVs cooperation. Additionally, real-world experiments are conducted to validate the proposed ideas and algorithms. The main contributions of this paper can be summarized as follows:

- We formulate a novel problem, that is mapping a compact and obstacle-rich environment with a team of resource-constrained heterogeneous NAVs.
- Based on the existing method OctoMap, we improve and proposed the TinyOctoMap, which focuses on a much more memory-restricted situation such as in the NAV. In TinyOctoMap, we optimize the data structure and memory allocation in performing the 3D mapping.
- We design an active exploration algorithm for autonomous mapping. We consider both information profit and memory-reducing profit, taking advantage of the feature that octrees could be pruned. Our algorithm allows NAVs to explore for a longer time.
- We propose a heterogeneous NAVs framework for cooperative mapping. The framework allows NAV to offload mapping tasks to the other NAV. We propose strategies to enhance the communication throughput. We implement cross-platform and cross-protocol communication within the framework for more efficient collaboration.
- We implement and deploy the autonomous mapping system on Crazyflie 2.1. Extensive simulation and real-world experiments are conducted to verify the proposed algorithms and framework mentioned in the paper.

The rest of the paper is organized as follows. Section II introduces the basic knowledge of UAV 3D mapping and communication. Section III and IV present the detailed design of the autonomous navigation and edge-assisted offloading framework. Section V gives the implementation details. Experiment results are discussed in Section VI. Section VII concludes this paper.

II. Preliminary

Octree is a tree-like data structure used to describe 3D space. It divides the space into a series of voxels of different sizes, each of which is called a node. Each node has eight child nodes, which represent eight sub-regions. The nodes at different levels of the octree occupy the same memory. An example of OctoTree physical and memory structure is given in Fig. 2.

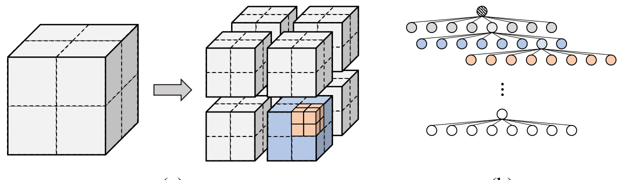


Fig. 2: OctoTree physical and memory structure

Each node has two known states: occupied and free. The node state is determined by the occupancy probability, which is in the logarithmic form to describe the relative likelihood of occupancy. The occupation probability of any leaf node m_i within the octree is usually updated by the RayCast method using sensing data obtained by Light Detection and Ranging (LiDAR). To simplify the representation of occupancy probability, instead of storing the probability values directly, the nodes of the octree store the log probability values to describe the confidence states of the nodes while avoiding the loss of accuracy caused by too small or too large probability values. This gives the updated equation for the occupation probability:

$$l(m_i \mid z_{(1:t)}) = l(m_i \mid z_{(1:t-1)}) + l(m_i \mid z_t) \tag{1}$$

where $l(m_i \mid z_t)$ represents the amount of change in the logarithmic occupation modification of the nodes associated with each LiDAR sensing data. Specifically, the value of $l(m_i \mid z_t)$ is determined by whether the laser can cross the node or not. When all sub-nodes are in the same state, they can be pruned to reduce memory usage. This feature is quite suitable for resource-constrained NAVs.

III. Design of Autonomous Exploration Algorithm

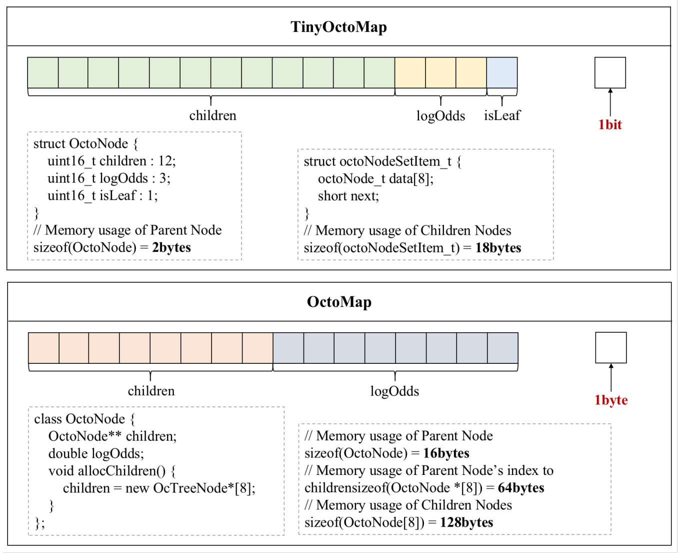
Since the memory is highly constrained in NAVs, this chapter first optimizes the classic OctoMap model to compress information storage, then the classic active exploration strategy is improved to according to memory usage.

A. Design of memory-adapted map model: TinyOctoMap

The classic open-source implementation of the OctoMap has been iterating for years. However, these improvements have also led to an ever-expanding memory occupation of the basic data structure. For resource-sensitive NAVs, the memory occupation of the basic data structure should be minimized.

