Received <day> <Month>, <year>; | Revised <day> <Month>, <year>; | Accepted <day> <Month>, <year>

DOI: xxx/xxxx

RESEARCH ARTICLE

An autonomous UAV system based on adaptive LiDAR Inertial Odometry for practical exploration in complex environments

Boseong Kim | Maulana Bisyir Azhari | Jaeyong Park | David Hyunchul Shim

[1]Department of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea

Correspondence
Boseong Kim, KAIST. Email: brian.kim@kaist.ac.kr

Funding Information
This research was supported by the Institute of Civil Military Technology Cooperation funded by the Defense Acquisition Program Administration and Ministry of Trade, Industry and Energy of Korean government, Grant/Award Number: UM22206RD2

Abstract

Unmanned aerial vehicles (UAVs) offer many advantages over ground vehicles, including quadruped robots, based on high maneuverability when performing exploration in complex and unknown environments. However, due to their limited computational capability, UAVs require light-weight but accurate state estimation algorithms for reliable exploration. In this paper, we propose an segmented map based exploration system based on LiDAR-based state estimation for UAVs. The proposed system includes capabilities such as exploration, obstacle avoidance, and object detection with localization using 3D dense maps generated by tightly coupled LiDAR Inertial Odometry (LIO). Our proposed system is a hybrid system that can switch between guided and exploration modes, making it practical for search and rescue missions in disaster scenarios. The proposed LIO algorithm adapts to its surroundings, allowing for fast and accurate state estimation in complex environments. The proposed exploration algorithm is designed to cover specific regions in the 3D dense map generated by proposed LIO, with the UAV determining if map points are included within the coverage area. We tested the proposed system in both simulation and real-world environments and validated that proposed system outperforms state-of-the-art algorithms in various aspects such as localization accuracy and exploration efficiency in complex environments.

KEYWORDS:
Collision avoidance, exploration, LiDAR inertial odometry, perception, UAV

1 | INTRODUCTION

Exploration using uncrewed platforms is a critical area of research in field robotics. This practical application has a great potential to significantly reduce casualties and property damage by acquiring information on various risks that exist in unknown environments before human intervention. In particular, exploration includes missions such as rescue, search, and geometric information acquisition, which are essential elements in disaster environments or unknown spaces. The emergence of interest in exploration in the field robotics community has been largely influenced by the DARPA Subterranean Challenge[1], where various unmanned platforms, including UAVs, UGVs, and quadruped robots, were utilized for navigation in unknown spaces in the underground environment with the objective of finding specific objects and reporting their locations. The competition has demonstrated that unmanned platforms can be effectively used for exploration through the application of reliable state estimation using various sensors such as LiDAR, vision, and IMU in GPS-denied environments.

In the field of robotics, the utilization of UAVs has demonstrated significant capabilities due to their high maneuverability compared to other platforms such as UGVs and quadruped robots. UAVs can be operated reliably in various environments, including stairs, cliffs. However, designing unmanned systems using UAVs presents several challenges. The platform has significant weight and size constraints, leading to limited computational power, sensor configuration, and operating time. Therefore, to operate UAVs efficiently within these constraints, it is crucial to design reliable algorithms with low computational cost.

[1]https://www.subtchallenge.com/

Safe and reliable exploration requires basic navigation components, such as robust and stable state estimation algorithms. Additionally, Given that UAVs are operated in 3D space, it is essential to design light-weight 3D-based algorithms (e.g., localization, path planning). The Simultaneous Localization and Mapping (SLAM) algorithm has developed remarkably in the last decade, relying on various sensors such as LiDAR, vision, and IMU, particularly in GPS-denied environments. Typically, SLAM consists of a front-end module that estimates LiDAR odometry and a back-end module that detects loops and optimizes the states. However, in disaster environments where the UAV might not be operated in a loop area, precise and accurate LiDAR odometry estimation becomes one of the most important factors for exploration missions. To achieve accurate state estimation without a loop closure algorithm, we propose a kNN-based sub-map generation to optimize the state of UAVs. Additionally, the proposed LIO algorithm adapts to the surroundings, which enables reliable state estimation in various complex environments, such as multi-floor and large cave environments.

Another critical component for exploration is path planning and collision avoidance. It is usually divided into grid search-based and motion primitive-based approaches, which require a balance between computational cost and path planning performance. For a light-weight and reliable path planning, we propose a method that represents a 3D environment as several 2D planes and selects the best path among the results of the A* algorithm for each grid map. This method is combined with a motion primitive-based path planning algorithm to enable reliable obstacle avoidance for fast exploration of UAVs within limited operation time. The proposed method can significantly improve the exploration capabilities of UAVs, reducing the risk of collisions and improving the efficiency of exploration missions.

The exploration algorithm, which serves as the primary mission planner in the system proposed in this paper, is designed to ensure that the UAV includes map points generated only within a specific coverage. Existing exploration algorithms have primarily developed based on either motion primitive-based or Rapidly-exploring Random Tree (RRT)-based path planning methods in unknown areas. However, these methods are limited in terms of computational cost and exploration efficiency and are challenging to use within a specific area. The proposed method in this paper aims for efficient and fast exploration only within a specific area with a limited time. Also, the UAV can be operated in a hybrid mode of guided mode and exploration mode, allowing it to explore only within the desired area by the operator, resulting in the practicality of the proposed system. Additionally, during exploration, object detection and localization are performed for specific objects, allowing the operator to verify the location of specific objects in the map generated by the proposed LIO.

In this paper, we propose a fast and accurate LIO based autonomous exploration system that can be practically implemented on a UAV in a real-world environment, as shown in Fig. 1 . To the best of the authors' knowledge, this paper represents a pioneering effort in validating the realistic operation of UAV within disaster scenarios. It involves launching the UAV from the outside, navigating it inside buildings, performing exploration, and escaping to specific locations. Additionally, it includes conducting object detection and localization to provide the operator not only with a precise 3D map but also with the positions and types of objects. Our proposed system comprises several subsystems including exploration, LIO, collision avoidance, and object detection and localization. Through a series of flight tests in both simulation and real-world environments, we validated that the proposed system outperforms existing LIO algorithms in terms of computational cost and accuracy, and is more efficient and faster than existing exploration algorithm.

The main contributions of this paper are highlighted below:

- We propose a tightly-coupled LIO that can adapt to the surroundings. The proposed LIO analyzes the geometric conditions of the surroundings by analyzing the current scan data and setting the registration parameters accordingly. Moreover, we optimize the UAV's state through kNN-based sub-map generation and matching, enabling robust and reliable real-time state estimation even in challenging environments.
- We propose a fast and efficient exploration algorithm based on Euclidean distance clustering. The proposed exploration algorithm transforms the 3D map generated from proposed LIO into several segmented regions. The priority of the regions to be explored will be determined by the exit-aided exploration score, improving the practicality and efficiency of the algorithm.
- We propose a computationally-efficient object detection and localization pipeline using EfficientDet-Lite, which runs on the Google Coral Edge TPU, along with Euclidean distance clustering to estimate their 3D position.
- We implemented the proposed system on a real drone, conducted exploration missions in various environments, and reported the locations of the target objects detected in the 3D map generated by the proposed LIO. Through this, we demonstrated the practicality of the proposed system.

The structure of this paper is as follows. Section 2 provides an introduction to various previous studies related to the proposed system. Section 3 presents an overview of the proposed system. In Section 4, each sub-system that makes up the proposed system is elaborated upon. Section 5 analyzes and verifies the performance of the proposed system in both simulation and real-world environments. Finally, in Section 6, the study concludes.

2 | RELATED WORK

2.1 | Localization

Localization is the most critical component for a robot to operate in an unknown environment. LiDAR odometry, which involves using a LiDAR sensor to estimate a robot's motion over time, is a well-established research area in the field of robotics and computer vision. Several methods have been proposed in the literature to accurately and efficiently

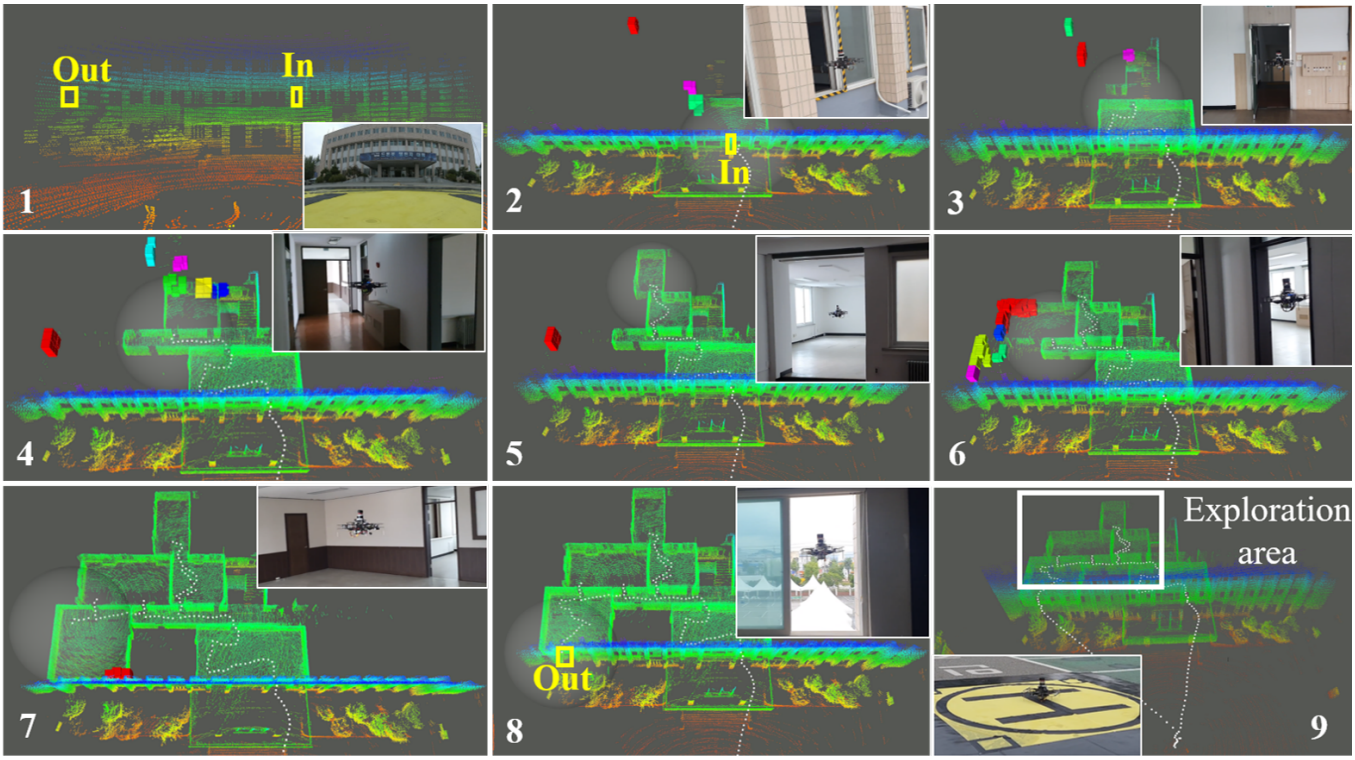


FIGURE 1 Snapshots of the proposed fast and accurate LIO-based exploration system in a time sequence. 1) At the starting point, the entrance and exit are set to use the guided mode, and the area to be explored is selected. 2) After flying in guided mode to the entrance, the UAV switches to exploration mode. 3-7) The UAV explores by including map points in a specific coverage, as shown by the transparent gray sphere. The colorful voxels represent the area to be explored, and their colors indicate different regions based on Euclidean distance-based clustering of map points. 8)After exploring within the pre-defined area, the UAV switches back to guided mode towards the exit, and then performs auto-homing to the starting point. The white dots represent the UAV trajectory.

determine the robot's pose. One popular approach for LiDAR odometry is the use of feature-based methods, such as LOAM [49], LeGO-LOAM [31], and Fast-LOAM [42]. These methods extract features such as edges or corners from LiDAR data and track them over time to estimate the translation between LiDAR scans. Another recent approach is $\mathcal{PLC}$-LiSLAM [52], which utilizes geometric information about planes, lines, and cylinders in the point cloud features and checks the cost in the back-end to minimize the errors introduced by data association. By coupling LiDAR with an inertial measurement unit (IMU) using the pre-integration method [11], the estimation accuracy and speed are improved, as demonstrated by methods like LIO-SAM [32]. Fast-LIO [45] and Fast-LIO2 [44] combine LiDAR points with IMU through an iterated extended Kalman filter, enabling robust estimation in a fast-moving vehicle.

Several matching algorithms have been developed to estimate the translation and rotation between two consecutive LiDAR scans. These methods include Iterative Closest Point (ICP) [1, 6], Generalized ICP (GICP) [30], Voxelized GICP (VGICP) [20], and Normal Distribution Transform (NDT) [2]. HDL-graph-SLAM [19] can utilize NDT, ICP and GICP for high definition mapping in real-time. LiTAMIN [47] employs a stabilized ICP with a normalized Frobenius norm cost function. Loop-closure algorithms are also utilized in many SLAM algorithms to refine odometry drift and global map inaccuracies when a robot revisits a place. Specialized loop-closure detection methods, such as Scan-Context [17, 16], perform well in urban environments.

However, conventional LIO approaches suffer from a lack of state compensation at the global scale, without loop closure, which can lead to inaccuracies in long-term state estimation. Furthermore, fixed keyframe generation interval can fail to align common geometrical features in complex and narrow environments such as multi-floor and subterranean environments, limiting state estimation performance. Our method addresses these issues by adaptively generating keyframes and updating corresponding matching parameters through an analysis of the surrounding environment.

This paper is an extended version of our conference paper [15]. The contribution is to enable fast and accurate state estimation in various environments using scan matching parameters that adapt to the surroundings. Furthermore, this journal version integrates the method with

exploration, path planning, and object detection & localization algorithms, enabling UAVs to perform efficient and practical exploration for missions such as search and rescue in disaster environments.

2.2 | Path Planning & Exploration Algorithm

Aerial exploration has garnered significant attention in recent years, and a multitude of researchers have focused on addressing this challenging problem. Initially, frontier-based exploration was proposed in [46], which involves identifying regions at the boundary between explored and unexplored areas. Subsequently, Rapid-exploration [8] extended this approach to aerial vehicles by leveraging the velocity of Unmanned Aerial Vehicles (UAVs) to choose the next frontier to visit. Researchers have made further advancements in frontier-based exploration, with FUEL [51] introducing a fast and agile replanning strategy to maximize UAV velocity. In addition, several studies such as [5, 4, 51] have integrated path planning and exploration using traveling-salesman-problem, leading to agility and robustness in large and complex areas.

Sampling-based methods have been proposed as another solution for exploration. These methods utilize Rapidly-exploring Random Tree (RRT) or Rapidly-exploring Random Graph (RRG) [14] to search for traversable paths in the environment and select a branch as the exploration path. The "next-best-view" planner (NBVP) proposed by [3] employs RRT and selects the best branch based on the volume of the unexplored space in the corresponding branch. In [9], RRG is extended to find paths that optimize the exploration gain within a local area while incrementally constructing a global graph to anticipate dead-end areas and perform return-to-home using the constructed graph. Fast and agile sampling-based exploration has also been proposed by [10], which constructs RRT together with motion primitives resulting in a smoother trajectory for faster exploration.

However, the methods mentioned earlier assume that the robot starts exploration from the center of the area to be explored, which is not applicable in some scenarios. For example, in a search and rescue mission in a disaster-struck building, the UAV will start from outside the building and enter through a specific entry point, provided by a human operator. Furthermore, more specific information such as the area to be explored and the exit route out of the building may be available. Our proposed method integrates this information into the exploration algorithm, resulting in a hybrid mode of guided mode and exploration mode. We utilize the exit waypoint in the proposed exploration score, and the motion primitive planning enables faster exploration while ensuring safety in a constrained area inside the building.

2.3 | Onboard Object Detection & Localization

Onboard object detection and localization of detected objects are essential components of an exploration mission, but it can be challenging due to limited computational power, occlusion, and varying illumination. Recently, deep learning-based object detection methods have gained popularity due to their exceptional performance and robustness to environmental changes compared to classical methods. The state-of-the-art object detectors on COCO dataset [22] are currently based on ResNext, R-CNN, Transformer, YOLO, and FCOS [18, 26, 35, 36, 54, 23, 41, 39, 40, 53]. Although real-time object detectors are mainly based on YOLO [39, 41, 40] and FCOS [35, 36], these methods usually require powerful GPUs such as NVIDIA V100 or NVIDIA A100 that are not available for most micro aerial vehicles. To address this limitation, various studies have developed deep learning-based object detectors for edge or onboard computing. For instance, light-weight CNN backbones like MobileNet [13, 29, 12], ShuffleNet [50, 24], and EfficientNet [33] are designed to support single-CPU object detection [7, 34].

