IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 21, 2020, accepted January 13, 2021, date of publication January 27, 2021, date of current version March 1, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3055066

Multilayer Mapping Kit for Autonomous UAV Navigation

SHENGYANG CHEN[1], HAN CHEN[2], (Graduate Student Member, IEEE), CHING-WEI CHANG[1], AND CHIH-YUNG WEN[1,2,3]

[1]Department of Mechanical Engineering, The Hong Kong Polytechnic University, Hong Kong
[2]Interdisciplinary Division of Aeronautical and Aviation Engineering, The Hong Kong Polytechnic University, Hong Kong
[3]Research Institute for Sustainable Urban Development, The Hong Kong Polytechnic University, Hong Kong

Corresponding author: Chih-Yung Wen (cywen@polyu.edu.hk)

This work was supported by the Research Institute for Sustainable Urban Development, The Hong Kong Polytechnic University, through the Emerging Frontier Area (EFA) Scheme.

ABSTRACT Mapping, as the back-end of perception and the front-end of path planning in the modern UAV navigation system, draws our interest. Considering the requirements of UAV navigation and the features of the current embedded computation platforms, we designed and implemented a novel multilayer mapping framework. In this framework, we divided the map into three layers: awareness, local, and global. The awareness map is constructed in cylindrical coordinate, enabling fast raycasting. The local map is a probability-based volumetric map. The global map adopts dynamic memory management, allocating memory for the active mapping area, and recycling the memory from the inactive mapping area. We implemented this mapping framework in three parallel threads: awareness thread, local-global thread, and visualization thread. Finally, we evaluated the mapping kit in both the simulation environment and the real-world scenario with the vision-based sensors. The framework supports different kinds of map outputs for the global or local path planners. The implementation is open-source for the research community.

INDEX TERMS Mapping, reconstruction, unmanned aerial vehicle, navigation, simultaneous localization and mapping, navigation.

I. INTRODUCTION

The autonomous UAV navigation system senses the environment and reacts to it accordingly, so that the UAV can move from one place to another safely even in an unknown environment. A typical navigation system consists of a series of modules of localization, mapping and path planning. The localization module estimates the motion from the on-board sensors while the path planning module plans a safe trajectory [1], [2]. The mapping kit aims to bridge the gap between the localization and path planning modules. It provides a perception ability to reconstruct the surroundings, and the reconstructed map is the foundation for safe and efficient path planning. A good UAV navigation oriented mapping kit should satisfy the following requirements:

- Bridge the gap between perception and reaction modules and provide enough information for path planning.
- Provide human-readable representation. In UAV applications, humans often provide the high-level navigation goal of the UAV, and therefore the mapping kit should contain a visualization module.
- Be robust with respect to the sensor noise. The mapping kit must deal with the uncertainties caused by the depth measurement noises from the on-board sensors.
- Support a large mapping area with a limited overhead of memory.
- Show good dynamic performance. The map should update the occupancy information of newly detected objects within a certain period of time.
- Can be processed on the limited resources of the onboard computation platform in real-time.

In previous work, many mapping investigations have focused on balancing the accuracy of the map and the

The associate editor coordinating the review of this manuscript and approving it for publication was Heng Wang.

overhead of storage and on achieving fast access to the map elements. The approaches for meeting these objectives include storing the map with the octrees data structure [3]–[5], applying hash table to speed up the searching process [6], and introducing the signed distance field (SDF) information to achieve sub voxel accuracy of the surface reconstruction [7].

Meanwhile, an examination of the contemporary embedded systems shows that the memory size is no longer the bottleneck of the embedded systems. We consider the task of creating a navigation map in a $50m \times 50m \times 5m$ field with a voxel dimension of $0.2m \times 0.2m \times 0.2m$. If every voxel stores 64 bytes of information, the memory needed for such a map is merely 100 MB. An embedded computer such as Raspberry Pi 4 has a 4 GB memory. Therefore, the storage overhead is a minor concern in the current mapping operation. The dynamic memory management will be applied to the sub-map instead of every single voxels.