This paper therefore propose TinyOctoMap, and the comparison with classical open-source implementation of the OctoMap are shown in Fig.3. As the most important in the octree structure, the memory occupied per unit element of the tree node directly affects the resource utilization of the whole map. In order to achieve efficient resource utilization in indoor Nano-UAVs navigation scenarios, TinyOctoMap makes the following optimizations in the design of tree nodes:

1) Subnode index children: Since TinyOctoMap uses the memory pool to manage all tree nodes, and the sibling nodes are stored in consecutive nodes, the child node location only keeps the array index of the first child node in the memory pool.
2) Node occupation level logOdds: OctoMap model for node occupation state evaluation, usually using log-odds method, and floating-point value is used. Instead, the unsigned integer stored occupancy level is used in TinyOctoMap.
3) Use a bitfield to store members. Considering the range of child node indexes and occupation level, the members can be stored in a 16-bit unsigned integer using a structured bit field to further compress the space occupied by a single node



```
struct OctoNode {
    uint16_t children : 12;
    uint16_t logOdds : 3;
    uint16_t isLeaf : 1;
}
// Memory usage of Parent Node
sizeof(OctoNode) = 2bytes
```

```
struct octoNodeSetItem_t {
    octoNode_t data[8];
    short next;
}
// Memory usage of Children Nodes
sizeof(octoNodeSetItem_t) = 18bytes
```

```
class OctoNode {
    OctoNode** children;
    double logOdds;
    void allocChildren() {
        children = new OcTreeNode*[8];
    }
};
```

```
// Memory usage of Parent Node
sizeof(OctoNode) = 16bytes
// Memory usage of Parent Node's index to
childrensizeof(OctoNode *[8]) = 64bytes
// Memory usage of Children Nodes
sizeof(OctoNode[8]) = 128bytes
```

Fig. 3: Memory usage analysis of TinyOctoMap and OctoMap

As a result, TinyOctoMap significantly reducing the memory occupation of tree nodes, makes it possible to deploy the map model on resource-constrained nano UAVs. Limited by the number of bits of Subnode indexes specified in bit fields in tree nodes, TinyOctoMap can record a maximum set of 4096 tree nodes, occupying about 65KB of memory space. If the resolution of the octree is set to 4cm, TinyOctoMap can record about $51.2m^2$ of obstacle surface area, which seems enough to cope with the demand of indoor simple autonomous navigation scenarios, and also meet the memory limit of the experimental platform used in this paper.

B. Design of Active Exploration

UAV autonomous exploration algorithms are usually based on Next Best View (NBV) theory. Using strategies such as boundary point extraction, a candidate set of next waypoints is selected from the current location. Then, an evaluation function is established to evaluate the candidate waypoints based on the map coverage, navigation cost, state uncertainty, and other metrics. Finally, the optimal waypoint for the next moment at the current position is selected based on the greedy strategy, so as to continuously explore the map space and maximize the benefits of exploration.

From the above steps, we can concluded that design a good profit evaluation function is critical for designing a good active exploration strategy. Our core idea of designing the profit evaluation function for memory-constrained nano UAVs is that let this evaluation function be aware of the memory usage. More specifically, a good profit evaluation function should balance between exploration efficiency and memory constraint. This subsection therefore proposes 1) the information profit related to exploration efficiency, 2) octree pruning profit related to memory constraint, and 3) how the two profit functions combined to improve active exploration strategy.

1) Information Profit: Information profit refers to the amount of information that the UAV updates map information by scanning unknown areas with LiDAR sensor after reaching a candidate waypoint $q(q \in Q)$. How to evaluate the information profit of a candidate waypoint without first reaching it? To solve this problem, we integrate the OctoMap and LiDAR detection characteristics, estimate the volume of the unknown area that the UAV can detect after reaching a candidate waypoint by building a probabilistic model, and then evaluate the information profit.

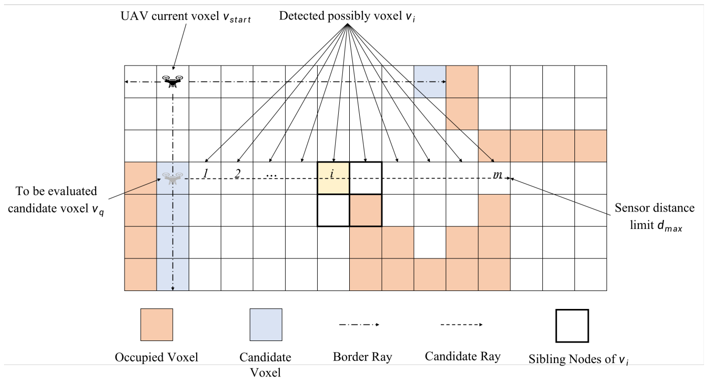

UAV current voxel v_{start}
Detected possibly voxel v_i
1
2
...
i
m
To be evaluated candidate voxel v_q
Sensor distance limit d_{max}
Occupied Voxel
Candidate Voxel
Border Ray
Candidate Ray
Sibling Nodes of v_i

Fig. 4: Candidates evaluation

Fig. 4 gives the possible voxels passed by the UAV in the detection direction L at the candidate waypoint q. For any candidate waypoint, q within the candidate waypoints set Q, the information profit that can be obtained in direction L is as follows:

$$W_{info}(q, L) = \sum_{i=1}^{m}(\omega_{info}(i)), q \in Q, \tag{2}$$

where m denotes the number of voxels that may be explored in the detection direction L, and $\omega_{info}(i)$ denotes the information profit from the ith voxel in the detection direction L.

In order to represent the information profit more precisely for unknown voxel i, we define two important concepts: (a) the *occupation probability* and (b) the *reachability probability* of the unknown voxel i.