After detecting the target objects, the exploration mission typically necessitates the robot to localize the target in 3D space. However, simply projecting the detection bounding boxes onto the camera and 3D cloud map is not sufficient since the number of objects in the local area must also be taken into account. Therefore, some form of detection association or clustering is required to determine the actual number of objects in the surrounding area. For example, CTU-CRAS-NORLAB team [25] employed an artifact localization filter based on a Kalman filter developed in [38], to reduce false-positives and ensure spatio-temporal consistency of the localized object. In a similar approach, Team Cerberus [37] used a binary Bayes filter that recursively updates after the robot ray-casts all of the pixels in the detection bounding box to the volumetric map.

In this paper, we present a simpler yet effective approach to cluster and filter the detected objects using euclidean distance clustering in the 3D space.

3 | SYSTEM OVERVIEW

An overview of the proposed system is presented in Fig. 2 . The system comprises several sub-systems including localization, object detection & localization, exploration, and 3D path planning. The first sub-system, localization, is crucial and we propose a surrounding-adaptive LIO, which adaptively defines registration parameters used for the scan matching algorithm based on the surrounding geometric volume. The inputs to this module are LiDAR scans and IMU data, while the outputs are UAV states, 3D map points, and keyframes related to the surrounding. The outputs from the localization module are fed into the exploration module. The proposed exploration strategy is to cover the pre-defined area set by operator. In this step, several segmented areas to be explored are generated by Euclidean distance clustering of the 3D map, and the priority is determined based on the exit-aided objective function. As we mentioned before, the proposed system is a hybrid mode of guided and exploration, where in the guided mode, the waypoint set by the operator becomes a direct output, and the exploration area is pre-defined by the operator.

The segmented area selected in the exploration module or the waypoint set by the operator is passed to the 3D path planning module, along with the outputs of the localization module. The proposed

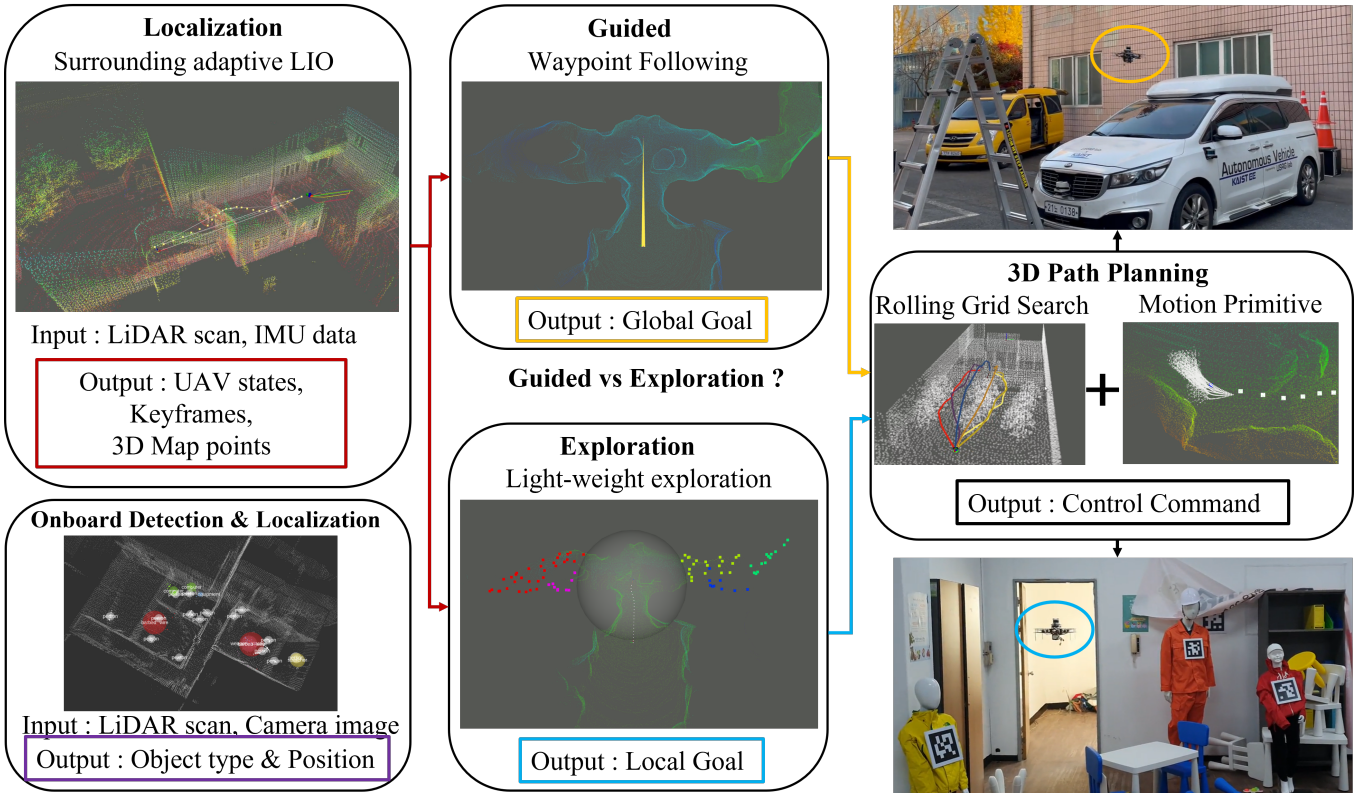


FIGURE 2 Sub-system configuration of the proposed system. The 3D path planning module changes its goal point depending on the guided or exploration mode. The onboard detection and localization module is executed only in the exploration mode. The arrows' color indicates the flow of output data from each module.

3D path planning module combines grid search-based and motion primitive-based path planning. For real-time computation of grid search-based path planning in a 3D complex environment, we represent the 3D environment as 2D grids rotated in the roll direction. For the motion primitives, we generate path candidates in advance and choose the group with the minimum cost through collision check with 3D map points from the proposed LIO. Finally, the output of the 3D path planning module is a control command, which is passed to the flight controller. The object detection & localization module is an optional module that is executed only in the exploration mode and saves the information of detected objects such as type and position. More details of each sub-system are described in Section 4.

4 | METHODS

In this section, we present detailed information about each sub-system of the proposed system, including localization, exploration, path planning, and object detection & localization. It should be noted that the main focus of the proposed system is to ensure reliable and practical exploration in various environments by utilizing fast and accurate localization system for UAV.

4.1 | Localization

3D LiDAR-based localization is a critical component for unmanned vehicles navigating in a GPS-denied environment. LIO has the advantage of providing accurate state estimation results and generating maps that can be implemented in various environments, making it popular in unmanned platforms. However, LIO typically demands high computational resources, necessitating a powerful computer. UAVs, which are limited by size and weight, require reliable state estimation results in complex 3D environments with limited computational power. To address this, we propose a low-cost, high-accuracy adaptive LIO algorithm, as illustrated in Fig. 3 . Our proposed LIO employs down-sampled scans for low computational cost, and its down-sampling and registration parameters are defined using the spatial volume based on the current scan.

4.1.1 | Spatial volume based parameter adjustment

The proposed adaptive LIO algorithm defines all registration parameters based on the spatial volume identified from the voxelization of the current scan, using a pre-defined parameter. To achieve reliable state estimation in various environments, the proposed method uses

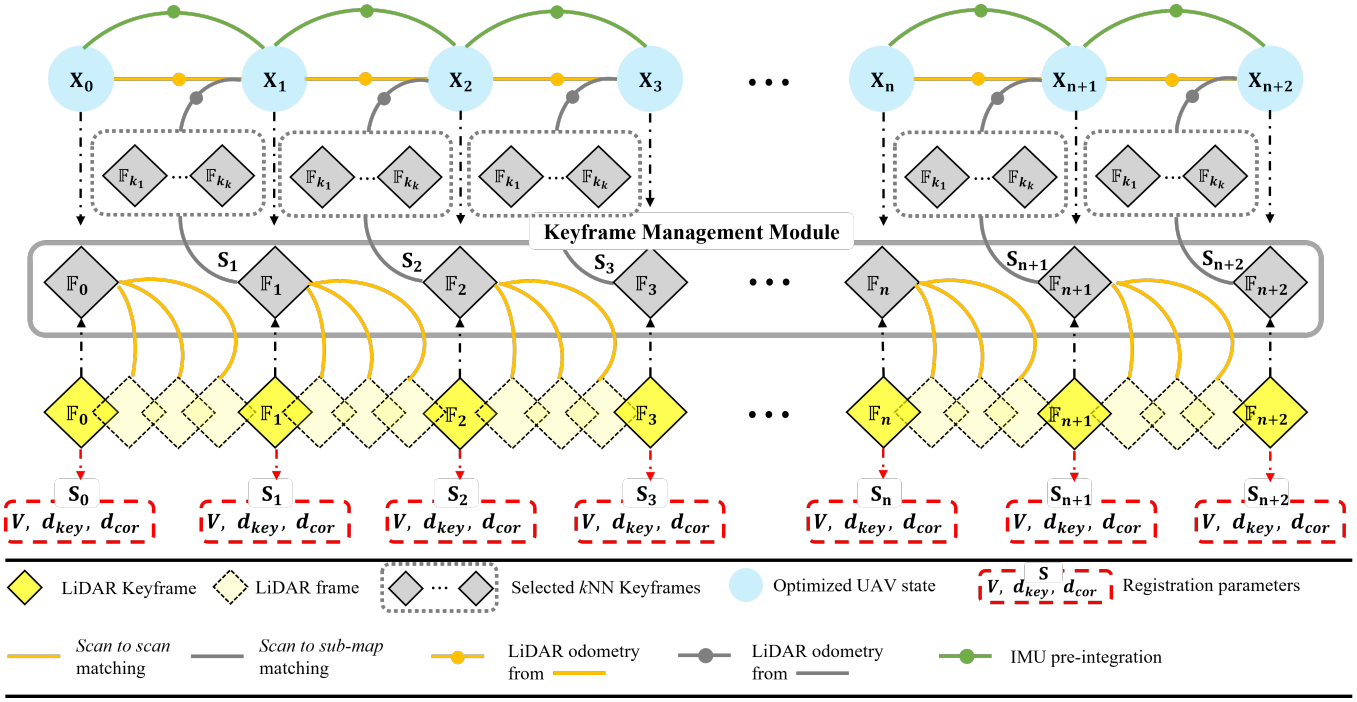


FIGURE 3 The system structure of the proposed surrounding-adaptive LIO. The system performs two types of scan matching registration: *scan to scan* matching to estimate the relative transformation between two consecutive LiDAR frames, and *scan to sub-map* matching to optimize the UAV state for each keyframe. The sub-map used for *scan to sub-map* matching is generated using the k-Nearest Neighbors (kNN) algorithm, which takes the keyframes saved in the keyframe management module and the current UAV state as inputs.

spatial volume that is not utilized by conventional SLAM or LIO algorithms that rely on fixed parameters. The performance of scan matching registration can be divided into two categories, namely computation speed and accuracy. In terms of computational speed, voxelization of the current scan is essential for real-time processing within the limited computational capacity of the onboard computer. While several existing LIO algorithms perform feature extraction to reduce computation cost, this has a significant impact on accuracy in environments with relatively scarce geometric features, such as corridors, narrow areas, and stairs. Therefore, the proposed LIO algorithm uses down-sampled points through voxelization for registration. However, fixed voxelization parameters cannot cover various environments, and therefore the proposed LIO analyzes the geometric volume through the current scan to define the voxelization parameter V proportionally. In terms of accuracy, to optimize the UAV state in the global scale, the proposed LIO uses *scan to sub-map* matching registration when the new keyframe is generated.

The proposed surrounding-adaptive LIO algorithm takes into account various factors affecting accuracy, including keyframe generation interval d_{key} and maximum correspondence distance d_{cor}. Existing LIO algorithms use fixed values for d_{key} and d_{cor}, which can result in reduced accuracy or high computation cost. To address this issue, the proposed LIO analyzes the surrounding spatial volume and adjusts the values of d_{key} and d_{cor} accordingly. Specifically, a large d_{key} is used in a large area

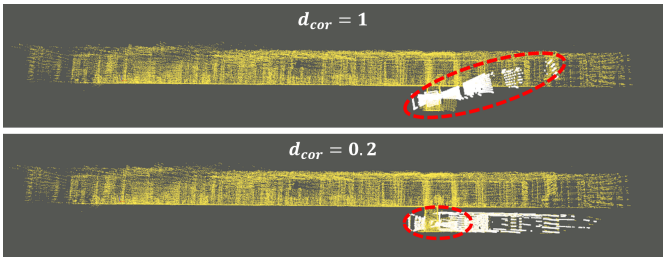


FIGURE 4 *Scan to sub-map* matching performance according to d_{cor} in a multi-floor environment. Yellow scan, white scan, and red dotted circle represent sub-map, aligned scan, and aligned areas, respectively. Because a large d_{cor} tries to align the current scan to the sub-map as accurate as possible, reliable state estimation is difficult in narrow environments such as stair area.

to reduce computation cost, while a small d_{key} is used in a small area to increase accuracy. Similarly, a large d_{cor} is used in a large area to align the current scan with the sub-map as accurately as possible, while a small d_{cor} is used in a narrow area to align the current scan with the limited area in the sub-map. The results of *scan to sub-map* matching according to d_{cor} are shown in Fig. 4 . To determine the surrounding spatial volume, the proposed algorithm uses voxelization of the current scan with a pre-defined parameter $V_{\text{env}} = 10\text{m}$. The voxelization parameter V is

Algorithm 1 Spatial volume based parameter adjustment

Input: Current LiDAR scan $\hat{\mathbf{Z}}_t$, voxel size $V_{\text{env}} = 10$m, maximum voxelization size $V_{\text{scan}}^{\max}$, maximum keyframe generation distance $d_{\text{key}}^{\max}$, maximum correspondence distance $d_{\text{cor}}^{\max}$, exponential decay value α, β, and γ.

Output: Surrounding information $\mathbf{S}$=$\{V, d_{\text{key}}, d_{\text{cor}}\}$ as shown in Fig. 3

$'\hat{\mathbf{Z}}_t \leftarrow$ Outlier filtering($\hat{\mathbf{Z}}_t$) ▷ Initialization

$'\hat{\mathbf{Z}}_t^{V_{\text{env}}} \leftarrow$ Voxelization($'\hat{\mathbf{Z}}_t$) with V_{env} ▷ Surrounding voxelization

$\mathbf{N}_t^{V_{\text{env}}} \leftarrow$ Number of points($'\hat{\mathbf{Z}}_t^{V_{\text{env}}}$) ▷ Spatial volume

if $\mathbf{N}_t^{V_{\text{env}}} \geq 100$ **then** ▷ Tier 1

  $V = V_{\text{scan}}^{\max}$, $d_{\text{key}} = d_{\text{key}}^{\max}$ and $d_{\text{cor}} = d_{\text{cor}}^{\max}$

else if $\mathbf{N}_t^{V_{\text{env}}} \geq 60$ **then** ▷ Tier 2

  $V = \alpha V_{\text{scan}}^{\max}$, $d_{\text{key}} = \beta d_{\text{key}}^{\max}$ and $d_{\text{cor}} = \gamma d_{\text{cor}}^{\max}$

else if $\mathbf{N}_t^{V_{\text{env}}} \geq 35$ **then** ▷ Tier 3

  $V = \alpha^2 V_{\text{scan}}^{\max}$, $d_{\text{key}} = \beta^2 d_{\text{key}}^{\max}$ and $d_{\text{cor}} = \gamma^2 d_{\text{cor}}^{\max}$

else if $\mathbf{N}_t^{V_{\text{env}}} \geq 20$ **then** ▷ Tier 4

  $V = \alpha^3 V_{\text{scan}}^{\max}$, $d_{\text{key}} = \beta^3 d_{\text{key}}^{\max}$ and $d_{\text{cor}} = \gamma^3 d_{\text{cor}}^{\max}$

else ▷ Tier 5

  $V = \alpha^4 V_{\text{scan}}^{\max}$, $d_{\text{key}} = \beta^4 d_{\text{key}}^{\max}$ and $d_{\text{cor}} = \gamma^4 d_{\text{cor}}^{\max}$

end if

return $\mathbf{S} = \{V, d_{\text{key}}, d_{\text{cor}}\}$

defined proportionally to the surrounding geometric volume, allowing the algorithm to adapt to various environments.

The parameter definition according to the spatial volume is described in Algorithm 1, and the results are shown in Fig. 5 . It should be noted that the pre-defined parameters in Algorithm 1 are based on Ouster 32-channel LiDAR and can be adjusted for other sensors.