In this novel mapping framework (Figure 1), voxels are stored in an organized data structure. Every voxel is indexed with its position information and can be modified directly. Hence, the modification and update of the map can be fast.

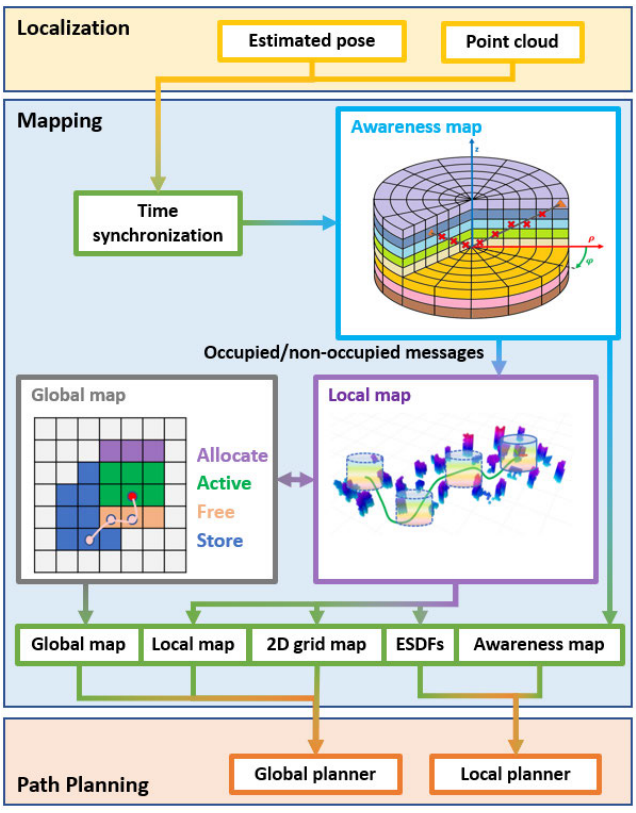


FIGURE 1. Overview of multilayer mapping framework. The input of the mapping framework consists of the point cloud and the estimated pose. The output of the mapping framework consists of different representations of the environment, that can be used for path planning.

In addition, depth measurement always involves some uncertainties and random noises. The framework takes two measurements to reduce the uncertainties and increase the map's robustness with the raycasting-based visibility check and probability-based occupancy state. The measurement data from the sensor only include the occupied information. Applying the visibility check in the awareness map enables measurement with the non-occupied information. The update of the local map is based on both occupied and non-occupied information. The local map occupancy state is represented with a probability value that increases with the occupied measurements and decreases with the non-occupied measurements. Even if uncertainties exist in every measurement, the robust estimation of the occupancy state can still be fused from multiple measurements.

In the system implementation, we decoupled the visualization part from the mapping kit, so that the onboard computer only focuses on the mapping task and the heavy-load visualization task is conducted by a stand-along thread on the ground station computer.

To summarize, the contributions of this work are as follows:

- Designed an awareness-local-global three levels mapping framework for UAV navigation. The framework supports various kinds of map output format including 2D grid map, global map, local ESDFs, and local map.
- Adopted different measurements to increase the robustness and feasibility of the mapping kit. These measurements include efficient raycasting-based visibility check, probability-based representation of the occupancy state and dynamic memory management.
- Tested the mapping framework in the simulator and in the real-world scenario.
- Provided the open-source framework to the research community.

II. RELATIVE WORKS

Many autonomous UAV navigation works have been published in recent years. The navigation framework's workflow starts with the localization module, which estimated the 6-DoF pose from the camera or from LiDAR. Then, the mapping module stitches sensor measurements concerning their associated poses and reconstructs the surroundings. Finally, based on the reconstructed map, the motion planning module generates obstacle-awareness and smooth trajectories. The mapping module in the navigation work bridges the gap between localization and planning modules and provides a human-readable map to the system monitor. This section reviews three kinds of map representation that are widely used in the aerial robotics field.