The occupation probability refers to the probability that the voxel is occupied. Considering the occupation status of the sibling nodes of voxel i and the global information together, the occupation probability of unknown voxel i is calculated as follows:

$$\begin{gathered} p_{occ}\left(i\right) = p_{glabal} + \left(n_{occ} - n_{free}\right) * p_{inc}, \\ p_{glabal} \in [0,1], p_{inc} \in [0,1], p_{occ} \in [0,1], \end{gathered} \tag{3}$$

where p_{glabal} is a constant for the global occupation probability, n_{occ} is the number of nodes known to be in the occupied state among the sibling nodes of voxel i, n_{free} is the number of nodes known to be free among the sibling nodes of voxel i, and p_{inc} is the parameter for the influence of each sibling node on the state of voxel i, which is also a constant.

The reachability probability indicates the probability that the voxel can be detected from a candidate waypoint. It is important to note that the laser cannot penetrate the occupied block. Combining the LIDAR properties and the occupation probability defined earlier, the reachability probability of the unknown voxel i can be defined as:

$$p_{arr}\left(i\right) = \sum_{j=1}^{i-1} \left(1 - p_{occ}(j)\right). \tag{4}$$

Based on the probabilistic model defined above, we can define the information profit from the ith voxel:

$$\omega_{info}(i) = p_{arr}\left(i\right) * c_{info}, \tag{5}$$

where c_{info} is the unit size of the information profit.

2) Octree Pruning Profit: An important feature of OctoMap is the ability to perform pruning operations, thus saving memory space while ensuring accuracy. Therefore, in this paper, for the memory-constrained nano-UAVs, whether a candidate waypoint can trigger the pruning of the Octree is taken as one of the factors to measure its profit. For the candidate waypoint q, its pruning profit is calculated as:

$$W_{prune}\left(q, L\right) = \sum_{i=1}^{m} \left(p_{arr}(i) * p_{prune}(i) * c_{prune}\right) \tag{6}$$

where $p_{arr}(i)$ is the reachable probability of voxel i, calculated as 4,$p_{prune}(j)$ denotes the triggered pruning probability of voxel j, and c_{prune} denotes the unit pruning profit.

In the trigger analysis of the pruning operation, it can be learned that the octree pruning operation will be triggered and only when the states of all sibling nodes are consistent. Therefore, the prune trigger probability can be expressed as:

$$p_{\text{prune}}(i) = \begin{cases} p_{occ}(i)^{8-n_{occ}}, n_{occ} \neq 0, n_{free} = 0, \\ (1 - p_{occ}(i))^{8-n_{free}}, n_{occ} = 0, n_{free} \neq 0, \\ p_{occ}(i)^{8} + (1 - p_{occ}(i))^{8}, n_{free} = 0, n_{occ} = 0, \\ 0, else \end{cases} \tag{7}$$

where $p_{occ}(i)$ is the occupation probability of voxel i, calculated as 3, n_{occ} is the number of nodes known to be in the occupied state among the sibling nodes of voxel i, n_{free} is the number of nodes known to be free among the sibling nodes of voxel i.

3) Combined Profit Functions and Active Exploration Strategy: Considering the information profit and pruning profit together, the total profit function of the candidate waypoints in direction L is obtained in this paper as:

$$W_{sum}(q, L) = W_{info}(q, L) + W_{prune}(q, L)$$

In order to better consider the importance of each profit at different stages of exploration, the profit function is further optimized in this paper as follows:

$$\widetilde{W}_{sum}(q, L) = \alpha * W_{info}(q, L) + \beta * W_{prune}(q, L)$$

where β is the weight factor of pruning profit, which is equal to the ratio of the current memory pool allocated space size to the total memory pool size, i.e., the memory pool utilization rate. And α is the weight factor of information profit, i.e., α = 1 - β. The memory pool utilization reflects the current memory pool usage. The introduction of dynamic weights not only ensures the efficiency of exploration in the early stage of exploration but also triggers pruning to prevent memory pool overflow when the memory pool utilization increases as much as possible, which guarantees the security of the memory pool.

Count the profit of each detection direction for individual candidate waypoints, and sum them up to compare the total profit of each waypoint, We define the final decision function as:

$$\arg\max_{q \in Q} \left(\sum_{L} (\widetilde{W}_{sum}(q, L))\right)$$

With the next step waypoint being continuously selected based on the above equation, the NAV can complete the exploration of the space and maximize the benefits of exploration.

IV. Design of Edge-Assisted Offloading Framework

In this paper, we introduce two types of NAVs in the heterogeneous system: **Lidar-NAV** and **Edge-NAV**. The Lidar-NAV is responsible for using LiDAR to perceive the surrounding environment, while the Edge-NAV uses the environmental information from multiple Lidar-NAVs to build the map.

A. System overview

Generally, we can expand the capabilities of the NAV by installing different expansion boards. However, due to reasons such as payload, energy efficiency, and communication conflicts, the choice of expansion boards is mutually exclusive. Therefore, this paper installs the lidar-ranging expansion board for the Lidar-NAV and installs the expansion board that can provide ultra-low power edge computing capabilities for Edge-NAV. The Lidar-NAV will continue to send request packets to the Edge-NAV. For the request packets that need a response, the Edge-NAV will return the corresponding response packet.

For the cooperative autonomous mapping in the heterogeneous NAV, we design three kinds of packets shown in Table. I.