4.1.2 | IMU pre-integration

Compared to wheeled or quadruped robots, unmanned aerial vehicles (UAVs) have ability to rotate in all three directions of roll, pitch, and yaw. However, this can lead to missed opportunities to detect common features along the vertical axis due to a limited Vertical Field of View (VFOV), especially when using LiDAR-based localization algorithms. To mitigate this problem, an initial guess value integrated with Inertial Measurement Unit (IMU) data is crucial when performing scan matching registration between two consecutive LiDAR frames. IMU pre-integration is a widely used method for this purpose in SLAM and LIO algorithms [11]. For a given measured angular velocity $\hat{\omega}_{\mathbf{B}}$ and acceleration $\hat{a}_{\mathbf{B}}$, the IMU raw measurement model can be defined as:

$$\hat{\omega}_{\mathbf{B}}(t) = \omega_{\mathbf{B}}(t) + b^{\omega}(t) + \eta^{\omega}(t) \tag{1}$$

$$\hat{a}_{\mathbf{B}}(t) = R_{\mathbf{WB}}^{\top}(t)(a_{\mathbf{W}}(t) - g_{\mathbf{W}}) + b^{a}(t) + \eta^{a}(t), \tag{2}$$

where $\omega_{\mathbf{B}}(t)$ and $a_{\mathbf{W}}(t)$ are actual states in body and world frame at time t and g is the gravity vector in **W** respectively. $b^{\omega}(t)$ and $b^{a}(t)$ are biases of the gyroscope and accelerometer respectively that can be modeled by random walk. Also, $\eta^{\omega}(t)$ and $\eta^{a}(t)$ are modeled as Gaussian white noise of the gyroscope and accelerometer. Note that, $R_{\mathbf{WB}}$ is the rotation matrix from **B** to **W** and $R_{\mathbf{WB}}^{\top} = R_{\mathbf{BW}}$. Using IMU measurement, the relative motion between t and $t + \delta t$ can be expressed as a discrete time system. Using properties such as $\dot{R}_{\mathbf{WB}} = R_{\mathbf{WB}}\omega_{\mathbf{B}}$, $\dot{v}_{\mathbf{W}} = a_{\mathbf{W}}$, and $\dot{p}_{\mathbf{W}} = v_{\mathbf{W}}$, relative rotation R, velocity v, and position p can be defined as follows.

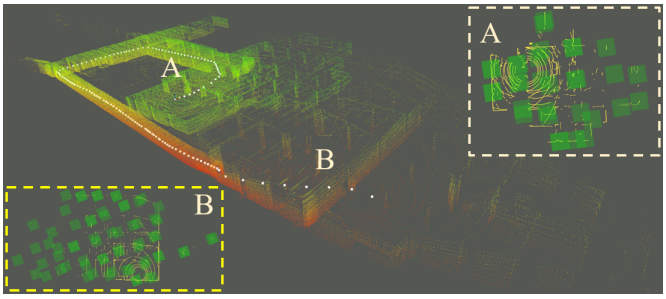


FIGURE 5 Keyframe generation interval according to the surrounding spatial volume. White dots and green voxels represent generated keyframes and $'\hat{\mathbf{Z}}_t^{V_{\text{env}}}$ in Algorithm 1, respectively. Area A is Tier 4 with $\mathbf{N}_t^{V_{\text{env}}} = 20$ and has a small d_{key}, while Area B is Tier 2 with $\mathbf{N}_t^{V_{\text{env}}} = 65$ and has a large d_{key}.

$$R(t+\delta t) = R(t)\text{Exp}\big((\hat{\omega}(t) - b^{\omega}(t) - \eta^{\omega}(t))\delta t\big) \tag{3}$$

$$v(t+\delta t) = v(t) + g\delta t + R(t)(\hat{a}(t) - b^{a}(t) - \eta^{a}(t))\delta t, \tag{4}$$

$$p(t+\delta t) = p(t) + v(t)\delta t + \frac{1}{2}g\delta t^2 + \frac{1}{2}R(t)(\hat{a}(t) - b^{a}(t) - \eta^{a}(t))\delta t^2. \tag{5}$$

Using (3), (4) and (5), we can define the IMU pre-integration measurement δR_{ij}, δv_{ij}, and δp_{ij} between the ith and jth frames as follows.

$$\delta R_{ij} \doteq \prod_{k=i}^{j-1} \text{Exp}\big((\hat{\omega}_k - b_k^{\omega} - \eta_k^{\omega})\delta t\big) \tag{6}$$

$$\delta v_{ij} \doteq \sum_{k=i}^{j-1} \delta R_{ik}(\hat{a}_k - b_k^{a} - \eta_k^{a})\delta t, \tag{7}$$

$$\delta p_{ij} \doteq \sum_{k=i}^{j-1} \big[\delta v_{ik}\delta t + \frac{1}{2}\delta R_{ik}(\hat{a}_k - b_k^{a} - \eta_k^{a})\delta t^2\big]. \tag{8}$$

We recommend the reader to refer to [11] for a thorough explanation of equations (3) through (8). The final optimization of the UAV's state is accomplished by employing the IMU pre-integration measurements δR_{ij}, δv_{ij}, and δp_{ij} in conjunction with the *scan to sub-map* matching registration results, while simultaneously updating the IMU biases b_k^{ω} and b_k^{a}. Notably, during the *scan to scan* matching registration stage, only δR_{ij} is employed as the initial guess, and it is assumed that δp_{ij} is negligible.

4.1.3 | Scan matching via Generalized-ICP

The proposed surrounding-adaptive LIO incorporates two types of scan matching registration. The first is the *scan to scan* matching registration, which estimates the UAV state between two consecutive keyframes.

The second is the *scan to sub-map* matching registration, which optimizes the UAV state in the global scale. We employ Generalized Iterative Closest Point (GICP) as a registration technique and the voxelization parameters are determined according to the surrounding spatial volume, as shown in Algorithm 1. This helps reduce the computational cost by using a large V in large areas and increases the accuracy by using a small V in narrower areas, balancing between accuracy and computational cost.

The Gaussian distribution model for input LiDAR points $\hat{\mathbf{Z}}_{\text{in}} = \{\hat{Z}_{\text{in},i}\}$ and target LiDAR points $\hat{\mathbf{Z}}_{\text{tar}} = \{\hat{Z}_{\text{tar},i}\}$ can be expressed as:

$$Z_{\text{in},i} \sim \mathcal{N}(\hat{Z}_{\text{in},i}, C_{\text{in},i}), \quad Z_{\text{tar},i} \sim \mathcal{N}(\hat{Z}_{\text{tar},i}, C_{\text{tar},i}), \tag{9}$$

where $C_{\text{in},i}$ and $C_{\text{tar},i}$ are covariance matrices. If there is no error or noise between the input points and the target points, we can express the perfect transformation $\mathbf{T}^* = [R|p]$ as follows.

$$\hat{\mathbf{Z}}_{\text{in}} = \mathbf{T}^* \hat{\mathbf{Z}}_{\text{tar}}, \quad \mathbf{T}^* \in \text{SE(3)} \tag{10}$$

Using (10), we can express the distance error term $d_i^{\mathbf{T}}$ for arbitrary transformation $\mathbf{T}$ as :

$$d_i^{\mathbf{T}} = Z_{\text{in},i} - \mathbf{T} Z_{\text{tar},i} \tag{11}$$

and we can define the Gaussian distribution model for $d_i^{\mathbf{T}^*}$ as follows.

$$\begin{aligned} d_i^{\mathbf{T}^*} &\sim \mathcal{N}(\hat{Z}_{\text{in},i} - (\mathbf{T}^*)\hat{Z}_{\text{tar},i}, \;\; C_{\text{in},i} + (\mathbf{T}^*) C_{\text{tar},i} (\mathbf{T}^*)^\top) \\ &= \mathcal{N}(0, C_{\text{in},i} + (\mathbf{T}^*) C_{\text{tar},i} (\mathbf{T}^*)^\top). \end{aligned} \tag{12}$$

Finally we can compute the $\mathbf{T}$ using the Maximum Likelihood Estimation (MLE) from (12) as follows.

$$\begin{aligned} \mathbf{T} &= \arg\max_{\mathbf{T}} \prod_i p(d_i^{\mathbf{T}}) \\ &= \arg\max_{\mathbf{T}} \sum_i \log(p(d_i^{\mathbf{T}})) \\ &= \arg\min_{\mathbf{T}} \sum_i {d_i^{(\mathbf{T})}}^\top (C_{\text{in},i} + \mathbf{T} C_{\text{tar},i} (\mathbf{T})^\top) d_i^{\mathbf{T}}. \end{aligned} \tag{13}$$

As mentioned previously, the proposed LIO performs *scan to scan* matching (indicated by the yellow line in Fig. 3) to estimate the UAV state between two consecutive keyframes. In this process, the last keyframe is used as the target, and the current LiDAR frame is used as the input. The last keyframe is denoted as $\hat{\mathbf{Z}}_{\text{key}}$, the current distance error term $d_t^{\mathbf{T}}$ with respect to the surrounding information $\mathbf{S}$ defined in Algorithm 1 can be expressed as follows:

$$\begin{aligned} d_{t,i}^{\mathbf{T}} &= {}'\hat{Z}_{t,i}^V - \mathbf{T}\,{}'\hat{Z}_{\text{key},i}^V \\ &\sim \mathcal{N}(0, \;\; C_{t,i} + \mathbf{T} C_{\text{key},i} (\mathbf{T})^\top). \end{aligned} \tag{14}$$

where $C_{t,i}$ and $C_{\text{key},i}$ are the covariance matrices. It should be noted that the points used to compute the distance error term $d_{t,i}^{\mathbf{T}}$ are selected based on the distance threshold d_{cor} specified in $\mathbf{S}$. With this information, the relative transformation $\delta\mathbf{T}$ obtained by the *scan to scan* matching registration can be computed as follows:

$$\delta\mathbf{T}_t = \arg\min_{\mathbf{T}} \sum_i {d_{t,i}^{(\mathbf{T})}}^\top (C_{t,i} + \mathbf{T} C_{\text{key},i} (\mathbf{T})^\top) d_{t,i}^{\mathbf{T}}. \tag{15}$$

If the position vector of $\delta\mathbf{T}_t$ is larger than the d_{key} in the $\mathbf{S}$, the target is updated with the newly generated keyframe as shown in Fig. 3 , and a *scan to submap* matching algorithm is performed to optimize the UAV state. When $\mathbf{T}^*$ represents the UAV state that was optimized from the last keyframe, the current UAV state $\mathbf{T}_t$ can be expressed as follows:

$$\mathbf{T}_t = \mathbf{T}^* \delta\mathbf{T}_t. \tag{16}$$

The proposed GICP-based scan matching using surrounding information $\mathbf{S}$ is described in Algorithm 2.

Algorithm 2 GICP-based scan matching using surrounding information

Require: $\hat{\mathbf{Z}}_t$, $\hat{\mathbf{Z}}_{\text{key}}$, C_t, C_{key}, δR_{ij}, and $\mathbf{S} = \{V, d_{\text{key}}, d_{\text{cor}}\}$
Ensure: $\mathbf{T}_t$, $\mathbf{T}^*$, and $\delta\mathbf{T}_t$

$\mathbf{T}_t, \mathbf{T}^*, \delta\mathbf{T}_t \leftarrow I$ ▷ Initialization
if New LiDAR scan is obtained **then**
  $\mathbf{S}$= $\{d_{\text{key}}, d_{\text{cor}}, V\} \leftarrow$ Algorithm 1($\hat{\mathbf{Z}}_t$)
  ${}'\hat{\mathbf{Z}}_t^V, {}'\hat{\mathbf{Z}}_{\text{key}}^V \leftarrow$ Voxelization($\hat{\mathbf{Z}}_t, \hat{\mathbf{Z}}_{\text{key}}$) with V
    ▷ Adaptive voxelization parameter
  $\delta\mathbf{T}_t^{\text{guess}} \leftarrow$ Initial Guess($\delta\mathbf{T}_{t-1}, \delta R_{t-1,t}$)
  GICP.BuildSource(${}'\hat{\mathbf{Z}}_t^V$, C_t)
  GICP.BuildTarget(${}'\hat{\mathbf{Z}}_{\text{key}}^V$, C_{key})
  GICP.Align($\delta\mathbf{T}_t^{\text{guess}}, d_{\text{cor}}$)
    ▷ Adaptive correspondence distance parameter
  $\delta\mathbf{T}_t \leftarrow$ **GICP**.GetTransform()
  $\mathbf{T}_t \leftarrow \mathbf{T}^* \delta\mathbf{T}_t$
  if $\|p(\delta\mathbf{T}_t)\| \geq d_{\text{key}}$ **then**
    ▷ Adaptive keyframe generation interval
    $\mathbf{T}^* \leftarrow$ Optimization() ▷ In Section. 4.1.4
    $\mathbf{T}_t \leftarrow \mathbf{T}^*$ ▷ UAV state update
    $\hat{\mathbf{Z}}_{\text{key}} \leftarrow \hat{\mathbf{Z}}_t$ ▷ Keyframe update
  end if
  $t \leftarrow t + 1$
end if
$p(\mathbf{T})$ *is the position vector* p *in* $\mathbf{T} = \begin{bmatrix} R & | & p \end{bmatrix}$

4.1.4 | Optimization

Existing SLAM algorithms typically utilize loop closure algorithms to optimize the state estimation when a robot revisits a previously explored area. However, in practical UAV operations in real-world environments, there is no guarantee that the UAV will return to its previous location. Even if it does return, the mission may fail due to the accumulation of a large state estimation error. Therefore, our main objective is to accurately estimate the UAV state in real-time without relying on loop closure algorithms. In Section 4.1.3, we described a GICP-based scan matching algorithm that utilizes surrounding information $\mathbf{S}$. However, relying solely on *scan to scan* matching for a prolonged period without optimization can lead to the accumulation of errors and difficulty in obtaining reliable results. To address this issue, we propose a method for optimizing the UAV state in the global map perspective by generating a sub-map when a keyframe is generated. For optimization, we perform *scan to submap* matching, and generate the sub-map using the kNN algorithm with the current UAV state in the keyframe management

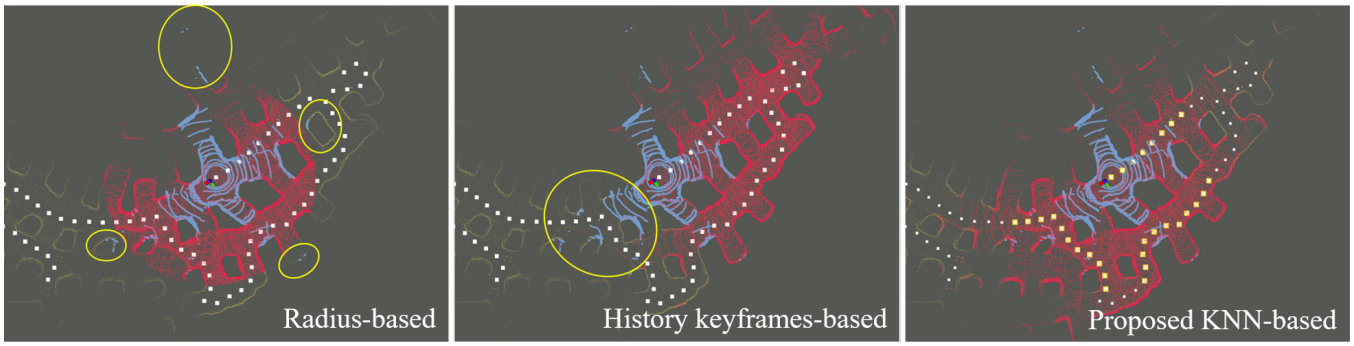


FIGURE 6 The sub-map generation results of radius-based, history keyframes-based, and the proposed kNN-based methods. The yellow points, red points, blue scan, white squares, and yellow squares denote the complete map points, sub-map points, the current scan, keyframe, and selected k(=30) keyframes, respectively. Despite the robot revisiting an area, the sub-map generated by the radius-based and 30 history keyframes-based methods fail to cover the entire current scan. The yellow circles indicate the non-overlapping current scan points in the sub-map.