A. POINT CLOUD MAP

The point cloud is the lowest-level representation of the 3D measurements. The most significant advantage of using this kind of map is its ease of generation. The construction of a point cloud map can be carried out by stitching the point clouds from different measurements. Adding every new measurement into the map will rapidly increase the size of the map. The voxel grid filter [8] keeps only one

point in a size-defined region and filters out all other points, preventing memory over-expansion. The periodical application of such a filter to the whole map can eliminate the redundancy points. However, the point cloud itself is in an unorganized data structure. Accessing or modifying a certain point is time-consuming because these operations must scan the whole map. Since the sensor noises and dynamic objects cannot be modified and updated in real time, this type of map is generally suitable for high-precision sensors in static environment scenarios. In the work of Gao *et al.* [9] and Lin *et al.* [10], UAV obstacle avoidance was achieved using an onboard LiDAR sensor. A KD-tree-based point cloud map [11] was used in their navigation framework.

B. OCCUPANCY VOLUMETRIC MAP

As the most commonly used map type, the volumetric map discretizes the entire mapping space into organized voxels that are a grid of cubic volumes of equal size. The occupancy status and other information are stores in each voxel. The conventional map implementation is based on the fixed-size grid, and the occupancy information only contains two states (occupied or not occupied). Such a map is unresponsive to noise and incurs a high overhead of memory. Elfes [12] introduced the probabilistic sensor measurement model to represent the observed and unknown spaces explicitly. In the work by Nießner *et al.* [13], only the occupied voxels were stored in the memory, and these voxels supported fast access with the help of a hash table. The most notable work on the volumetric occupancy map is that of the Octomap [5] that uses a hierarchical data structure to store occupancy probabilities for voxels. The Octomap is robust with respect to sensor noises and supports the incremental growth of the map size. Many successful works on autonomous navigation and path planning [14]–[16] have been based on this mapping kit.

C. SIGNED DISTANCE FIELDS

Unlike the occupancy map, in signed distance fields (SDFs), the distance value is stored in every cell. This representation is widely used in 3D computer graphics because it can achieve surface reconstruction with sub-voxel accuracy [17], [18]. In robotics, Euclidean signed distance fields (ESDFs) have been becoming increasingly popular recently. The distance in ESDFs represents the Euclidean distance to the nearest occupied voxel. The modern optimization-based path planning algorithms generate the collision-free trajectory by optimizing a collision cost function that can be easily generalized from the distance values. Moreover, it is straightforward to perform a collision check on ESDFs. The ESDFs can be computed from an occupancy map. Many approaches focus on the acceleration of the ESDFs construction process. In the work of Cao *et al.* [19], the generation of the ESDFs is accelerated using GPU parallel computation. Lau *et al.* [20] divided the entire mapping plane with the Voronoi diagram and carried out an incremental update of the map. Another kind of SDFs is the truncated signed distance fields (TSDFs), in which the distance represents the distance from the occupied surface to the center of the sensor along the ray direction. TSDFs can be easily obtained from the depth camera. The ESDFs map can be approximated by the multiple TSDFs captured from different viewpoints. The notable methods following this approach for map generation include Voxblox [21] and Fiesta [22].

III. SYSTEM OVERVIEW AND NOTATION

As shown in Figure 2, there are five reference systems in this multilayer mapping framework. They are: global frame ($\boldsymbol{g}$), local frame ($\boldsymbol{l}$), awareness frame ($\boldsymbol{a}$), body frame ($\boldsymbol{b}$), and sensor frame ($\boldsymbol{s}$). In UAV applications, we use the IMU frame as the body frame and the camera frame as the sensor frame. The UAV is flying in a fixed inertial frame (global frame). The vehicle's pose is represented by the transformation from the body frame to the global frame ${}^{g}_{b}\boldsymbol{T}$, and the installation geometry of the perception sensor can be represented by the transformation from the sensor frame to the body frame ${}^{b}_{s}\boldsymbol{T}$.