TABLE I: Cooperative autonomous packet payload

Message Type	Data Content
Mapping Request	Variable-length structure array
Exploration Request	current coordinate, attitude, and range
Exploration Response	next coordinate

1) Mapping Request Packet: The mapping request from multiple Lidar-NAVs will be sent to the Edge-NAV, and the Edge-NAV will use the multi-source sensor information to construct the TinyOctoMap designed in Section. III-A for autonomous mapping. The payload of the mapping request packet is an array of coordinate pairs, containing the start and end coordinates of the LiDAR ray, and the times of this ray can represent.

2) Exploration Request Packet: In order to achieve active exploration, the Lidar-NAV needs to select its next movement location based on the surrounding environment. The exploration request packet provides the necessary information for the NBV-based active exploration algorithm designed in Section III-B. The exploration request needs a response, as the Lidar-NAV requires a returned next coordinate to move.

3) Exploration Response Packet: After receiving a request packet for exploration, the Edge-NAV calculates the next position to which the Lidar-NAV should move. As a result, the data field of the exploration response packet only contains a single coordinate.

B. Merge Rays and Accelerate Building Map

The time required for updating the OctoMap increases with the depth of the tree. In the context of autonomous mapping, not only the octree nodes corresponding to obstacles need to be updated but also using the Bresenham line algorithm to traverse and update every node that the LiDAR passes through between its starting and ending points. Compared to the frequency of sensor data acquisition, updating the OctoMap always takes more time, and classic methods have a lot of redundancy in updating the map.

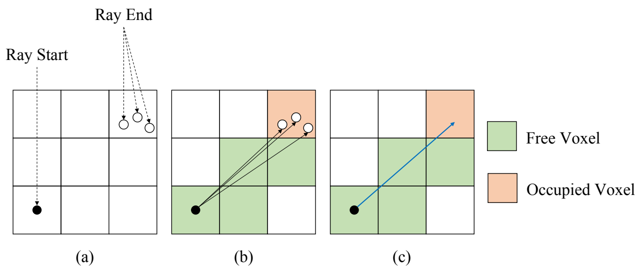


Fig. 5: Redundant computation while updating OctoMap

Fig. 5 illustrates the redundant computation while updating OctoMap. Fig. 5(a) and Fig. 5(b) show that when the NAV is stationary or moves insignificantly, the endpoints of several consecutive LiDAR rays correspond to the same node in the OctoMap, and the nodes traversed by the Bresenham line algorithm are also the same. Using classic methods, the LiDAR rays need to be sent independently, and the OctoMap updating program needs to recursively traverse the same nodes and change the occupancy probability of each node.

Fig. 5(c) shows the proposed improvement strategy, where the Edge-NAV combines multiple mergeable LiDAR rays, and the batch updating of octree nodes is performed by the Edge-NAV based on the number of merges. This method not only reduces the number of mapping requests sent but also accelerates the map updating speed of the Edge-NAV without compromising accuracy. The merging process involves selecting a representative ray and adding a counter to record the number it represents.

Therefore, this paper references the Super Ray [24] and proposes a linear time complexity ray merging algorithm to compare whether two LiDAR rays update the same node in the map. The Super Ray algorithm needs to split the LiDAR ray, not suitable for heterogeneous multi-NAV since it requires transmitting more additional data.

Our algorithm first checks if the starting and ending points of the two rays are the same. If they are not the same, the algorithm returns False. Then, the algorithm calculates the path of each ray using the Bresenham algorithm. If the number of nodes in the paths of the two rays is different, the algorithm returns False. Finally, the algorithm compares each node in the paths of the two rays. If there is any mismatch, the algorithm returns False. If all nodes match, the algorithm updates the count of the representative ray and returns True. The time complexity of the algorithm is $O(N)$, where N is the number of nodes in the paths of the two rays.

C. Adjust Edge-NAV Positions Dynamically

The communication of NAVs is constrained by their weak communication module, and the packet loss rate of wireless communication is affected more significantly by the communication distance. As the position of Lidar-NAV constantly changes during autonomous mapping, the communication quality between Lidar-NAV and Edge-NAV also varies. Unstable communication channels can cause short-term delays or packet loss. Therefore, it is necessary to dynamically adjust the position of the Edge-NAV to reduce the overall packet loss rate and improve the system's communication quality. The goal is to optimize the position of the Edge-NAV to maximize network coverage and communication quality.

The artificial potential field (APF) method is applied in this paper to determine the positions of Edge-NAV. The Lidar-NAV are regarded as repulsive forces to balance the distance between Edge-NAV and each Lidar-NAV. Meanwhile, the Lidar-NAVs with high packet loss rates are used as attractive forces, so that the edge Edge-NAV can provide better services for Lidar-NAVs with poor communication quality. The optimization goal of this paper is to reduce the packet loss rate of the offloading uplink. To calculate the repulsive and attractive forces, the position and packet loss rate of Lidar-NAVs are required. The necessary data can be obtained by using the current position of the NAV and the sequence number in the mapping request packet. This further verifies the rationality of the designed packet payload in this paper.