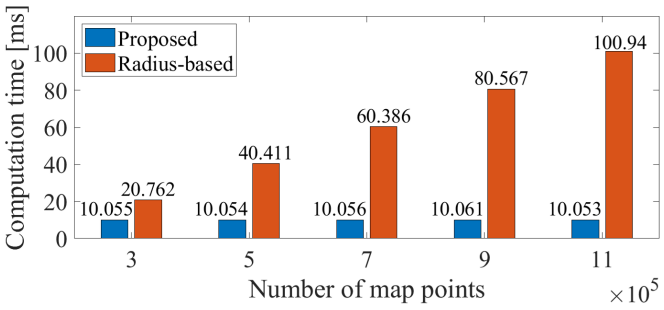


FIGURE 7 Comparison of computation time for sub-map generation. The radius-based method exhibits an increasing computational cost as the map size grows, in contrast to the proposed method which maintains a low computational cost. The history keyframe-based method, on the other hand, does not involve any specialized computation, but its optimization performance suffers as it takes into account only the most recently generated keyframes.

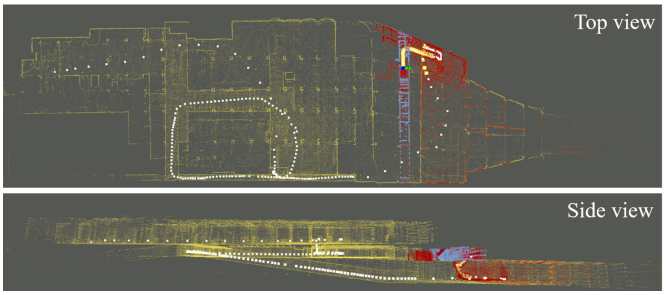


FIGURE 8 Optimized state via kNN-based sub-map generation in a multi-floor environment. The yellow points, red points, blue scan, white squares, and yellow squares represent ${}'\hat{\mathbf{M}}^{\mathrm{A}}$, ${}'\hat{\mathbf{M}}$, $({\mathbf{T}^*}'\hat{\mathbf{Z}}_t^V)$, $\mathbf{K}$, and $\mathbf{K}^{\mathrm{K}}$ in Algorithm. 3, respectively. The proposed LIO utilizes a keyframe generation interval and parameters for scan matching that adapt to the surrounding spatial volume. By optimizing from a global perspective in wide areas and from a local perspective in narrow areas, the proposed LIO ensures reliable localization performance in challenging environments.

module, as shown in Fig. 3 . Note that we use the kNN algorithm based on the Euclidean distance between $\mathbf{T}^*$s and $\mathbf{T}_t$, rather than between map points and $\mathbf{T}_t$. In existing SLAM algorithms, several sub-map generation methods have been proposed for global state optimization, such as the radius-based sub-map generation method or the method using a specific number of history keyframes. However, the radius-based sub-map generation method suffers from insufficient overlapping points at the edges of the current scan, which reduces scan matching performance and incurs increasing computation cost over time, as shown in Fig. 7 . The sub-map generation method using a specific number of history keyframes only considers recent keyframes, and therefore leads to continuous degradation in state estimation performance due to the assumption that errors accumulate. The sub-map generation results for each technique are presented in Fig. 6 .

In the proposed kNN-based sub-map generation method, k keyframes are selected based on their Euclidean distance from the current UAV state, without considering the order in which they were generated. This approach increases the likelihood of selecting keyframes with less accumulated error and improves the overall optimization performance in the global map perspective. When ${}'\hat{\mathbf{M}} = \{{}'\hat{M}_i\}$ is the sub-map generated by the proposed kNN-based method, the optimized UAV state $\mathbf{T}^*$ can be expressed as follows using equations (14) and (15).

$$\begin{aligned} d_{t,i}^{\mathbf{T}^*} &= {}'\hat{Z}_{t,i}^V - \mathbf{T}^{*\prime}\hat{M}_i^V \\ &\sim \mathcal{N}(0,\ C_{t,i} + \mathbf{T}^* C_{M,i} (\mathbf{T}^*)^\top) \end{aligned} \tag{17}$$

$$\mathbf{T}^* = \arg\min_{\mathbf{T}^*} \sum_i {d_{t,i}^{(\mathbf{T}^*)}}^\top (C_{t,i} + \mathbf{T}^* C_{M,i} (\mathbf{T}^*)^\top) d_{t,i}^{\mathbf{T}^*}. \tag{18}$$

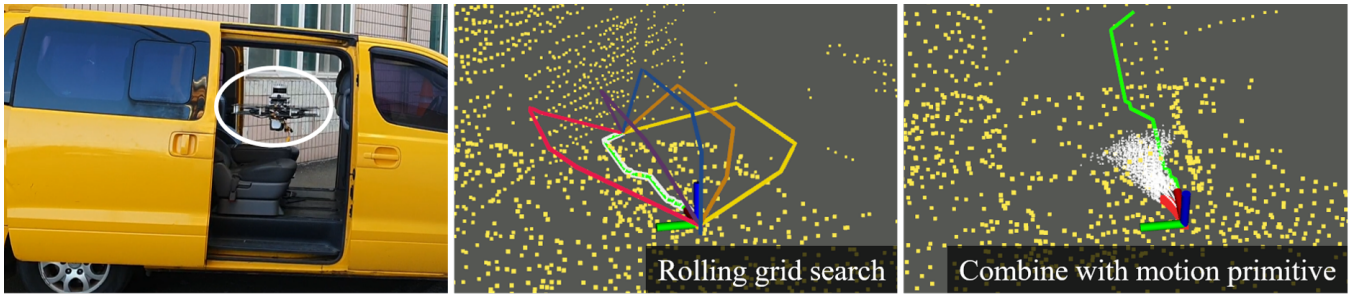


FIGURE 9 An overview of the proposed path planning algorithm. The rolling grid search-based method divides the 3D map into 2D maps rotated in the roll axis, and performs the A* algorithm in each 2D map. In the rolling grid search method, the color of each path planning result represents the result of the A* algorithm in each plane shown in Fig. 10 . Among a total of 6 paths, the best path (white bold line) is selected according to the cost function, and motion primitive-based obstacle avoidance is performed along the best path.

Finally, the optimized UAV state $\mathbf{T}^*$ is passed to the GICP-based scan matching module to compute $\mathbf{T}_t$ using *scan to scan* matching registration. The proposed kNN-based sub-map generation method is described in Algorithm 3, and its results are shown in Fig. 8 .

4.2 | Path Planner

Reliable 3D obstacle avoidance and safe path planning to the destination are essential for performing various missions such as exploration and guided flight in 3D complex environments. However, the 3D obstacle avoidance algorithm requires high computation cost for collision check with 3D map points or path planning to the destination, which is a challenging problem within the on-board computer. Therefore, we propose an algorithm that combines the motion primitive-based method and the grid search-based method for 3D path planning with low computation cost. In particular, for grid search-based path planning, we proposed a method of composing 3D map points into 2D grids rotated along the roll direction, which greatly reduced the computation cost compared to conventional 3D grid search-based methods. In addition, as a motion primitive-based method, we propose an approach from the perspective of the map points rather than the perspective of the path candidates to perform the collision check between more than 40,000 path candidates and 3D map points. This can greatly reduce the computation cost compared to the existing motion primitive-based method that perform point-by-point collision checks between map points and path candidates, and can generate as many path candidates as possible. An overview of the proposed path planning algorithm combining the grid search-based method and the motion primitive-based method is shown in Fig. 9 .

4.2.1 | Rolling grid search-based planning

The A* algorithm is a widely used grid search-based path planning method that is commonly used in unmanned platforms due to its reliable results. However, the 3D version of the A* algorithm has an exponentially higher computation cost than its 2D counterpart. This presents a challenge for practical operation, as reducing the search space or grid resolution can lead to issues such as the local minima problem or the algorithm becoming unable to operate in narrow areas. To effectively reduce computational cost, we propose a method of expressing a 3D map as 2D maps rotated in the roll direction, as depicted in Fig. 10 . From $'\hat{\mathbf{M}}$ generated by the kNN-based sub-map generation method in Algorithm 3, we can express the 2D maps $\mathbf{G}^r(\subset {}'\hat{\mathbf{M}}) = \{\mathbf{G}_i^r\}$ rotated by r degree in the roll direction as:

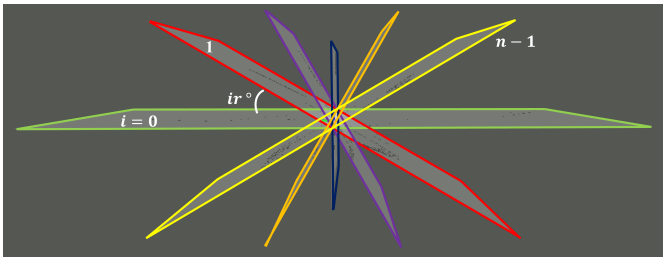


FIGURE 10 Rolling grid costmap generation result. We propose a novel approach to represent a 3D map by decomposing it into 2D maps rotated along the roll direction. Specifically, we divide the 3D map into n planes with an interval of r degrees in the roll direction. The size and resolution of each plane are determined based on the priority of flying in the horizontal plane. Moreover, we incorporate a safety distance that is proportional to the slope, i.e., the steeper the slope, the larger the safety distance.

$$\mathbf{G}_i^r = z\text{lim}_{d_z}(R_{x_{ir}}^{-1}\mathbf{T}_t^{-1\prime}\hat{\mathbf{M}}), \quad i = \{0, 1, \cdots, \frac{\pi}{r} - 1\}. \tag{19}$$

where $()^{-1}$, $R_{x_{ir}}$ and $z\text{lim}_{d_z}()$ are the matrix inversion, roll rotation matrix with ir degree, and filtering function for points where the absolute value of z is less than d_z, respectively and the results are shown in Fig. 11 . Among the paths $\mathbf{L}^r = \{\mathbf{L}_i^r\}$ generated by using the A* algorithm in $\mathbf{G}^r$ as shown in Fig. 12 , the best path $\mathbf{L}_{i_{\text{best}}}^r = \{L_{i,j}^r\}$ can

Algorithm 3 Optimization via KNN-based sub-map generation

Require: All keyframes $\mathbf{K} = \{K_i\}$, $\delta\mathbf{T}_t$, $\hat{\mathbf{Z}}_t$, δR_{ij}, and $\mathbf{S} = \{V, d_{\text{key}}, d_{\text{cor}}\}$

Ensure: Entire map points ${}'\hat{\mathbf{M}}^{\text{A}} = \{{}'\hat{\mathbf{M}}^{\text{A}}_i\}$, k-nearest neighbor keyframes $\mathbf{K}^{\text{K}}(\subset \mathbf{K})$, $\mathbf{T}^*$, and ${}'\hat{\mathbf{M}}$

```
T* ← I,  K, K^K ← ∅                                   ▷ Initialization
K_0 = p(T*) and 'M^A_0 = T*'Z^V_0                     ▷ Algorithm. 1
if ||p(δT_t)|| ≥ d_key then                           ▷ Optimization()
    KDtree.Build(K)
    KDtree.Search(T_t, k)
    {k_0, k_1, ··· , k_{k-1}} ← KDtree.ExtractIndex
    K^K ← {K_{k_0}, K_{k_1}, ··· , K_{k_{k-1}}}
    'M ← {'M^A_{k_0}, 'M^A_{k_1}, ··· , 'M^A_{k_{k-1}}}   ▷ Sub-map
    T^guess_t ← Initial Guess(T_{t-1}, δR_{t-1,t})
    GICP.BuildSource('Z^V_t, C_t)
    GICP.BuildTarget('M^V, C_M)                       ▷ Target update
    GICP.Align(T^guess_t, d_cor)                      ▷ Scan to sub-map
    T* ← GICP.GetTransform()
    New area = true
    for j = 0 to i − 1 do
        if ||p(T*) − K_j|| ≤ d_key then
            New area = false                          ▷ Visited area
            Break
        end if
    end for
    if New area then
        K.PushBack (p(T*))                            ▷ New keyframe
        'M^A.PushBack (T*'Z^V_t)                      ▷ Map update
    end if
end if
p(T) is the position vector p in T = [R | p]
```

be obtained using the following object function.

$$i_{\text{best}} = \arg\min_i \sum_j w_r \sin(ir + \epsilon_i)\|L^r_{i,j} - L^r_{i,j-1}\| + w_g(i - i_{\text{prev}}), \quad L^r_{i,j} \in \mathbb{R}^3. \tag{20}$$

Note that, w_r, w_g, i_{prev}, and ϵ represent the roll weight, group weight, previous best group, and roll bias along the plane, respectively. The proposed object function is divided into a path length term considering the rotation angle of each plane and a group term to prevent abrupt change of the selected plane that generates the path.

Once selected, $\mathbf{L}^r_{i_{\text{best}}}$ is maintained until the UAV reaches its destination or there is a collision in the newly updated ${}'\hat{\mathbf{M}}$, and is integrated with motion primitive-based methods.

4.2.2 | Motion primitive-based planning

The motion primitive-based obstacle avoidance algorithm selects the path with the minimum cost among collision-free paths through a collision check between path candidates generated offline and 3D map points. This method offers the advantage of being relatively simple compared to grid search-based and RRT algorithms, with small uncertainty in the path planning result. However, the computation cost of collision check with 3D map points increases exponentially as the number of motion primitive path candidates for a complex 3D environment increases, making it challenging to implement on a UAV with limited computational power. To solve this problem, we utilize an approach to the collision check algorithm from the perspective of 3D map points rather than the perspective of path candidates.

To generate paths offline, we refer to [48] and slightly modify the cost function to select the group with the minimum cost. The forward distance, maximum horizontal angle, maximum vertical angle, horizontal angle interval, and vertical angle interval for generating offline paths are denoted by d_f, HOR_{max}, VER_{max}, d_{HOR}, and d_{VER}, respectively. The total number of paths N_{path} is $N_{\text{path}}=(2\frac{\|\text{HOR}_{\text{max}}\|}{d_{\text{HOR}}}+1)^3(2\frac{\|\text{VER}_{\text{max}}\|}{d_{\text{VER}}}+1)^3$, and the maximum forward distance d_f^{max} is $3d_f$. When paths are generated, corresponding group list ζ^{list} is also generated in the same order. From the path list $\mathbf{U} = \{\mathbf{U}_i\}$, we can obtain the voxel grid list $\mathbf{V}^{\text{list}} = \{\mathbf{V}^{\text{list}}_j\}$ containing paths belonging to the grid, as shown in Algorithm 4.

Using $\mathbf{U}$ and $\mathbf{V}^{\text{list}}$ obtained from the offline processing, we can determine the collision-free path $\mathbf{U}^{\text{free}}$ that exists within ${}'\hat{\mathbf{M}}$, the 3D obstacle

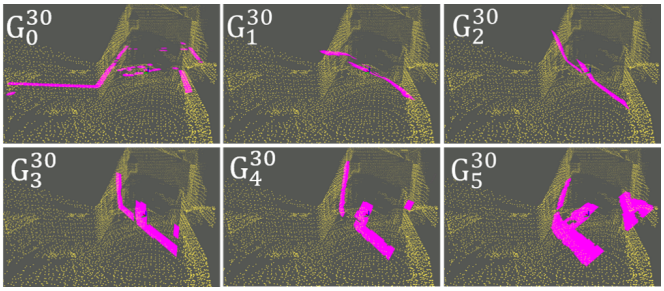


FIGURE 11 Results of generating the proposed 2D maps in world frame. Yellow map points and pink grids represent ${}'\hat{\mathbf{M}}$ and $\mathbf{G}^r_i$, respectively. Here, we use $d_z = 0.2$ and $r = 30$, and each $\mathbf{G}^r_i$ has a different map size and resolution for the safe and reliable path generation.

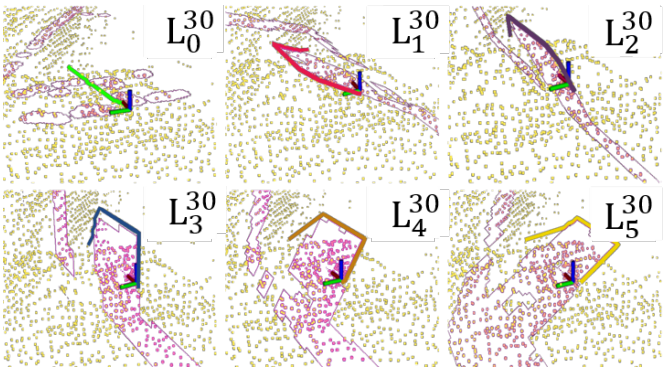


FIGURE 12 $\mathbf{L}^r_i$ results generated using the A* algorithm at each $\mathbf{G}^r_i$. Among $\mathbf{L}^r_i$, the best path is selected through the proposed object function.