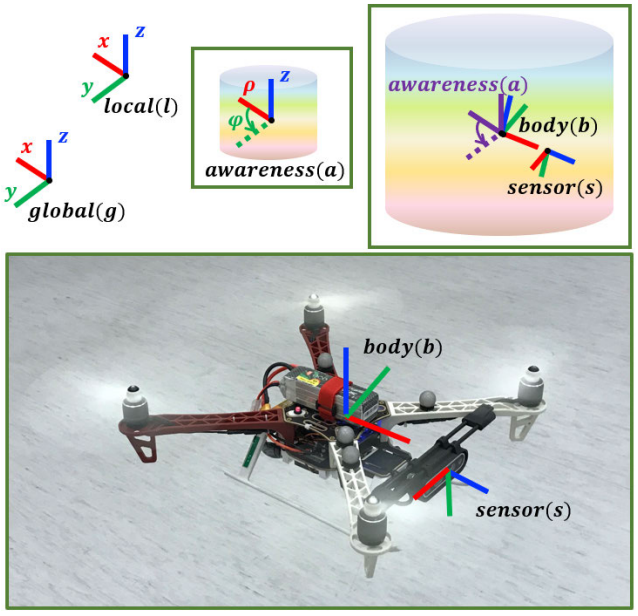


FIGURE 2. Coordinated system of the awareness-local-global mapping framework. In UAV application, the body frame in fixed on the IMU frame and the sensor frame is fixed on the camera frame.

This multilayer mapping framework maintain the global, local, and awareness maps on the global, local, and awareness frames, respectively. The global and local maps are based on the Cartesian coordinate system (x-y-z) while the awareness map is established on the cylindrical coordinate system (ρ-ϕ-z). The global frame is fixed on the map origin. Both local and awareness frame have the same orientation as the global map. The awareness ρ-axis ($\phi = 0$) and local x-axis are parallel to the global x-axis. The awareness z-axis and local z-axis are parallel to the global z-axis.

The awareness frame is attached to the body frame. This means that the translation part of the transformation from awareness map to global map ${}^{g}_{a}\boldsymbol{T}$ is the same as ${}^{g}_{b}\boldsymbol{T}$. The local map represents the active mapping region, and

every map cell in the local map can be directly accessed and updated. The dynamic memory allocation mechanism (see Section IV-C) dynamically adjusts the position of the local frame to keep the body frame inside the central region of the local map.

The input of the mapping framework is the synchronized point cloud ${}^{s}\boldsymbol{PC}$ and pose ${}^{g}_{b}\boldsymbol{T}$. We note that the point cloud is captured in the sensor frame. For every point in the input point cloud (${}^{s}p = (p_x, p_y, p_z)^T \in {}^{s}\boldsymbol{PC}$), its position in the global frame ${}^{g}p$ and awareness frame ${}^{a}p$ can be represented by:

$$ {}^{g}p = {}^{g}_{b}\boldsymbol{T} \cdot {}^{b}_{s}\boldsymbol{T} \cdot {}^{s}p \tag{1} $$

$$ {}^{a}p = {}^{a}_{g}\boldsymbol{T} \cdot {}^{g}_{b}\boldsymbol{T} \cdot {}^{b}_{s}\boldsymbol{T} \cdot {}^{s}p. \tag{2} $$

According to the definitions of the coordinate systems, the ${}^{a}_{g}\boldsymbol{T}$ in Equation 2 can be derived from ${}^{g}_{b}\boldsymbol{T}$ through

$$ {}^{a}_{g}\boldsymbol{T} = {}^{g}_{a}\boldsymbol{T}^{-1} = \begin{bmatrix} 1 & 0 & 0 & {}^{g}_{b}\boldsymbol{T}.tx \\ 0 & 1 & 0 & {}^{g}_{b}\boldsymbol{T}.ty \\ 0 & 0 & 1 & {}^{g}_{b}\boldsymbol{T}.tz \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}, \tag{3} $$

where ${}^{g}_{b}\boldsymbol{T}.tx$, ${}^{g}_{b}\boldsymbol{T}.ty$, and ${}^{g}_{b}\boldsymbol{T}.tz$ refer to the x, y, and z values of the translation part of ${}^{g}_{b}\boldsymbol{T}$. We note that ${}^{a}p$ is in the Cartesian representation form ($p = (p_x, p_y, p_z)^T$), and we need to transform it into the cylindrical representation ($p = (p_\rho, p_\phi, p_z)^T$). The p_z values are the same in both representation forms. The p_ρ and p_ϕ can be calculated by:

$$ p_\rho = \sqrt{p_x^2 + p_y^2} \tag{4} $$

$$ p_\phi = \arctan \frac{p_y}{p_x}. \tag{5} $$

IV. MODELS OF MULTILAYER MAPPING FRAMEWORK

A. RAYCASTING ON CYLINDRICAL COORDINATE

One of the most significant advantages of establishing the awareness map in the cylindrical coordinate system is that it follows the nature of the sensor's perception mode. Compared to the sensor's perception range, the displacement between the body frame and the sensor frame is small and can be neglected. Thus, we can use the center of the cylindrical coordinate system to approximate the sensor center. As shown in Figure 3, if we cast a ray from the sensor center to the observed point, the ray will pass through several voxels. All of these voxels share the same ϕ value, implying that the three-dimensional raycasting can be simplified to the raycasting in a two-dimensional plane ($\rho - z$ plane).

The application of raycasting on a two-dimensional plane can be implemented efficiently with the incremental phase of the traversal algorithm [23]. That is, for a observed point $p = (p_\rho, p_z)$, we incrementally decrease p_ρ and p_z by the step length of a voxel $d\rho$ and dz and set the state of the traversed voxel to non-occupied (Algorithm 1).

B. PROBABILISTIC OCCUPANCY STATE

The probabilistic occupancy state was first introduce by Moravec and Elfes [24]. It achieves the robust occupancy

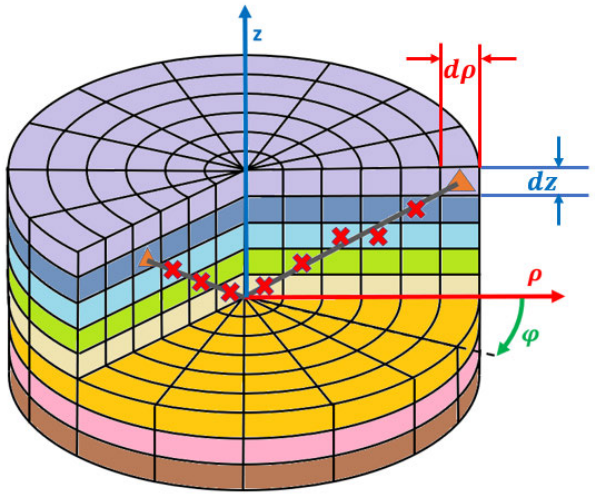


FIGURE 3. Raycasting on the cylindrical coordinates. The gray line is the casting ray, the orange triangle represents the occupied measurement and the red cross represent non-occupied measurement.

Algorithm 1 Fast Raycasting Algorithm

```
foreach p = (p_ρ, p_φ, p_z) ∈ Observation do
    while p_ρ > 0 do
        p_ρ = p_ρ − dρ ;
        p_z = p_z − dz ;
        set voxel (p_ρ, p_φ, p_z) as non-occupied;
    end
end
```

state estimation with the multiple measurements from a noisy sensor. P_n is the probability that a certain voxel n is occupied. P_n ranges from 0 to 1 (non-occupied to occupied). The sensor measurement at time t (z_t) has two statuses (non-occupied and occupied). Given the sensor measurements $z_{1:t}$, the occupancy probability $P_n(z_{1:t})$ of a voxel (with index n) can be calculated according to:

$$ P_n(z_{1:t}) = (1 + \frac{1 - P_n(z_t)}{P_n(z_t)} \cdot \frac{1 - P_n(z_{1:t-1})}{P_n(z_{1:t-1})} \cdot \frac{1 - P_n}{P_n})^{-1}. \tag{6} $$