Let X denote the position of the Edge-NAV at time t, X_i denote the position of the i-th laser Lidar-NAV, and L_i denote the corresponding packet loss rate between the Edge-NAV and the i-th laser rangefinder Lidar-NAV. The corresponding repulsive potential function can be calculated as:

$$U_{\text{rep}}(X)=\begin{cases}\frac{1}{2}K_{\text{rep}}\left(\frac{1}{\rho(X,X_l)}-\frac{1}{\rho_0}\right)^2, \rho(X,X_l)\leq\rho_0\\ 0, \rho(X,X_i)>\rho_0\end{cases} \tag{8}$$

In the equation, K_{rep} is the repulsion field coefficient, which is used to adjust the strength of repulsion. ρ_0 represents the maximum range of the repulsion field, and $\rho(X, X_i)$ represents the distance between the two NAVs. The direction of repulsion is opposite to the direction from X to X_i.

$$F_{rep}(X)=-\nabla\left(U_{rep}(X)\right) \tag{9}$$

Similarly, the calculation of the attraction field function is as follows:

$$U_{att}(X)=\frac{1}{2}K_{att}\left(X-X_l\right)^2 \tag{10}$$

Where K_{att} is the attraction field coefficient, which is used to adjust the strength of attraction. The calculation of the attraction force is as follows:

$$F_{att}(X)=-\nabla\left(U_{att}(\text{X})\right) \tag{11}$$

The resultant force acting on the Edge-NAV is as follows:

$$F(X)=\sum_{i=1}^{N}F_{att}^{l}(X)+F_{\text{rep}}^{l}(X) \tag{12}$$

Update the next position $X^{'}$ of the Edge-NAV:

$$X^{'}=X+F(X) \tag{13}$$

V. Implementation

We implement the proposed cooperative and autonomous mapping system on Crazyflies, an STM32-driven NAV with a tiny STM32 MCU, inertial measurement unit, 2.4GHz communication module, and reserved pins for mounting expansion boards. We install a LiDAR expansion board (Multi-Ranger Deck) with four single-line LiDARs on Lidar-NAV and an edge computing expansion board (AI-Deck) with a GAP8 chip on the Edge-NAV to enhance the NAV's capabilities.

For data acquisition, we use HTC Vive Lighthouse, an optical indoor positioning system, to acquire absolute NAV position information with millimeter-level accuracy. We use the IMU on the NAV to acquire the attitude information of the NAV in real-time and use the LiDAR to collect the distance between the obstacles in the environment and the NAVs.

For data transmission, we use the 2.4GHz module on board the NAV for P2P communication between different NAVs. Inside the Edge-NAV, we use CPX protocol based on UART communication between STM32 and GAP8 chips.

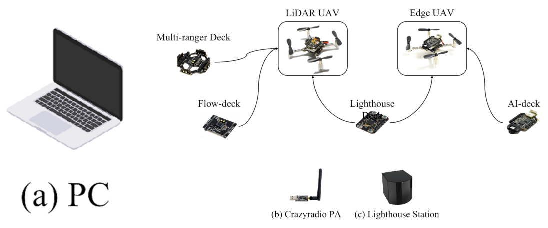


Fig. 6: Experiment devices

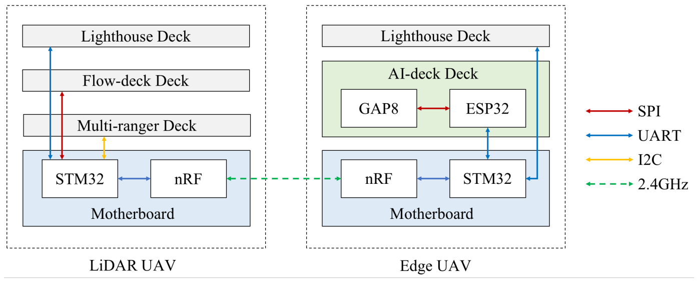


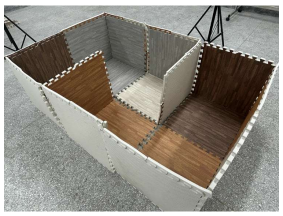
Fig. 7: Communication link

VI. Experiment

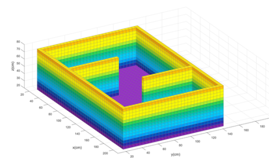
This section presents the simulation experiments and real experimental results of autonomous exploration and map building in unknown narrow environments by NAV equipped with adaptive exploration strategies in single- and multi-NAVs experiments.

A. Single-NAV Experiments

1) Single-NAV Simulation Results: To verify the efficiency of the exploration algorithm described in Section III and to analyze the memory pool usage, this section first designs a set of simulation experiments. The experiments are implemented on a Python platform using a Mac Air computer equipped with an Apple M2 chip and 16GB of memory size. In the simulations, the UAV is defined as a dynamic model with current coordinates p(x,y,z) and a fixed velocity of 0.16m/s. We build an equal-scale global simulation map model with reference to the real experimental environment, which is shown in Fig.8.



(a) Real map

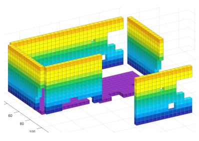


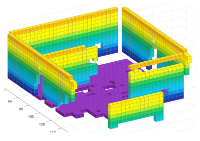
(b) Simulation map model

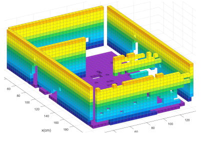
Fig. 8: Experimental map model

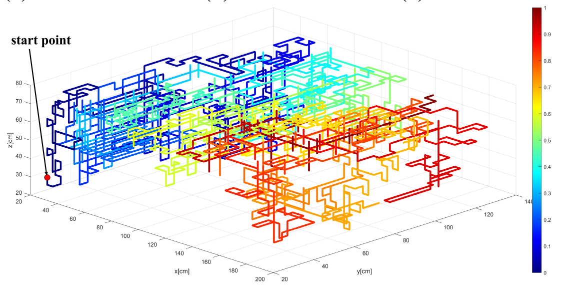
Due to the limited performance and load of the NAV, we use a Lidar model with 6 channels and a range of 3m for distance sensing.