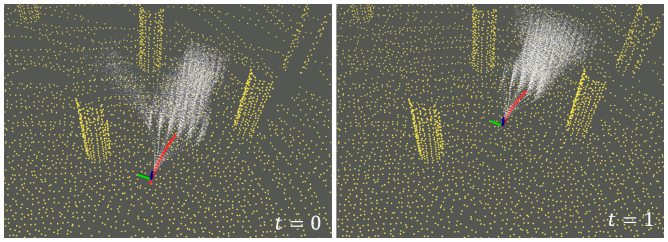


FIGURE 13 The proposed motion primitive-based obstacle avoidance results. The yellow map points, white paths, and red paths correspond to ′$\hat{\mathbf{M}}$, $\mathbf{U}^{\text{free}}$, and the executed path of o^{Best}, respectively.

Algorithm 4 Offline processing to obtain voxel grid list $\mathbf{V}^{\text{list}}$

Require: $\mathbf{U}$, maximum x,y, and z value of all paths x_{max}, y_{max}, z_{max}, voxel grid resolution v_{res}, and searching radius d_{s}
Ensure: $\mathbf{V}^{\text{list}}$ and path index list $\{K^p\}$

$j = 0$ ▷ Initialization
for $i = 0$ to $x_{\text{max}}/v_{\text{res}} - 1$ **do**
  $x = x_{\text{max}} - i * v_{\text{res}}$
  for $ii = 0$ to $2y_{\text{max}}/v_{\text{res}} - 1$ **do**
    $y = y_{\text{max}} - ii * v_{\text{res}}$
    for $iii = 0$ to $2z_{\text{max}}/v_{\text{res}} - 1$ **do**
      $z = z_{\text{max}} - iii * v_{\text{res}}$
      $\{K^p\} = \text{Rangesearch}(\mathbf{U}, \{x, y, z\}, d_{\text{s}})$
      $\mathbf{V}^{\text{list}}_j = \{K^p\}$
      $j = j + 1$
    end for
  end for
end for

Rangesearch$(\mathbf{U}, \{x, y, z\}, d_{\text{s}})$ *returns indices of* $\mathbf{U}$ *within the Euclidean distance* d_{s} *between* $\mathbf{U}$ *and* $\{x, y, z\}$.

map. To find the corresponding voxel index j for a given point x, y, z within ′$\hat{\mathbf{M}}$, we can use the reverse order of Algorithm 4, which is as follows:

$$x_{\text{ind}} = \frac{x_{\text{max}} + \frac{v_{\text{res}}}{2} - x}{v_{\text{res}}}, y_{\text{ind}} = \frac{y_{\text{max}} + \frac{v_{\text{res}}}{2} - y}{v_{\text{res}}}$$
$$z_{\text{ind}} = \frac{z_{\text{max}} + \frac{v_{\text{res}}}{2} - z}{v_{\text{res}}}, \tag{21}$$

$$j = (\frac{2y_{\text{max}}}{v_{\text{res}}} + 1)(\frac{2z_{\text{max}}}{v_{\text{res}}} + 1)x_{\text{ind}} + (\frac{2z_{\text{max}}}{v_{\text{res}}} + 1)y_{\text{ind}}z_{\text{ind}}. \tag{22}$$

Now, we can select the group with the minimum cost using collision free paths and the proposed cost function with j obtained in (22). The online process for selecting the group o^{Best} in ′$\hat{\mathbf{M}}$, a 3D obstacle map, is described in Algorithm 5 and its results are shown in Fig. 13 .

4.3 | Autonomous Exploration

Numerous exploration algorithms have been proposed recently for carrying out multiple missions in unknown areas. However, a crucial

Algorithm 5 Online processing to obtain o^{Best}

Require: ′$\hat{\mathbf{M}}$, $\mathbf{U}$, $\mathbf{V}^{\text{list}}$, ζ^{list}, group score list $\mathbf{O}$, and number of group k
Ensure: $\mathbf{U}^{\text{free}}$, o^{Best}, and collision paths $\mathbf{U}^{\text{collision}}$

$\mathbf{O} \leftarrow 0_{1\times k}$, k=$(2\frac{\|\text{HOR}_{\text{max}}\|}{d_{\text{HOR}}} + 1)(2\frac{\|\text{VER}_{\text{max}}\|}{d_{\text{VER}}} + 1)$
for All $\{x, y, z\}$ points in ′$\hat{\mathbf{M}}$ **do**
  if $\|x\| \le x_{\text{max}}$ and $\|y\| \le y_{\text{max}}$ and $\|z\| \le z_{\text{max}}$ **then**
    $j \leftarrow$ from (21) and (22)
    $\mathbf{U}^{\text{collision}}$.PushBack($\mathbf{U}[\mathbf{V}^{\text{list}}_j]$)
  end if
end for
$\mathbf{U}^{\text{free}} \leftarrow \mathbf{U} - \mathbf{U}^{\text{collision}}$
for All path index i in $\mathbf{U}^{\text{free}}$ **do**
  $\mathbf{O}[\zeta_i] +=$ Endpitch$(\mathbf{U}_i) *$ Endyaw$(\mathbf{U}_i)$
end for
$o^{\text{Best}} \leftarrow \arg\min_o \left(w_{\text{s}}\mathbf{O}[o] + w_g(o - o^{\text{prev}})\right)$

Endpitch$(\mathbf{U}_i)$ *and* Endyaw$(\mathbf{U}_i)$ *returns pitch and yaw value of end point of* $\mathbf{U}_i$*, respectively. Note that,* w_s, w_g, *and* o^{prev} *are state weight values, group weight value, and previous selected group, respectively.*

limitation of these studies is the lack of defined entry and exit points, rendering them impractical in real disaster scenarios. Therefore, the proposed exploration algorithm aims to overcome this limitation by introducing a hybrid operation that combines guided mode and exploration mode for practical search and rescue using UAVs. Furthermore, the algorithm is designed to conduct exploration only within a specific area designated by the operator. It is a low-computation cost approach that relies on the 3D map generated by the localization module and uses Euclidean distance-based clustering to segment the exploration area. An overview of the proposed exploration algorithm is shown in Fig. 15 . The exploration area's voxel map ′$\hat{\mathbf{M}}^{\text{E}}$, which is classified using the voxlization parameter V^{E}, can be obtained for the entire map ′$\hat{\mathbf{M}}^{\text{A}}$ generated by the localization module. The segmented exploration area ′$\hat{\mathbf{M}}^{\text{S}}(\subset {}'\hat{\mathbf{M}}^{\text{E}}) = \{'\hat{\mathbf{M}}^{\text{S}}_i\}$ can be obtained through Euclidean distance-based clustering, as described in Algorithm 6. The results are presented in Fig. 14 . Using the segmented exploration area ′$\hat{\mathbf{M}}^{\text{S}}$, we can define local goal points g_i corresponding to each ′$\hat{\mathbf{M}}^{\text{S}}_i = \{'\hat{M}^{\text{S}}_{i,j}\}$ as follows:

$$g_i = \sum_j \frac{'\hat{M}^{\text{S}}_{i,j}}{n}, \ j = \{0, 1, 2, \cdots .n-1\}, '\hat{M}^{\text{S}}_{i,j} \in \mathbb{R}^3 \tag{23}$$

As previously stated, the proposed exploration algorithm is operated as a hybrid with the guided mode, and the UAV first flies in the guided mode to the entrance set by the operator. The exploration score, considering the exit location $\mathbf{X}_{\text{exit}}$ set by the operator as prior information, can be defined as follows:

$$\textbf{ExplorationScore}(g_i) = \textbf{Volume}('\hat{\mathbf{M}}^{\text{S}}_i)\frac{w_{\text{exit}}\|g_i - \mathbf{X}_{\text{exit}}\|}{\|g_i - p(\mathbf{T}_t)\|} \tag{24}$$

Note that **Volume**(′$\hat{\mathbf{M}}^{\text{S}}_i$) represents the number of elements (j in (23)) in ′$\hat{\mathbf{M}}^{\text{S}}_i$, and w_{exit} denotes the weight for the distance to the exit. Finally, we pass g_i with the highest exploration score to our 3D path planning

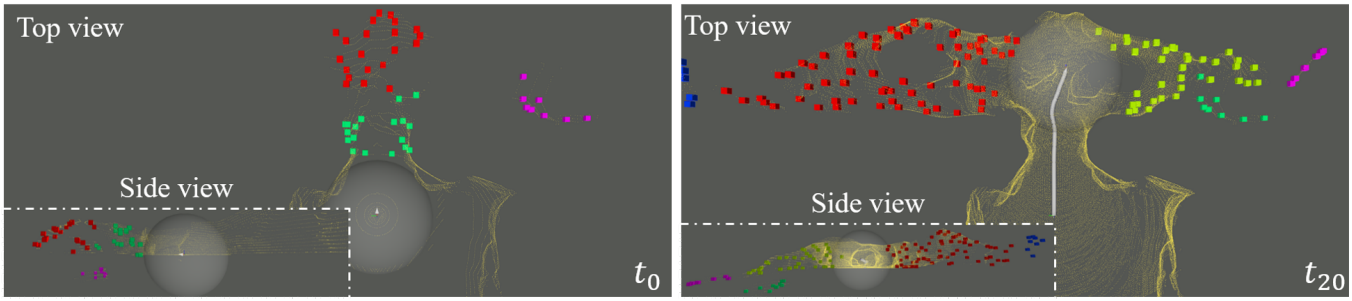


FIGURE 14 The result of segmented exploration area generation. The colored voxel represents segmented area $'\hat{\mathbf{M}}^{\mathsf{S}}$ to be explored, the yellow map points represent entire map $'\hat{\mathbf{M}}^{\mathsf{A}}$, and the gray circles indicate coverage d_{cover}. Note that, flight trajectories (white line) are used to identify uncovered areas based on the Euclidean distance.

Algorithm 6 Generate segmented exploration area $'\hat{\mathbf{M}}^{\mathsf{S}}$

Require: $'\hat{\mathbf{M}}^{\mathsf{A}}$, $\mathbf{K}$, V^{E}, coverage d_{cover}, and exploration area boundary $\mathbf{B}$
Ensure: Voxel map $'\hat{\mathbf{M}}^{\mathsf{E}}$ and segmented exploration area $'\hat{\mathbf{M}}^{\mathsf{S}}$

$'\hat{\mathbf{M}}^{V^{\mathsf{E}}} \leftarrow$ Voxelization($'\hat{\mathbf{M}}^{\mathsf{A}}$) with V^{E}
for All $\{x, y, z\}$ points in $'\hat{\mathbf{M}}^{V^{\mathsf{E}}}$ **do**
  if $\{x, y, z\}$ is outside $\mathbf{B}$ **then**
    Continue
  else
    Cover = $false$
    for All $\mathbf{K}$ **do**
      if $\|K_i - \{x, y, z\}\| \leq d_{\text{cover}}$ **then**
        Cover = $true$ ▷ Covered voxel
        Break
      end if
    end for
    if Cover= $false$ **then** ▷ Uncovered voxel
      $'\hat{\mathbf{M}}^{\mathsf{E}}$.Pushback($\{x, y, z\}$) ▷ Voxel map
    end if
  end if
end for
$'\hat{\mathbf{M}}^{\mathsf{S}} \leftarrow$ EuclideanClustering($'\hat{\mathbf{M}}^{\mathsf{E}}$)

module (discussed in Section 4.2), and the UAV explores the area while following the planned path.

4.4 | Onboard Object Detection and Localization

During the exploration mission, the target objects or artifacts within the area are visually detected by the UAV's front-facing camera. To analyze the objects, we utilized an EfficientDet-Lite2 model [34], which is trained incrementally using a manually collected dataset. The inference for detection is executed by leveraging the Google Edge TPU. The detected objects are then projected into the LiDAR scan, nearby objects

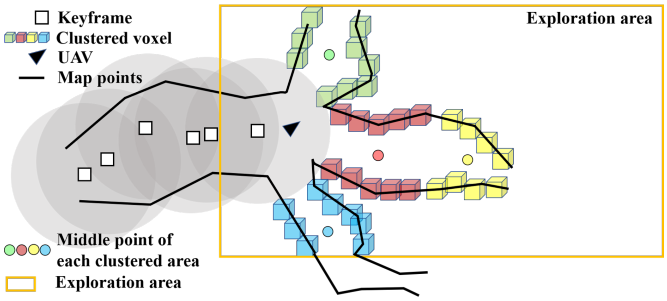


FIGURE 15 An overview of the proposed exploration algorithm. The UAV performs exploration only within the exploration area (orange box) set by the operator. Voxelization is performed on map points excluding the covered area (gray circle) corresponding to the UAV's trajectory, and areas to be explored are segmented through Euclidean distance-based clustering (colored voxel). The middle points (colored points) of each segmented area are designated as the UAV's local goal points, and the priority is determined according to the proposed exploration score

are clustered together to get the unique location per object. The proposed method provides efficient and low computation onboard object detection & localization pipeline.

4.4.1 | Object Detection

The object detection module in our UAV system utilizes the front-facing camera, and due to the absence of a dedicated GPU, we rely on the Google Edge TPU to perform inferencing on incoming images. Table 1 presents a comparison of various detection models supported by the Edge TPU. To select the model, we consider a maximum inference time of 100ms with the highest possible accuracy. We opt for the lightweight EfficientDet-Lite2 model [34], which has a low memory footprint, acceptable inference time, and is fully compatible with the Edge TPU. Utilizing the Edge TPU only requires one CPU from the onboard computer, freeing up the remaining CPUs for other modules such as SLAM and Planner. While the inference speed is usually 15-20 Hz, we limit it to 10 Hz as a faster inference rate is not necessary

given the assumption that the UAV is not moving very fast. This way, the UAV can detect the same object multiple times in one sweep, thereby reducing false-positive detections.

TABLE 1 Inference time comparison

Model	COCO Dataset		Building *A* Dataset	Building *B* Dataset
	Inference Time [ms]	mAP	Inference Time [ms]	Inference Time [ms]
EfficientDet-Lite0 [34]	54.4	30.4%	30.9	28.4
EfficientDet-Lite1 [34]	72.6	34.3%	42.4	40.2
EfficientDet-Lite2 [34]	123.3	36.0%	61.3	68.4
*EfficientDet-Lite3** [34]	144.8	39.4%	-	-
SSD Moblinet V2 [7]	10.5	25.6%	-	-
SSDLite MobileDet [43]	12.7	32.9%	-	-

**requires two TPUs*

4.4.2 | Object 3D Position Estimation

The positions of the detected objects are estimated by projecting their bounding boxes onto the current scan of the LiDAR sensor or the global point-cloud map generated by the localization module. In general, projecting the bounding box to the LiDAR scan is preferred due to its lower computational demand. However, if the bounding box is located outside the LiDAR's vertical field of view, the global map becomes necessary. The projection formula of the standard pinhole camera model $f_{proj} : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ can be presented as follows:

$$k \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{25}$$

where the f_u, f_v, u_0, v_0 are the camera model parameters, $[x, y, z]^T$ is the 3D point in the camera coordinate frame **C**, and u, v are the undistorted pixel coordinates in the image frame **I**. We assume that the transformation between the LiDAR coordinate frame **L** or world coordinate frame **W** to the camera coordinate frame **C** is known.

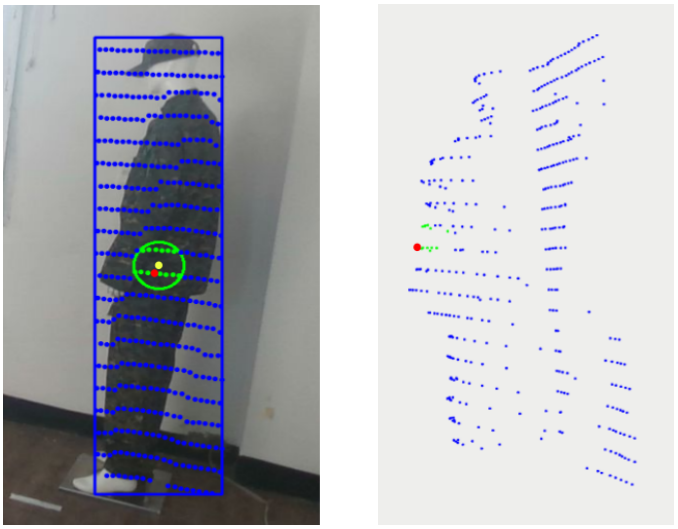
The input LiDAR scan is represented as a set of 3D points $\mathcal{S} = \{\mathbf{s}_i\}$ expressed in the camera coordinate frame C. Let us define a new set $\mathcal{S}_p$ containing 3D points whose projection $\mathbf{s}_I = [s_u, s_v]^T = f_{proj}(\mathbf{s})$ lies inside a certain pixel-radius r_p from the middle of the bounding box $b_0 = [b_u, b_v]^T$. The set can be representated as follows:

$$\mathcal{S}_p = \{\mathbf{s} \mid ||\mathbf{s}_I - \mathbf{b}_0||_2 \leq r_p, \quad \mathbf{s} \in \mathcal{S}\} \tag{26}$$

The 3D position of the detected object can be determined by selecting the nearest point from the camera in the z-direction from the set $\mathcal{S}_p$.

$$\begin{aligned} \hat{\mathbf{s}} &= \mathbf{s}_i, \\ \text{where} \quad i &= argmin\{z_s \mid \mathbf{s} \in \mathcal{S}_p\} \end{aligned} \tag{27}$$

where $\hat{\mathbf{s}}$ is the selected 3D point of the detection, $\mathbf{s}_i$ is the i-th item of $\mathcal{S}_p$ and z_s is the z component of $\mathbf{s}$. Choosing the point that lies in r_p circle is based on our observation that the center of the bounding box usually corresponds to the object while the bounding box corners correspond to the background. However, we do not select the exact center point of the bounding box because in some cases, such as when the object has a hollow center, the bounding box center may correspond to the background. Instead, we select the point in $\mathcal{S}_p$ with the lowest z-distance. Figure 16 provides a visualization of the 3D position estimation process.