In Equation 6, P_n refers to the initial probability. It is set to 0.5 in the map initialization because the occupancy state is unknown. $P_n(z_{1:t-1})$ refers to the estimated probability of the previous sample time (integrated from the 1st timestamp), while term $P_n(z_t)$ denotes the probability of voxel n in the measurement z_t. However, the direct storage and updating of the probability is time-consuming because it requires a high amount of floating point operations. The *logit* function, also called the *log-odd* function in Equation 7 that maps the values from (0, 1) to $(-\infty, +\infty)$, is applied to the probability value according to:

$$ logit(P_n) = log(\frac{P_n}{1 - P_n}). \tag{7} $$

Equation 6 can be rewritten as:

$$ logit(P_n(z_{1:t})) = logit(P_n(z_{t-1})) + logit(P_n(z_t)). \tag{8} $$

Comparing Equations 6 and 8, the computationally intensive update process of floating point computation is replaced by the simple addition operation. For a given sensor model, the log-odds values are fixed and do not need to be computed in every update step. Similar to Octomap [5], the probability and log-odds values are the turning parameters for the mapping framework. Experimentally, we found that $P_n(z_t) = 0.4$ and $logit(P_n(z_t)) = -0.4$ are suitable for both the high-precision sensor (such as LiDAR or points clouds in the simulation environment) and the low-precision sensor (such as the depth camera) when the voxel is non-occupied. It is only necessary to tune the probabilities and log-odds values of the occupied measurement (the recommended settings are listed in Table 1). We note that the log-odds value can be mapped into the probability and vice versa, and therefore instead of the occupancy probability, we store this value for each voxel.

TABLE 1. Probability and logit value of occupied/non occupied measurement.

	Occupied	Non occupied
Low precision sensor	$P_n(z_t) = 0.65$	$P_n(z_t) = 0.4$
	$logit(P_n(z_t)) = 0.6$	$logit(P_n(z_t)) = -0.4$
High precision sensor	$P_n(z_t) = 0.7$	$P_n(z_t) = 0.4$
	$logit(P_n(z_t)) = 0.85$	$logit(P_n(z_t)) = -0.4$

A voxel's occupancy status (occupied or non-occupied) is evaluated with a threshold value of the occupancy probability P_{sh}. A voxel is considered to be occupied only when the probability reaches the threshold. The default log-odd value in Equation 7 ranges from $-\infty$ to $+\infty$. However, in practical implementation, this value is limited to a certain range with a saturation module (Equation 9) because the same type of consecutive measurements will push the absolute log-odd value to a large number. Notably, switching the status will become impossible because many new measurements will be necessary to compensate for the previous record, i.e., the system will lack dynamic performance.

$$\begin{aligned} L &= logit(P_n(z_{t-1})) + logit(P_n(z_t)) \\ logit(P_n(z_{1:t})) &= \begin{cases} L_{max} & L > L_{max} \\ L_{min} & L < L_{min} \\ L & L \in [L_{min}, L_{max}] \end{cases} \end{aligned} \tag{9}$$

C. MEMORY MANAGEMENT

The framework adopts the dynamic memory allocation that empowers the system to reconstruct a large mapping with a limited overhead of memory and without the prior knowledge of the size of the mapping area. The basic idea is to allocate a fixed size of memory for every voxel in the activate mapping area (local map) and to only store the occupied voxels for the inactive area.

As shown in Figure 4, the local map is divided into 25 submaps. The red dot represents the position of the robot and is located in the center submap of the local map. When the robot leaves the center submap, it will trigger the map extension operation. That is, the algorithm calculates the distances between the robot position and the center positions of eight neighborhood submaps. The neighborhood submap with the minimum distance indicates the extension direction.

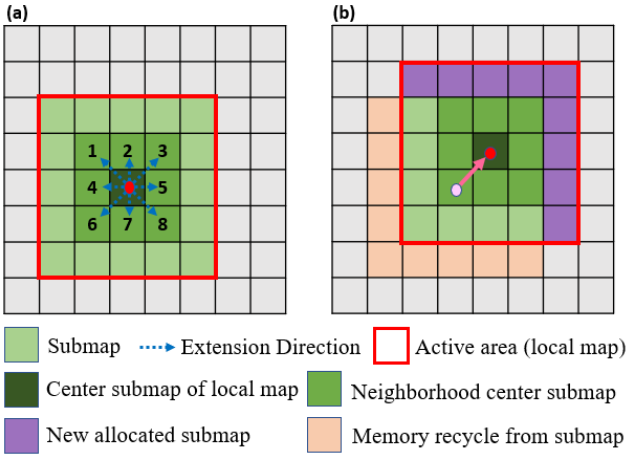


FIGURE 4. Expanding the global map from (a) to (b). Every local map is divided into 25 submaps and can extend in 8 different directions. The dynamic memory management mechanism will allocate new submaps to extend the active mapping area and recycle the memory from the inactive mapping area. Notably, the figure is drawn in 2D but the algorithm is implemented in 3D.