First, we demonstrate the ability of the NAV equipped with the proposed exploration strategy to explore and build maps independently and record the exploration completion and memory pool utilization in real-time. The exploration completion refers to the ratio of the explored area volume to the total area volume, while the memory pool usage refers to the ratio of the number of used nodes to the memory pool capacity. The results of the Single-NAV simulation experiments are as follows:


(a) time = 300s


(b) time = 650s


(c) time = 1000s



(d) flight path

Fig. 9: Single-NAV simulation map building results with 4cm accuracy

In Fig. 9, the effect of UAV map building with 4cm accuracy is shown. The mapping result verify the correctness of the UAV model and Lidar model, as well as to validate the effectiveness of the exploration strategy proposed in this paper

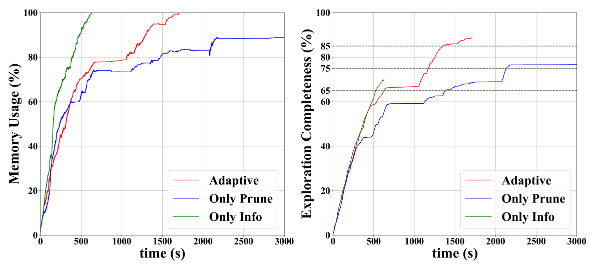


Fig. 10: Memory pool usage and exploration completeness with 2cm accuracy under different strategies

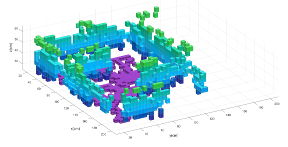
In order to more comprehensively evaluate the effectiveness of the exploration strategies proposed in this paper, We compare the active exploration strategy that only considers information profit, the strategy that only considers pruning profit and the strategy proposed in this paper. Fig. 10 shows the comparison of map exploration completion and memory pool usage for these strategies with 2cm accuracy, where the map exploration completion refers to the ratio of the explored volume of the map to the total volume of the map and the memory pool usage refers to the ratio of the number of used nodes to the memory pool capacity. The results show that the active exploration strategy considering only information profit has high exploration efficiency but the memory pool is depleted prematurely, resulting in the lowest final exploration completeness of only 70.01%. Although the active exploration strategy considering only pruning profit secures the memory pool as much as possible, it severely sacrifices the exploration efficiency. Only 77.67% exploration completeness is achieved after long enough exploration. The adaptive exploration strategy proposed in this paper balances exploration efficiency and memory pool usage and achieves a final detection completion rate of 88.7%. Compared with the classic active exploration strategy that only considers information profit, the final detection rate is improved by 26.70%.

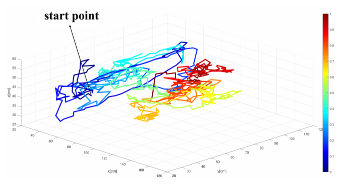
TABLE II: Time to reach 65%, 75%, and 85% exploration completeness

Strategies	65% exploration		75% exploration		85% exploration	
	time(s)	%	time(s)	%	time(s)	%
Adaptive	633.5	100	1174	100	1354	100
Only Info	526.5	83.11	∞	∞	∞	∞
Only Prune	1382	218.19	2135	181.82	∞	∞

In a comprehensive analysis of exploration completion, only the strategy proposed in this paper achieves 85% exploration completion; the time required to achieve 75% exploration completion is 54.17% less than that of the exploration strategy considering only pruning profit; the time required to achieve 65% exploration completion is only 20.32% more than that of the classic active exploration strategy considering only information profit. Taken together, the strategy proposed in this paper shows significant advantages for the task of active exploration and map building in a small environment with memory-constrained NAVs.

2) Single-NAV real experiments: In order to verify the effectiveness and robustness of our proposed active exploration strategy in a real environment, we conducted real flight experiments based on the Crazyflie platform. In order to prevent the NAV from crashing due to the strong influence of ground airflow, we set the bottom height to 25cm. The results of the singal-NAV real experiment are shown in Fig. 11.


(a) time = 175s



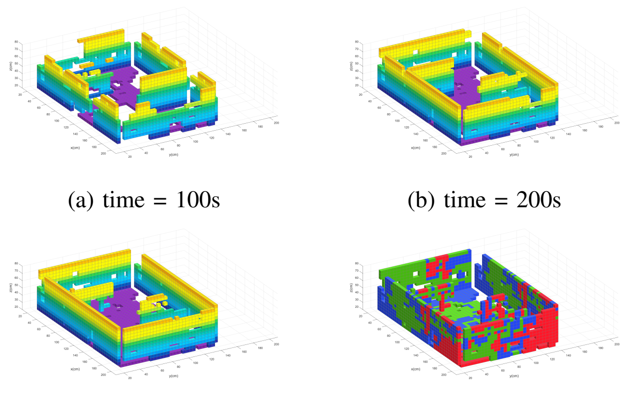
(b) flight path

Fig. 11: Single-NAV real map building results

Fig. 11a depicts the map created by a single UAV after 175 seconds of flight, revealing that the map's outline has essentially completed the scanning process, with a map exploration completion rate of 82.28% and memory pool usage of 59.53% at this point. Fig. 11b shows the real flight path of the NAV. Through real-world experiments on the Crazyflie platform, we confirm that our proposed active exploration strategy can cope with a variety of errors, such as localization errors and ranging errors, in a real environment and still complete the map construction task effectively. This further demonstrates the robustness and practicality of our proposed active exploration strategy.