(a) (b)

FIGURE 16 3D position estimation by projecting bounding box to the LiDAR point-cloud. (a) shows the LiDAR points inside the bounding box in blue, inside circle r_p in green, bounding box center in yellow, and the selected 3D point in red. (b) shows the reprojection of (a) in the 3D coordinate.

4.4.3 | Detection Association via Euclidean Clustering

To associate the detections with each other, we perform clustering of the detection points in 3D space using Euclidean distance. This is an intuitive and straightforward technique where points in close proximity are typically assumed to belong to the same object. We base this subsection mostly on the work of Rusu et al. [28], and for completeness, we will explain this method in detail.

Given the detection points set $\mathcal{D} = \{\mathcal{D}_c \mid c \in \mathcal{C}\}$ in the world frame W, where $\mathcal{C}$ is the set of object classes and $\mathcal{D}_c$ is the detection points for class c, we want to cluster the neighboring objects with the same class into the same cluster. First, we need to differentiate a detection point from another cluster. Let $O_i = \{\mathbf{d}_i \in \mathcal{D}_c\}$ be a distinct cluster from $O_j = \{\mathbf{d}_j \in \mathcal{D}_c\}$ if and only if the following condition is met:

$$min||\mathbf{d}_i - \mathbf{d}_j||_2 \geq d_{th,c} \tag{28}$$

where $d_{th,c}$ is the maximum distance threshold for class c, which is typically based on the size of the object. Additionally, we apply a filter to report a detected object only if it is based on a minimum number of

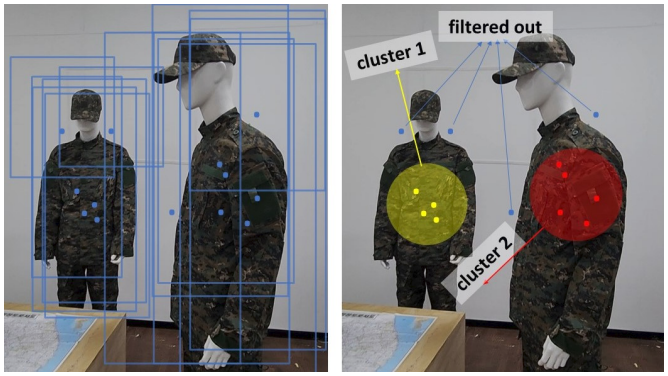


FIGURE 17 2D visualization of Algorithm 7. left image: detection bounding boxes and its reprojection point using method explained in 4.4.2. right image: Algorithm 7 result, the large red and yellow circles represent different clusters which will be reported as two objects, the small blue points will be filtered out due to lack of neighbours $n < n_c$ in the radius $d_{th,c}$.

detections n_c in a cluster $\mathcal{O}_c$. The value of n_c can vary across different object categories and is usually determined through an analysis of detection performance. A good rule of thumb for n_c is 5, which corresponds to detecting an object repeatedly over a period of 0.5 seconds. The complete process of estimating and reporting the 3D position of objects is presented in Algorithm 7.

Algorithm 7 Detection association via euclidean clustering

Input: Detection labels $\mathcal{C}$, detection 3D points $\mathcal{D} = \{\mathcal{D}_c \mid c \in \mathcal{C}\}$, distance threshold for each class $d_{th,c}$, minimum object per cluster for each class n_c

Output: Detection cluster for each class $\mathcal{O} = \{\mathcal{O}_c \mid c \in \mathcal{C}\}$

```
O ← ∅
for c ∈ C do
    O_c ← ∅                          ▷ list of clusters of class c
    Q_c ← ∅                          ▷ a queue of detection points of class c
    for d_i ∈ D_c do
        Q_c.PushBack(d_i)
        for d_i ∈ Q_c do
            search for neighbors set D_c^k of d_i in D_c
            with radius r ≤ d_th,c
            for d_i^k ∈ D_c^k do
                if d_i^k is not processed then
                    Q_c.PushBack(d_i^k)
                end if
            end for
        end for
        if Q_c.Size() ≥ n_c then
            O_c.PushBack(Q_c)
        end if
    end for
    O.PushBack(O_c)
end for
return O
```

5 | EXPERIMENTS

In order to evaluate the performance of our proposed exploration system, we conducted comprehensive analyses of each sub-system, including LIO, path planning, and exploration, in both simulated and real-world environments. To compare the accuracy and computation cost of the proposed LIO, we used ground truth values provided by ROS Gazebo in simulation environments. Furthermore, to evaluate the performance of the proposed exploration algorithm, we compared the exploration volume over time and exploration efficiency with GBPlanner2 [21], the current state-of-the-art algorithm. Exploration efficiency was defined as the median value of exploration ratio, and we defined exploration efficiency through the statistics of exploration ratio. In real-world environments, due to the lack of ground truth information, we measured the end-to-end (e2e) error by marking the start and end point of the UAV. To benchmark and analyze the proposed LIO, we compared it with state-of-the-art algorithms, including HDL-NDT [19], HDL-Fast GICP [19], LINS [27], LIO-SAM [32], and Fast-LIO2 [44]. Finally, we analyzed the exploration volume over time and exploration efficiency in the same manner as in the simulation environments.

TABLE 2 List of parameters used in simulation environments

Parameter	Value	Within	Parameter	Value	Within
$V_{\text{scan}}^{\max}$	1	Alg. 1	v_{res}	0.2	Alg. 4
$d_{\text{key}}^{\max}$	10	Alg. 1	$\text{HOR}_{\max}$	27°	Alg. 5
$d_{\text{cor}}^{\max}$	5	Alg. 1	$\text{VER}_{\max}$	27°	Alg. 5
α	0.6	Alg. 1	d_{HOR}	9°	Alg. 5
β	0.4	Alg. 1	d_{VER}	13.5°	Alg. 5
γ	0.4	Alg. 1	V^{E}	5	Alg. 6
k	30	Alg. 3	d_{cover}	10	Alg. 6
r	30	(19)			

5.1 | Simulation based evaluation

In this section, we present an analysis of the performance of the proposed system in simulation environments. Our system was tested in two simulated subterranean environments, Cave *01* and Cave *02*, provided by the DARPA Subterranean Virtual Competition, and operated in ROS Gazebo. The simulation utilized a 64-channel LiDAR sensor with a Horizontal Field of View (HFOV) of 360° and a Vertical Field of View (VFOV) of 30°, with IMU measurements provided by ROS Gazebo. We

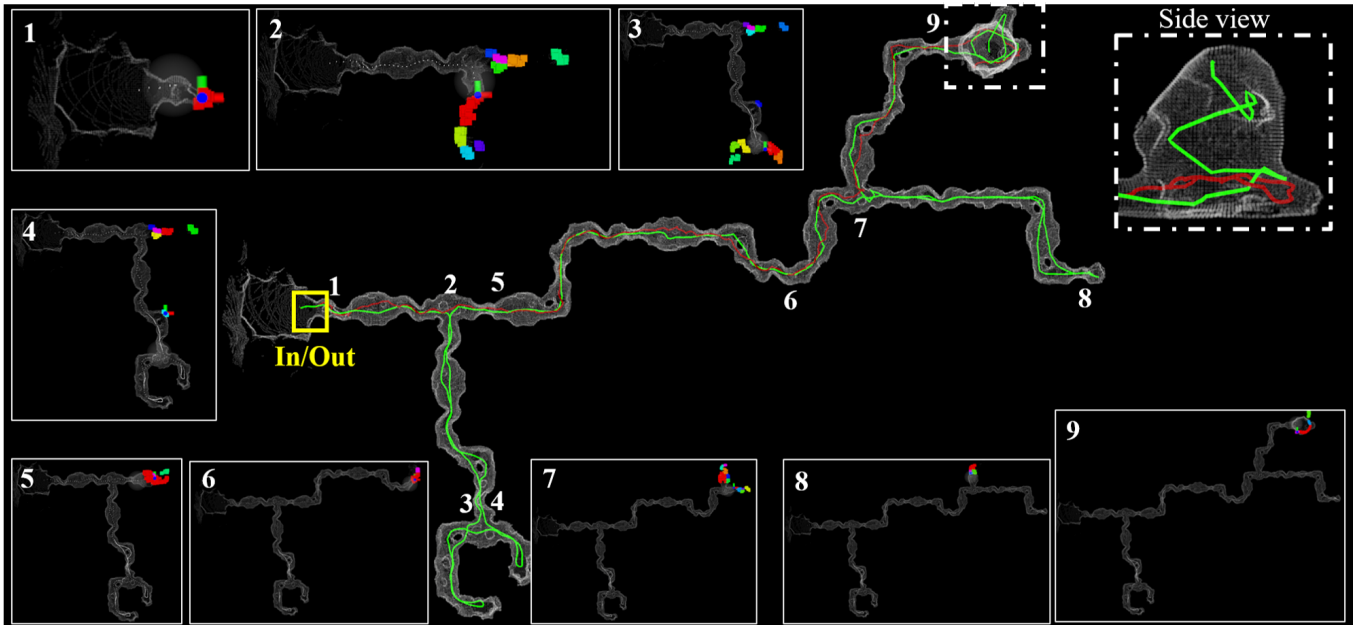


FIGURE 18 Results of the proposed UAV exploration system in the Cave *01* simulation environment. The green path, red path, and white map points are the trajectory using the proposed system, the trajectory using GB planner 2, and $'\hat{\mathbf{M}}^{\mathsf{A}}$, respectively. The colorful boxes in the sub-figure mean the segmented exploration area $'\hat{\mathbf{M}}^{\mathsf{S}}$ according to time. What is interesting is that the proposed method utilizes a map generated using the proposed LIO rather than a current scan, so as shown in the side view, reliable exploration is possible even within a environment with vertical features.

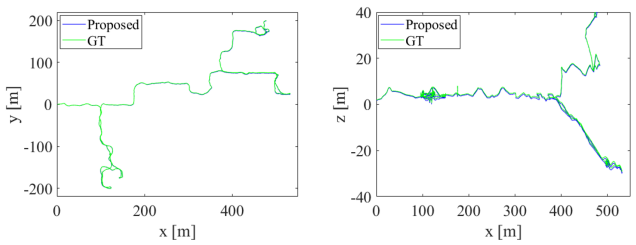


FIGURE 19 Comparison of state estimation results between the proposed LIO and ground truth in the Cave *01* environment.

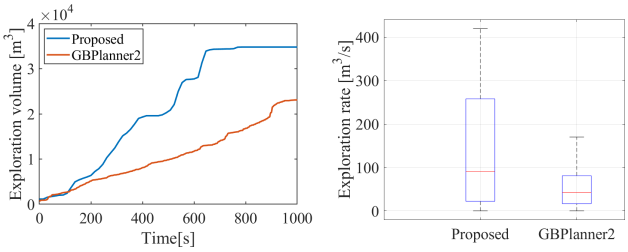


FIGURE 21 Comparison of exploration efficiency between the proposed exploration algorithm and GBPlanner2 in the Cave *01* environment.

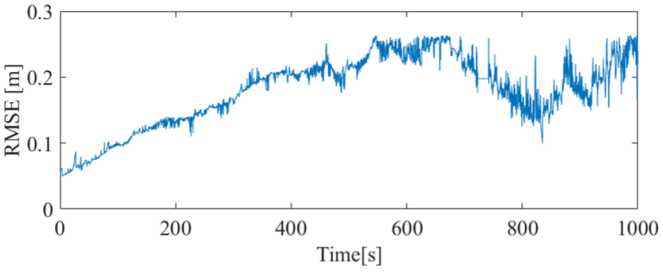


FIGURE 20 RMSE over time in a Cave *01* environment.

compared the performance of the proposed LIO with the ground truth provided by ROS Gazebo, and the proposed exploration algorithm with GBPlanner2. The maximum speed of the UAV was set to $2m/s$, and the exploration area was defined as all areas where $x \geq 10$. The entrance and exit positions in both simulation environments were set to the entrance of the cave. The parameters used for operating the proposed system in simulation environments are summarized in Table 2 .

The results of the proposed exploration system in the Cave *01* simulation environment is shown in Fig. 18 and the performances are shown in Fig. 19 , Fig. 20 , and Fig. 21 . As shown in Fig. 21 , the performance of the proposed exploration algorithm in the Cave *01* simulation environment achieved higher exploration volume and exploration efficiency than GBPlanner2. In particular, in terms of exploration efficiency, the proposed algorithm was 90 m^3/s, and the GBPlanner2 was 42 m^3/s,

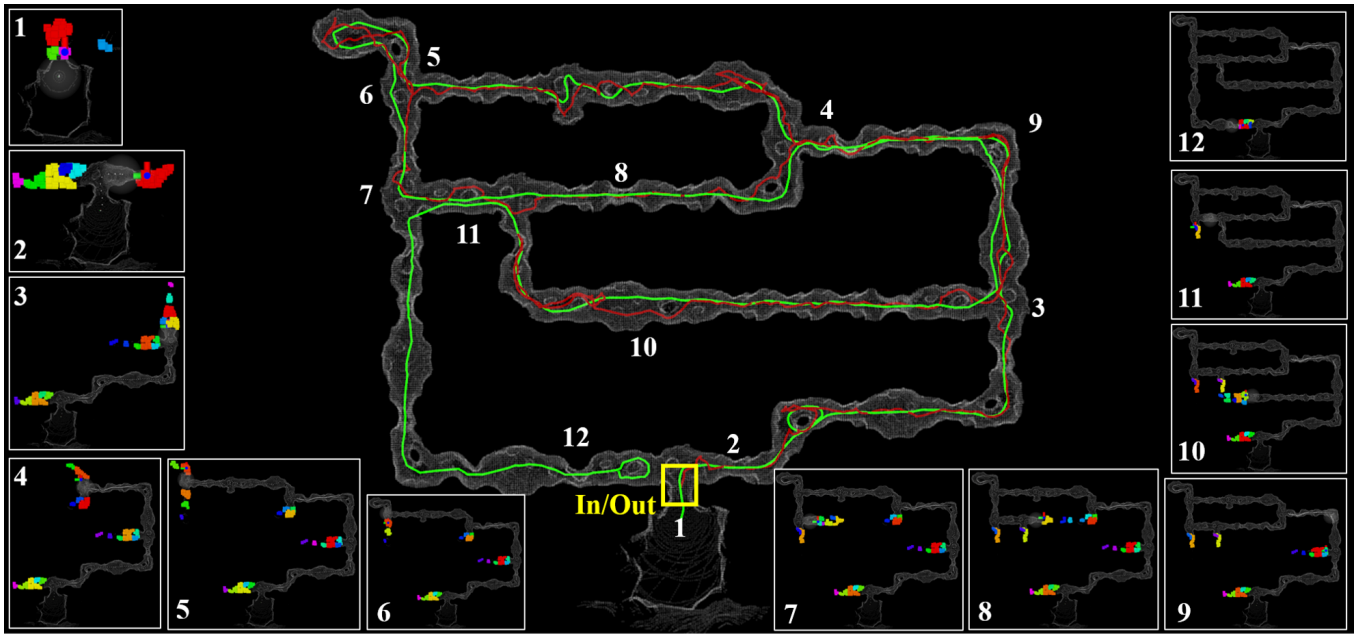


FIGURE 22 Results of the proposed UAV exploration system in the Cave *02* simulation environment. The green path, red path, and white map points are the trajectory using the proposed system, the trajectory using GB planner 2, and $'\hat{\mathbf{M}}^{A}$, respectively. The colorful boxes in the sub-figure mean the segmented exploration area $'\hat{\mathbf{M}}^{S}$ according to time. In this case, the proposed exploration algorithm showed high exploration efficiency with fewer back-and-forth maneuvers than GB planner2 due to the exit term defined in the proposed exploration score.

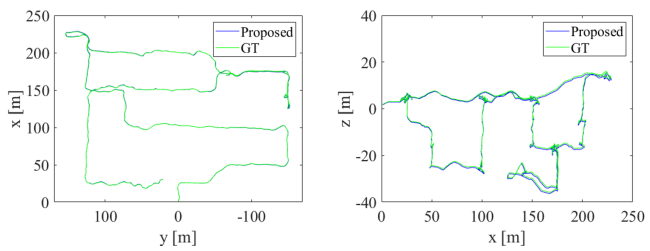


FIGURE 23 Comparison of state estimation results between the proposed LIO algorithm and ground truth in the Cave *02* environment

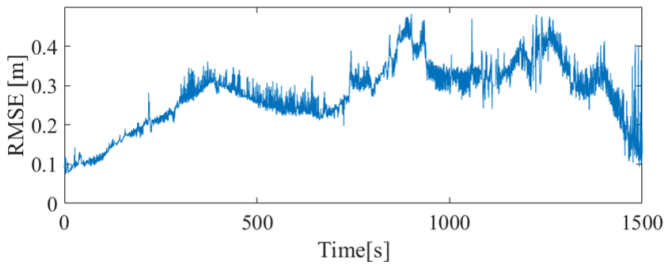


FIGURE 24 RMSE over time in a Cave *02* environment.

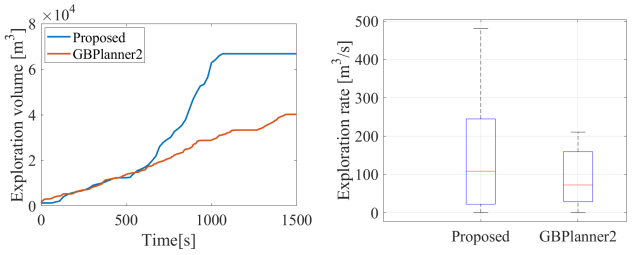


FIGURE 25 Comparison of exploration efficiency between the proposed exploration algorithm and GBPlanner2 in the Cave *02* environment

TABLE 3 The performance of the proposed LiDAR Inertial Odometry (LIO) in simulation environments

World	ATE [m]			RMSE [m]	Computation time [ms]	CPU [%]
	max	min	std			
Cave *01*	0.79	0.07	0.26	0.27	15.12	12.01
Cave *02*	0.97	0.14	0.17	0.39	14.98	12.13

showing about twice as high efficiency. We also analyzed the performance of the proposed system in the Cave 02 environment, which has higher complexity (many branches) than the Cave *01* environment, and the exploration results are shown in Fig. 22 and the performance is shown in Fig. 23 , Fig. 24 , and Fig. 25 . Similar to Cave *01*, in the Cave *02* environment, the exploration volume over time and exploration efficiency of the proposed exploration algorithm were higher than those of GBPlanner2, and the proposed algorithm achieved $108 m^3/s$, compared

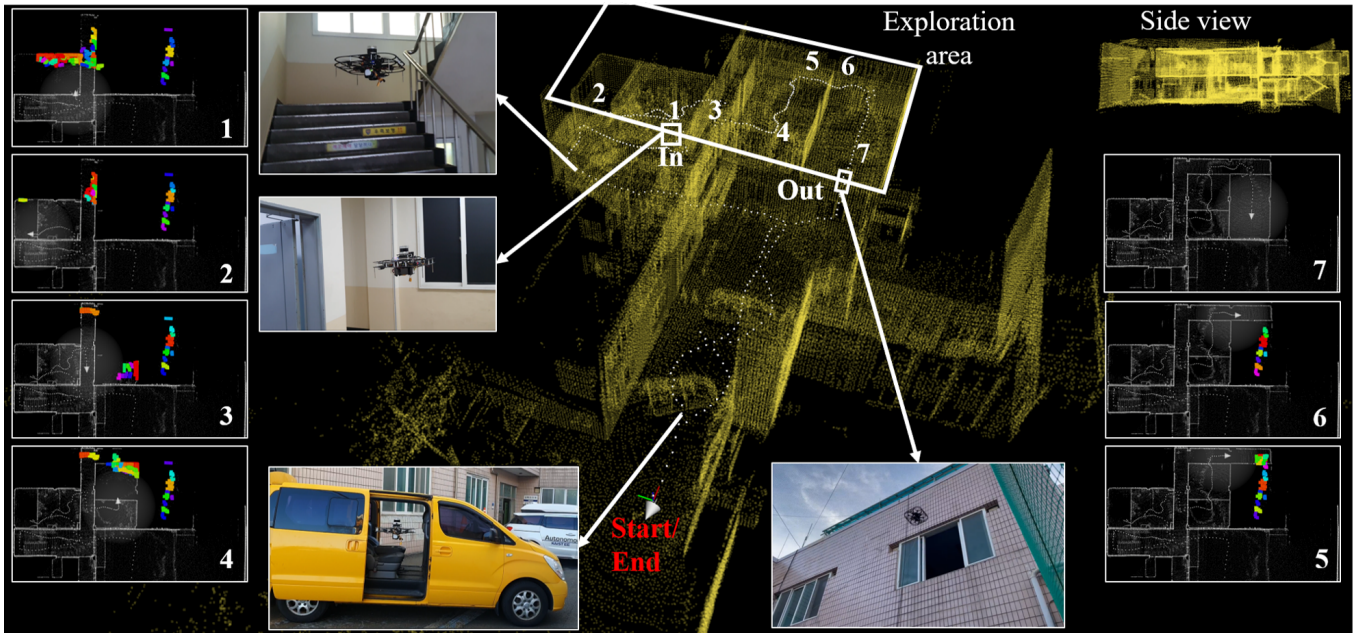


FIGURE 26 The results of the proposed UAV exploration system in Building *A* environment. The white squares and yellow map points in the figure represent $\mathbf{K}$ and ${}'\hat{\mathbf{M}}^{\mathrm{A}}$, respectively. The colorful boxes in the sub-figures show the segmented exploration area ${}'\hat{\mathbf{M}}^{\mathrm{S}}$ according to time. Additionally, the UAV thumbnails in the sub-figures depict the drone flight scenes at each location. In this environment, the entrance was set as a room on the second floor, and the drone was operated in guided mode until it reached the entrance. The exit was designated as a window on the second floor, and after completing the exploration within the designated exploration area, the drone escaped the building through the window and returned to the starting point.

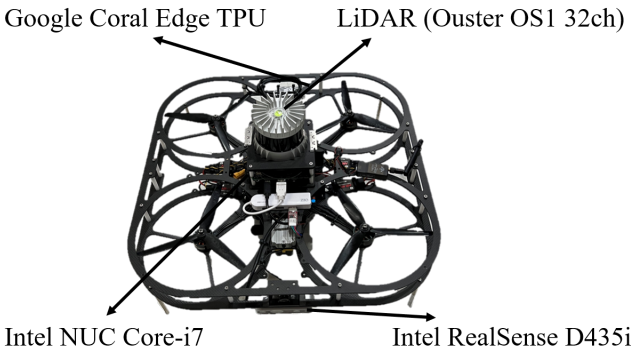


FIGURE 27 The configuration of aerial robot deployed in real-world experiments

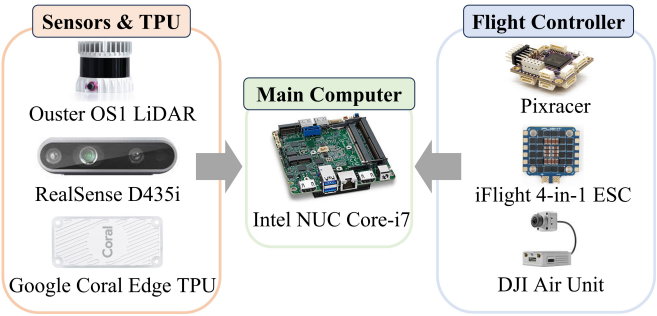


FIGURE 28 The hardware components of aerial robot deployed in real-world experiments

to 72 m^3/s for GBPlanner2. Also, as mentioned before, when operating the proposed exploration algorithm in simulation environments, we used the proposed LIO, and the results of the proposed LIO showed accurate state estimation peformance, as shown in Fig. 20 and Fig. 24 . The quantitative analysis of our LIO algorithm used while exploration in the Cave *01* and Cave *02* environments is shown in Table 3 . The Absolute Trajectory Error (ATE) between the ground truth provided by ROS Gazebo and the proposed LIO was a maximum of 0.79m, a minimum of 0.07m, and a standard deviation of 0.26 in the Cave *01* environment, and a maximum of 0.97m, minimum 0.14m, and standard deviation of 0.17 in the Cave *02* environment. In addition, the Root Mean Square Error (RMSE) was 0.27m and 0.39m in Cave *01* and Cave *02* environments, respectively, and the average state computation time per one LiDAR scan was 15.12ms and 14.98ms, respectively. Finally, the average CPU usage (with respect to one CPU) was 12.01% and 12.13%, respectively, and it was verified that the proposed LIO algorithm had high accuracy and low computation cost.

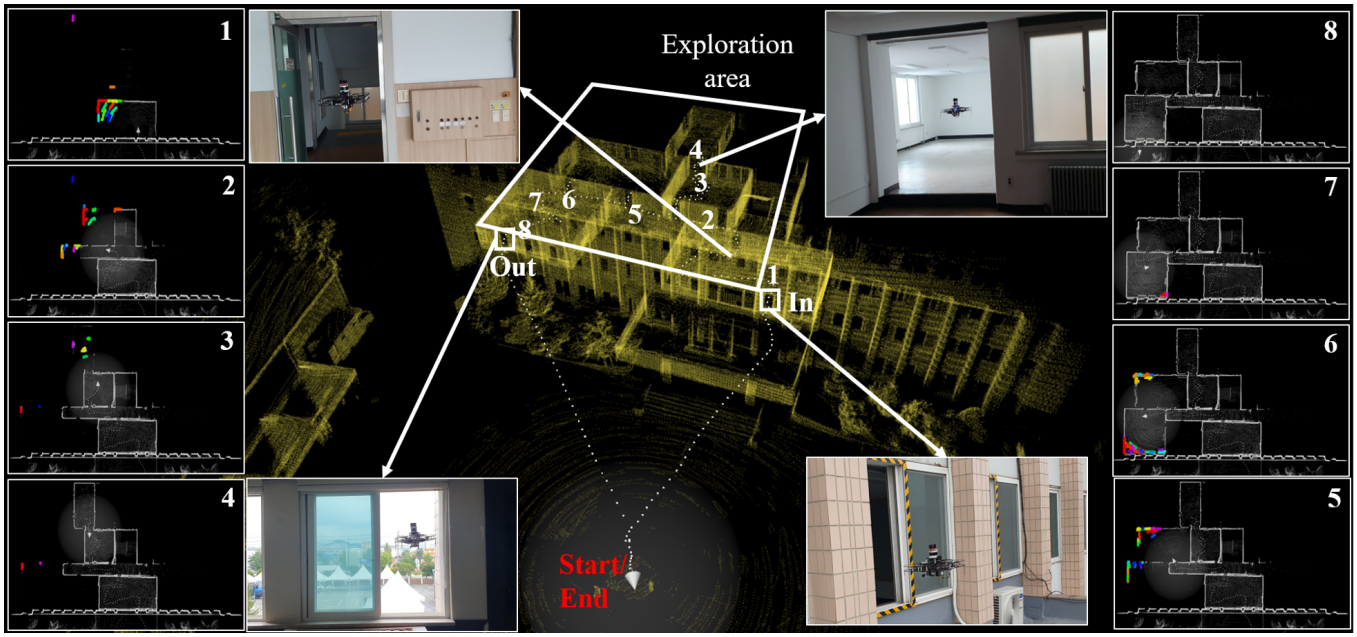


FIGURE 29 The results of the proposed UAV exploration system in Building *B* environment. The white squares and yellow map points in the figure represent $\mathbf{K}$ and $'\hat{\mathbf{M}}^{\mathrm{A}}$, respectively. The colorful boxes in the sub-figures show the segmented exploration area $'\hat{\mathbf{M}}^{\mathrm{S}}$ according to time. Additionally, the UAV thumbnails in the sub-figures depict the drone flight scenes at each location. In this environment, the entrance was designated as a second-floor window, and it was operated in guided mode until before. The exit was designated as another second-floor window, and after completing the exploration within the exploration area set by the operator, the drone escaped the building through the window and returned to the starting point.

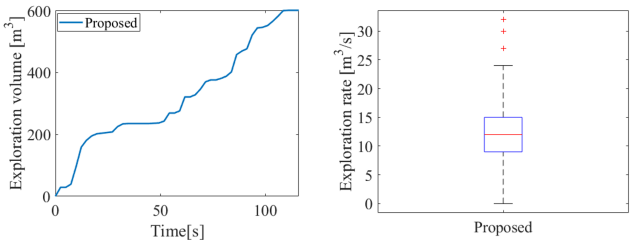


FIGURE 30 Exploration performances of the proposed exploration algorithm in the Building *A* environment

TABLE 4 List of parameters used in real-world environments

Parameter	Value	Within	Parameter	Value	Within
$V_{\mathrm{scan}}^{\max}$	1	Alg. 1	v_{res}	0.2	Alg. 4
$d_{\mathrm{key}}^{\max}$	10	Alg. 1	$\mathrm{HOR}_{\max}$	27°	Alg. 5
$d_{\mathrm{cor}}^{\max}$	5	Alg. 1	$\mathrm{VER}_{\max}$	27°	Alg. 5
α	0.6	Alg. 1	d_{HOR}	9°	Alg. 5
β	0.4	Alg. 1	d_{VER}	13.5°	Alg. 5
γ	0.4	Alg. 1	V^{E}	0.5	Alg. 6
k	30	Alg. 3	d_{cover}	5	Alg. 6
r	30	(19)			

5.2 | Experimental Evaluation

To verify the practicality of the proposed system and conduct further analysis, we conducted a real-world field test using a quadrotor aerial robot. The field test was conducted in two different buildings, Building *A* and Building *B*, with distinct characteristics. The drone platform used in the field test is shown in Fig. 27 . The drone is equipped with an Ouster 32-channel LiDAR, which provides a 360° horizontal field of view (HFOV) and a 45° vertical field of view (VFOV). The other components include Pixracer, a flight controller, and an Intel NUC computer with a 6-core i7-10710U CPU. The IMU measurements were provided by Pixracer, and we used the RealSense D435i camera for artifact detection and localization. The hardware components comprising the aerial robot are shown in Fig. 28 . All subsystems of the proposed system were performed in real-time on the onboard computer, and all flight tests were conducted in autonomous mode, without manual control by the operator.

Building *A* consists of stairs and narrow rooms, and was selected to validate the performance of LIO, exploration algorithm, and obstacle avoidance algorithm proposed in a multi-floor environment. Building *B* was selected to validate the practicality of the system proposed in a general building, as it is accessible only through the second-floor window and consists of a narrow and long corridor and complex rooms. In all field tests, the UAV was launched from outside a building, assuming a disaster environment, and was set to automatically return to its starting

TABLE 5 LiDAR Inertial Odometry (LIO) comparison results. (Red: Best, Blue: Second best)

Method	Building *A*			Building *B*		
	e2e Error[m]	Computation Time[s]	CPU[%]	e2e Error[m]	Computation Time[s]	CPU[%]
Proposed	0.08	11.89	12.14	0.06	13.32	12.82
HDL-NDT [19]	5.23	23.15	14.15	3.58	23.90	14.28
HDL-Fast GICP [19]	1.15	22.55	14.48	1.51	20.13	14.43
LINS [27]	0.85	12.28	13.13	8.57	14.71	11.58
LIO-SAM [32]	0.32	13.04	13.71	-	-	-
Fast-LIO2 [44]	0.12	10.49	11.02	0.13	10.03	13.11

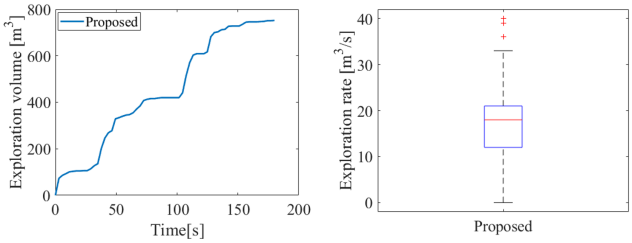


FIGURE 31 Exploration performances of the proposed exploration algorithm in the Building *B* environment

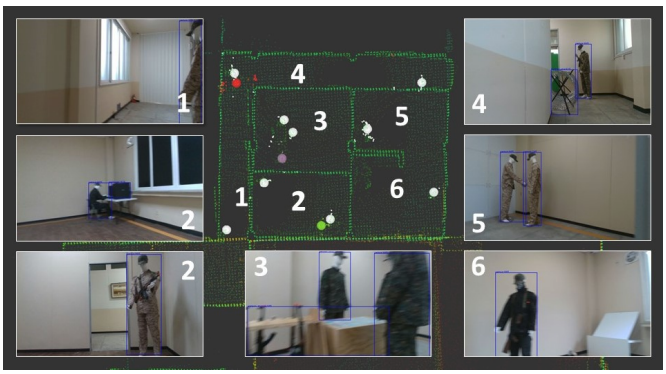


FIGURE 32 2D view of the object detection & localization results in the Building *A* environment. The small circles indicate the raw detections projected onto the floor plan, while the large circles represent the mean locations of the clustered detections. Each image is labeled with the corresponding location number, and the different colors represent different object categories.

point using a flight trajectory after exploration. The exploration results using the proposed system in Building *A* and Building *B* are shown in Fig. 26 and Fig. 29 , and the parameters of the proposed system used in the real-world field test are shown in the Table 4 . Note that, the maximum speed was set to $1.5m/s$ and the average speed was $0.7m/s$.