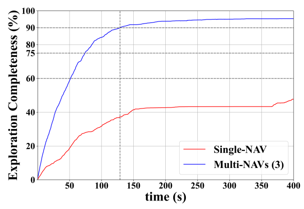
B. Multi-NAVs Experiments

1) Multi-NAVs Simulation Results: In order to further verify the effectiveness of the adaptive exploration strategy proposed in this paper in multi-machine exploration and the synergy of multi-machine cooperative map building, we conducted multi-NAVs simulation experiments. Based on the map building effect shown in Fig. 12, we verify the effectiveness of the adaptive exploration strategy proposed in this paper in a multi-NAVs environment. The voxels in different colors in Fig.12d indicate that different UAVs are responsible for the update, further demonstrating the collaborative map building effect of multiple UAVs Fig. 13a compares the exploration completion for single NAV and multiple NAVs at the same time. The data shows that when the exploration completion of the multi-NAVs reaches 90%, the completion of the single-NAV is only 37.26%, indicating that the exploration efficiency of the multi-NAVs is 2.42 times higher than that of the single-NAV.

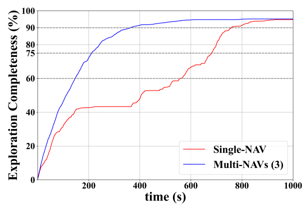

(a) time = 100s (b) time = 200s

(c) time = 400s (d) Tricolor map

Fig. 12: Multi-NAVs map building results



(a) Exploration completeness in the same time

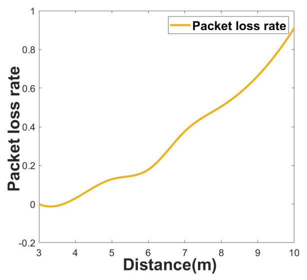


(b) Exploration completeness in the same number of explorations

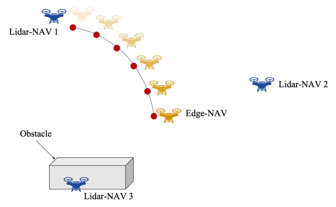
Fig. 13: Single NAV and multi-NAVs exploration completion

Fig. 13b compares the exploration completion of single NAV and multiple NAVs for the same number of explorations. The data shows that when the exploration completion reached 90%, the cumulative time spent by the multi-NAVs was 364s, while the time spent by the single-NAV was 757.75s, indicating that the exploration efficiency of the multi-NAVs was 2.0817 times higher than that of the single-NAV.

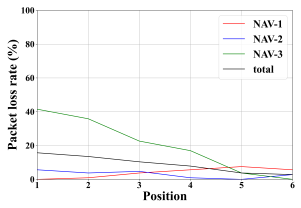
In summary, multi-NAVs exploration has obvious advantages in accelerating map building, which can greatly improve exploration efficiency and shorten exploration time.



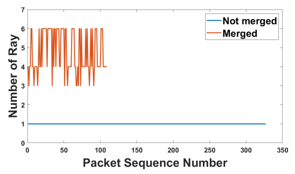
(a) Packet loss rate varies with distance



(b) Dynamic position



(c) Packet loss rate varies with position



(d) MergeNum changes with the sequence number

Fig. 14: Communication optimization strategy verification

2) Multi-NAVs real experiments: We conduct experiments to verify the effectiveness of the proposed strategies in Section IV. Fig. 14b shows the adaptive position based on the APF method. Fig 14c shows the packet loss in this process. Fig 14d shows the effectiveness of the strategy to merge similar rays, we reduce many packets that need to be sent and speed up the octree updating.

The results of the Multi-NAVs real experiment are shown in Fig. 15 The blank voxels in Fig.15a indicate the vacant area. The results demonstrate the higher efficiency of multi-machine exploration and verify the stability of the proposed unloading strategy and heterogeneous system in this paper.

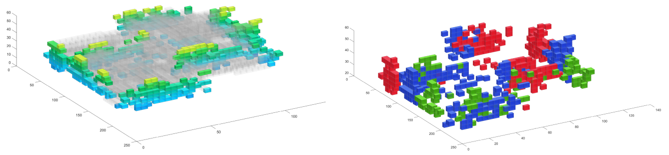

(a) time = 30s (b) Tricolor map

Fig. 15: Multi-NAVs real map building results

VII. Conclusion

In this paper, we propose a cooperative NAV system using heterogeneous NAVs for autonomous map building in unknown and confined environments. The system uses the TinyOctoMap model and the adaptive active exploration strategy proposed in this paper, which improves memory usage efficiency while ensuring exploration efficiency. At the same time, the system designs an efficient computing offloading strategy to offload the computing of resource-constrained NAV within the cluster. The proposed system is evaluated through a large number of simulations and real-world experiments, which proves the effectiveness and robustness of the system. The system has been proven to realize NAV heterogeneous collaboration to explore complex and narrow environments and build maps.

Acknowledgment

This work was supported by National Natural Science Foundation of China Grants No.62072101, No.62232004, and No.61972086, the Fundamental Research Funds for the Central Universities, Jiangsu Provincial Key Laboratory of Network and Information Security Grant BM2003201, Key Laboratory of Computer Network and Information Integration of the Ministry of Education of China Grant 93K-9.