As shown in Fig. 26 , the UAV in Building *A* safely passed through the car door using the proposed 3D path planning algorithm and flew to the second floor through stair using the reliable state estimation result by the proposed LIO algorithm. The UAV, which flew in the guided mode to the entrance set by the operator on the second floor, performed exploration using the proposed exploration algorithm in the exploration area set by the operator, and finally escaped the building through the exit and returned to the start point. The exploration efficiency using the proposed exploration algorithm in the exploration area was $12m^3/s$, showing high exploration efficiency that completed the exploration within about $117s$ as shown in Fig. 30 . Also, as shown in Fig. 29 , the UAV in Building *B* flew in guided mode to the window, the entrance set by the operator, and completed the exploration within about $180s$ with a exploration efficiency of about $18m^3/s$ as shown in Fig. 31 . The final exploration volumes were about $600m^3$ and $750m^3$ in Building *A* and Building *B*, respectively, and all map points within the exploration area were fully generated. Interestingly, due to the exit location term in the exploration score we set, the UAV had fewer back-and-forth maneuvers in both field tests.

Finally, we compared the performance of the proposed LIO algorithm with several benchmark algorithms, such as end-to-end (e2e) error, average odometry computation time per one LiDAR scan, and CPU usage (with respect to one CPU), using data obtained from field test environments. As presented in Table 5 , the accuracy of the proposed LIO for both Building *A* and Building *B* was the highest, at $0.08m$ and $0.06m$, respectively. Moreover, the average computation time was the second-lowest at $11.89ms$ and $13.32ms$, respectively, following Fast-LIO2. The CPU usage was 12.14% in Building *A* and 12.82% in Building *B*, the second-lowest following Fast-LIO2 and LINS. Therefore, we verified that the proposed LIO algorithm is highly competitive in terms of accuracy, computational cost, and practicality.

The performance of the object detection and localization system was evaluated during exploration and the summary of the results is presented in Table 6 . In Building *A*, the system was able to detect and localize 12 out of 14 objects belonging to five different categories, namely *person*, *computer*, *communication device*, *weapon storage*, and *barbed wire*. The two false negatives occurred because objects of the same category were too close to each other and were clustered as one.

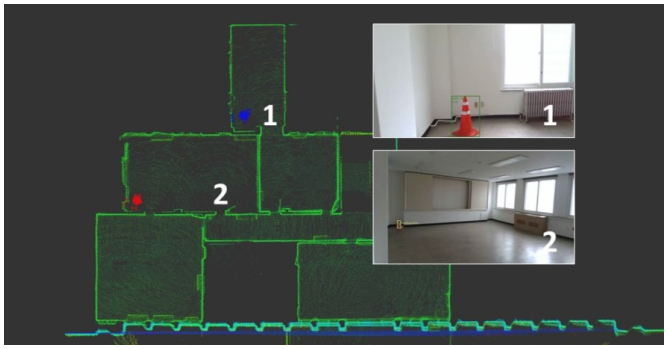


FIGURE 33 2D view of the object detection & localization results in the Building *B* environment. The small circles indicate the raw detections projected onto the floor plan, while the large circles represent the mean locations of the clustered detections. Each image is labeled with the corresponding location number, and the different colors represent different object categories.

TABLE 6 Object detection & localization performance

Building	Seen	Detected	Localized	F1 score
A	14	14	12	0.92
B	2	2	2	1.00

An example of incorrect clustering can be seen in Fig. 32 , where two people were clustered as one in location 5. Furthermore, the *communication device* category was not detected by the UAV's camera. The full pipeline of detection and localization achieved a F1 score of 0.92 in Building *A*. In Building *B*, which had objects belonging to two categories, namely *traffic cone* and *fire extinguisher*, the system successfully detected and localized all of the objects resulting in a perfect F1 score of 1.00. Overall, the detection and localization pipeline demonstrated great accuracy performance with limited computation resources.

6 | CONCLUSION

In summary, the proposed UAV system provides an efficient and reliable solution for exploring unknown environments, especially in disaster scenarios where human intervention is limited. The system includes a novel LIO algorithm that optimizes computational cost and accuracy through surrounding-adaptive keyframe generation and *scan to sub-map* matching. The 3D path planning and obstacle avoidance techniques ensure safe and reliable navigation in complex environments while maintaining low computation demands by utilizing 2D grids rotated in roll direction and motion primitives. The exploration algorithm based on 3D map points clustering enables fast and efficient exploration using exit-aided objective function. Furthermore, the system includes object detection and localization capabilities that enhance its practicality in real-world scenarios. Overall, the proposed system provides a comprehensive and practical solution for exploring unknown environments with limited human intervention.

References

[1] Besl, P. and N. D. McKay, 1992: A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **14**, no. 2, 239–256, doi:10.1109/34.121791.

[2] Biber, P. and W. Strasser, 2003: The normal distributions transform: a new approach to laser scan matching. *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 3, 2743–2748 vol.3.

[3] Bircher, A., M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, 2016: Receding horizon "next-best-view" planner for 3d exploration. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 1462–1468.

[4] Cao, C., H. Zhu, H. Choset, and J. Zhang, 2021: Tare: A hierarchical framework for efficiently exploring complex 3d environments. *Robotics: Science and Systems Conference (RSS)*, Virtual.

[5] Chao, C., Z. Hongbiao, C. Howie, and Z. Ji, 2021: Exploring large and complex environments fast and efficiently. *IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China.

[6] Chen, Y. and G. Medioni, 1991: Object modeling by registration of multiple range images. *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, 2724–2729 vol.3.

[7] Chiu, Y.-C., C.-Y. Tsai, M.-D. Ruan, G.-Y. Shen, and T.-T. Lee, 2020: Mobilenet-ssdv2: An improved object detection model for embedded systems. *2020 International Conference on System Science and Engineering (ICSSE)*, 1–5.

[8] Cieslewski, T., E. Kaufmann, and D. Scaramuzza, 2017: Rapid exploration with multi-rotors: A frontier selection method for high speed flight. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2135–2142.

[9] Dang, T., M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, and M. Hutter, 2020: Graph-based subterranean exploration path planning using aerial and legged robots. *Journal of Field Robotics*, **37**, no. 8, 1363–1388, wiley Online Library.

[10] Dharmadhikari, M., T. Dang, L. Solanka, J. Loje, H. Nguyen, N. Khedekar, and K. Alexis, 2020: Motion primitives-based path planning for fast and agile exploration using aerial robots. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 179–185.

[11] Forster, C., L. Carlone, F. Dellaert, and D. Scaramuzza, 2017: On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, **33**, no. 1, 1–21, doi:10.1109/tro.2016.2597321.
URL http://dx.doi.org/10.1109/TRO.2016.2597321

[12] Howard, A., M. Sandler, B. Chen, W. Wang, L. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, 2019: Searching for mobilenetv3. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE Computer Society, Los Alamitos, CA, USA, 1314–1324.
URL https://doi.ieeecomputersociety.org/10.1109/ICCV.2019.00140

[13] Howard, A. G., M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, 2017: *Mobilenets: Efficient convolutional neural networks for mobile vision applications*.
URL https://arxiv.org/abs/1704.04861

[14] Karaman, S. and E. Frazzoli, 2011: Sampling-based algorithms for optimal motion planning. *The International Journal or Robotics Research*, **30**, no. 7, 846–894.

[15] Kim, B., C. Jung, D. H. Shim, and A. akbar Agha-mohammadi, 2023: Adaptive keyframe generation based lidar inertial odometry for complex underground environments. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, accepted, To appear.

[16] Kim, G., S. Choi, and A. Kim, 2021: Scan context++: Structural place recognition robust to rotation and lateral variations in urban environments. *IEEE Transactions on Robotics*, accepted. To appear.

[17] Kim, G. and A. Kim, 2018: Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4802–4809.

[18] Kim, K. and H. S. Lee, 2020: Probabilistic anchor assignment with iou prediction for object detection. *ECCV*.

[19] Koide, K., J. Miura, and E. Menegatti, 2019: A portable three-dimensional lidar-based system for long-term and wide-area people behavior measurement. *International Journal of Advanced Robotic Systems*, **16**, doi:10.1177/1729881419841532.

[20] Koide, K., M. Yokozuka, S. Oishi, and A. Banno, 2021: Voxelized gicp for fast and accurate 3d point cloud registration. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 11054–11059.

[21] Kulkarni, M., M. Dharmadhikari, M. Tranzatto, S. Zimmermann, V. Reijgwart, P. De Petris, H. Nguyen, N. Khedekar, C. Papachristos, L. Ott, et al., 2022: Autonomous teamed exploration of subterranean environments using legged and aerial robots. *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 3306–3313.

[22] Lin, T., M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, 2014: Microsoft COCO: common objects in context. *CoRR*, **abs/1405.0312**.
URL http://arxiv.org/abs/1405.0312

[23] Liu, Z., H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, L. Dong, F. Wei, and B. Guo, 2022: Swin transformer v2: Scaling up capacity and resolution. *International Conference on Computer Vision and Pattern Recognition (CVPR)*.

[24] Ma, N., X. Zhang, H.-T. Zheng, and J. Sun, 2018: Shufflenet v2: Practical guidelines for efficient cnn architecture design. *Proceedings of the European Conference on Computer Vision (ECCV)*.

[25] Petrlik, M., P. Petrá ček, V. Krátký, T. Musil, Y. Stasinchuk, M. Vrba, T. Báča, D. Heřt, M. Pecka, T. Svoboda, and M. Saska, 2023: UAVs beneath the surface: Cooperative autonomy for subterranean search and rescue in DARPA SubT. *Field Robotics*, **3**, no. 1, 1–68, doi:10.55417/fr.2023001.
URL https://doi.org/10.55417%2Ffr.2023001

[26] Qiao, S., L.-C. Chen, and A. Yuille, 2021: Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10208–10219.

[27] Qin, C., H. Ye, C. E. Pranata, J. Han, S. Zhang, and M. Liu, 2020: Lins: A lidar-inertial state estimator for robust and efficient navigation. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 8899–8906.

[28] Rusu, R. B., 2009: *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD dissertation, Technische Universität München.

[29] Sandler, M., A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, 2018: MobileNetV2: Inverted Residuals and Linear Bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Salt Lake City, UT, 4510–4520.
URL https://ieeexplore.ieee.org/document/8578572/

[30] Segal, A., D. Haehnel, and S. Thrun, 2009: Generalized icp. *Robotics: Science and System*, Seattle, WA, volume 2, 435.

[31] Shan, T. and B. Englot, 2018: Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 4758–4765.

[32] Shan, T., B. Englot, D. Meyers, W. Wang, C. Ratti, and R. Daniela, 2020: Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 5135–5142.

[33] Tan, M. and Q. V. Le, 2019: Efficientnet: Rethinking model scaling for convolutional neural networks. doi:10.48550/ARXIV.1905.11946.
URL https://arxiv.org/abs/1905.11946

[34] Tan, M., R. Pang, and Q. V. Le, 2020: Efficientdet: Scalable and efficient object detection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[35] Tian, Z., C. Shen, H. Chen, and T. He, 2019: Fcos: Fully convolutional one-stage object detection. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[36] — 2022: FCOS: A simple and strong anchor-free object detector. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **44**, no. 4, 1922–1933, doi:10.1109/TPAMI.2020.3032166.

[37] Tranzatto, M., M. Dharmadhikari, L. Bernreiter, M. Camurri, S. Khattak, F. Mascarich, P. Pfreundschuh, D. Wisth, S. Zimmermann, M. Kulkarni, V. Reijgwart, B. Casseau, T. Homberger, P. De Petris, L. Ott, W. Tubby, G. Waibel, H. Nguyen, C. Cadena, R. Buchanan, L. Wellhausen, N. Khedekar, O. Andersson, L. Zhang, T. Miki, T. Dang, M. Mattamala, M. Montenegro, K. Meyer, X. Wu, A. Briod, M. Mueller, M. Fallon, R. Siegwart, M. Hutter, and K. Alexis, 2022: *Team cerberus wins the darpa subterranean challenge: Technical overview and lessons learned*.
URL https://arxiv.org/abs/2207.04914

[38] Vrba, M., D. Heřt, and M. Saska, 2019: Onboard marker-less detection and localization of non-cooperating drones for their safe interception by an autonomous aerial system. *IEEE Robotics and Automation Letters*, **4**, no. 4, 3402–3409, doi:10.1109/LRA.2019.2927130.

[39] Wang, C.-Y., A. Bochkovskiy, and H.-Y. Liao, 2022: YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*.

[40] Wang, C.-Y., A. Bochkovskiy, and H.-Y. M. Liao, 2021: Scaled-YOLOv4: Scaling cross stage partial network. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 13029–13038.

[41] Wang, C.-Y., I.-H. Yeh, and H.-Y. M. Liao, 2021: You only learn one representation: Unified network for multiple tasks. *arXiv preprint arXiv:2105.04206*.

[42] Wang, H., C. Wang, C. Chen, and L. Xie, 2020: F-loam : Fast lidar odometry and mapping. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[43] Xiong, Y., H. Liu, S. Gupta, B. Akin, G. Bender, Y. Wang, P.-J. Kindermans, M. Tan, V. Singh, and B. Chen, 2021: MobileDets: Searching for Object Detection Architectures for Mobile Accelerators. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Nashville, TN, USA, 3824–3833.
URL https://ieeexplore.ieee.org/document/9578555/

[44] Xu, W., Y. Cai, D. He, J. Lin, and F. Zhang, 2021: FAST-LIO2: fast direct lidar-inertial odometry. *CoRR*, **abs/2107.06829**.
URL https://arxiv.org/abs/2107.06829

[45] Xu, W. and F. Zhang, 2020: FAST-LIO: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter. *CoRR*, **abs/2010.08196**.
URL https://arxiv.org/abs/2010.08196

[46] Yamauchi, B., 1997: A frontier-based approach for autonomous exploration. *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, 146–151.

[47] Yokozuka, M., K. Koide, S. Oishi, and A. Banno, 2020: Litamin: Lidar-based tracking and mapping by stabilized icp for geometry approximation with normal distributions. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5143–5150.

[48] Zhang, J., C. Hu, R. G. Chadha, and S. Singh, 2020: Falco: Fast likelihood-based collision avoidance with extension to human-guided navigation. *Journal of Field Robotics*, **37**, no. 8, 1300–1313.

[49] Zhang, J. and S. Singh, 2014: LOAM: lidar odometry and mapping in real-time. *Robotics: Science and Systems X, University of California, Berkeley, USA, July 12-16, 2014*, D. Fox, L. E. Kavraki, and H. Kurniawati, Eds.
URL http://www.roboticsproceedings.org/rss10/p07.html

[50] Zhang, X., X. Zhou, M. Lin, and J. Sun, 2018: Shufflenet: An extremely efficient convolutional neural network for mobile devices. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

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

[52] Zhou, L., G. Huang, Y. Mao, J. Yu, S. Wang, and M. Kaess, 2022: $\mathcal{PLC}$-lislam: Lidar slam with planes, lines, and cylinders. *IEEE Robotics and Automation Letters*, **7**, no. 3, 7163–7170, doi:10.1109/LRA.2022.3180116.

[53] Zhou, X., V. Koltun, and P. Krähenbühl, 2021: Probabilistic two-stage detection. *arXiv preprint arXiv:2103.07461*.

[54] Zong, Z., G. Song, and Y. Liu, 2022: Detrs with collaborative hybrid assignments training. *arXiv preprint arXiv:2211.12860*.