References

[1] Y. Sun, Z. Huang, H. Zhang, and X. Liang, "3D Reconstruction of Multiple Objects by mmWave Radar on UAV," in *2022 IEEE 19th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, pp. 491–495, 2022.

[2] X. Chen, C. Ruiz, S. Zeng, L. Gao, A. Purohit, S. Carpin, and P. Zhang, "H-DrunkWalk: Collaborative and Adaptive Navigation for Heterogeneous MAV Swarm," *ACM Trans. Sen. Netw.*, vol. 16, no. 2, 2020.

[3] Q. Chen, H. Zhu, L. Yang, X. Chen, S. Pollin, and E. Vinogradov, "Edge Computing Assisted Autonomous Flight for UAV: Synergies between Vision and Communications," *IEEE Communications Magazine*, vol. 59, no. 1, pp. 28–33, 2021.

[4] O. Esrafilian, R. Gangula, and D. Gesbert, "3D-Map Assisted UAV Trajectory Design Under Cellular Connectivity Constraints," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, 2020.

[5] T. Suzuki, D. Inoue, and Y. Amano, "Robust UAV Position and Attitude Estimation using Multiple GNSS Receivers for Laser-based 3D Mapping," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4402–4408, 2019.

[6] J. Li, G. Zhang, Q. Shan, and W. Zhang, "A Novel Cooperative Design for USV-UAV Systems: 3D Mapping Guidance and Adaptive Fuzzy Control," *IEEE Transactions on Control of Network Systems*, pp. 1–11, 2022.

[7] S. Zhang and R. Zhang, "Radio Map-Based 3D Path Planning for Cellular-Connected UAV," *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1975–1989, 2021.

[8] D. Gesbert, O. Esrafilian, J. Chen, R. Gangula, and U. Mitra, "UAV-aided RF Mapping for Sensing and Connectivity in Wireless Networks," *IEEE Wireless Communications*, pp. 1–7, 2022.

[9] E. Akturk and A. O. Altunel, "Accuracy assessment of a low-cost UAV derived digital elevation model (DEM) in a highly broken and vegetated terrain," *Measurement*, vol. 136, pp. 382–386, 2019.

[10] Y. Rong, R. Gutierrez, K. V. Mishra, and D. W. Bliss, "Noncontact Vital Sign Detection With UAV-Borne Radars: An Overview of Recent Advances," *IEEE Vehicular Technology Magazine*, vol. 16, no. 3, pp. 118–128, 2021.

[11] S. Kemp and J. Rogers, "UAV-UGV Teaming for Rapid Radiological Mapping, year=2021," in *2021 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 92–97.

[12] U. Emmanuel and B. Yekini, "Review of Agricultural Unmanned Aerial Vehicles (UAV) Obstacle Avoidance System," in *2022 IEEE Nigeria 4th International Conference on Disruptive Technologies for Sustainable Development (NIGERCON)*, pp. 1–4, 2022.

[13] M. Collins and N. Michael, "Efficient Planning for High-Speed MAV Flight in Unknown Environments Using Online Sparse Topological Graphs," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11450–11456, 2020.

[14] A. Patel, C. Kanellakis, and G. Nikolakopoulos, "Traversable Frontiers Based Autonomous Exploration Strategy for Deploying MAVs in Subterranean Environments, year=2022," in *2022 Eighth Indian Control Conference (ICC)*, pp. 260–265.

[15] A. Dai, S. Papatheodorou, N. Funk, D. Tzoumanikas, and S. Leutenegger, "Fast Frontier-based Information-driven Autonomous Exploration with an MAV," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9570–9576, 2020.

[16] V. M. Respall, D. Devitt, R. Fedorenko, and A. Klimchik, "Fast Sampling-based Next-Best-View Exploration Algorithm for a MAV," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 89–95, 2021.

[17] 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.

[18] N. Funk, J. Tarrio, S. Papatheodorou, M. Popović, P. F. Alcantarilla, and S. Leutenegger, "Multi-Resolution 3D Mapping With Explicit Free Space Representation for Fast and Accurate Mobile Robot Motion Planning," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3553–3560, 2021.

[19] C.-C. Wong, C.-M. Vong, X. Jiang, and Y. Zhou, "Feature-Based Direct Tracking and Mapping for Real-Time Noise-Robust Outdoor 3D Reconstruction Using Quadcopters," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 20489–20505, 2022.

[20] O. Doukhi and D. J. Lee, "Deep Reinforcement Learning for Autonomous Map-Less Navigation of a Flying Robot," *IEEE Access*, vol. 10, pp. 82964–82976, 2022.

[21] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous robots*.

[22] C. Deng, X. Luo, and Y. Zhong, "Improved closed-loop detection and Octomap algorithm based on RGB-D SLAM," in *2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, pp. 73–76, 2020.

[23] H. Min, K. M. Han, and Y. J. Kim, "Accelerating Probabilistic Volumetric Mapping using Ray-Tracing Graphics Hardware," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5440–5445, 2021.

[24] Y. Kwon, D. Kim, I. An, and S.-e. Yoon, "Super rays and culling region for real-time updates on grid-based occupancy maps," *IEEE Transactions on Robotics*, vol. 35, no. 2, pp. 482–497, 2019.