After moving the center submap of the local map to one of the neighborhoods, the framework will allocate the memory to every voxel of the new submaps. The memory of the submaps outside the active mapping area will be recycled, which means that only the information of the occupied voxels are stored. The memory for the non-occupied voxels will be freed.

We note that we explain this mechanism using 2D schematics, while the mechanism is implemented in 3D; in other words, the local map is divided into 125 regions and has 26 potential extension directions.

V. IMPLEMENTATION OF THE MAPPING FRAMEWORK

A. SYSTEM OVERVIEW

Figure 5 shows the mapping framework that consists of the three independent threads: the awareness map thread, the local-global map thread, and the visualization thread. These threads are coordinated by ROS messages and run in parallel. The mapping module's input consists of the estimated pose from the localization module and the point cloud from the depth sensor. The input will be synchronized and sent to the awareness map thread. The awareness map thread conducts the range and visibility checks and passes the update information to the local-global map thread. Then, the local-global thread updates the probability of all of the relevant voxels. The framework also supports the projected 2D map and local ESDF map output. Users can enable or disable these two functions according to the requirements. Furthermore, all maps provide the information to the visualization thread. The visualization thread is independent and can run on the ground station computer.

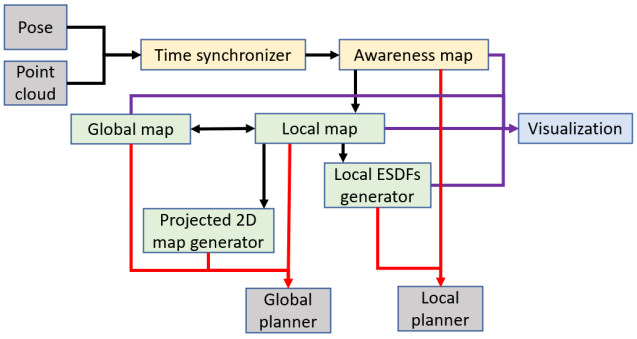


FIGURE 5. System architecture of the mapping framework. The awareness map thread, the local-global map thread, and the visualization thread are colored in yellow, green, and blue, respectively.

B. AWARENESS MAP UPDATE WORKFLOW

The awareness map will be updated upon receiving the synchronized point cloud ${}^{c}\boldsymbol{PC}$ and pose ${}^{w}_{B}\boldsymbol{T}$ information. The update process consists of two steps. The first step is to update the awareness map from the previous awareness map with the newest pose. The transformation between two sample times $\triangle\boldsymbol{T}$ can be calculated by the ${}^{w}_{B}\boldsymbol{T}$ and the previous pose ${}^{w}_{B}\boldsymbol{T}_{last}$ according to Equation 10. After making a screenshot of the original awareness map, we reset the occupancy state of the awareness map. For every occupied cell in the original map, we applied the transformation $\triangle\boldsymbol{T}$ to it, and found its position in the new awareness map. We set the occupancy of the voxel state as occupied.

$$\triangle\boldsymbol{T} = {}^{w}_{B}\boldsymbol{T} \cdot {}^{w}_{B}\boldsymbol{T}^{-1}_{last} \tag{10}$$

Algorithm 2 Awareness map thread workflow

Input: UAV pose ${}^{w}_{B}\boldsymbol{T}$, point cloud ${}^{B}\boldsymbol{PC}$
Output: Occupied/Non-occupied list $L_o[\,]/L_n[\,]$

```
for i ← 0 to SAMPLE_CNT by 1 do
    random pick a point ^B p from ^B PC;
    transfer ^B p to awareness frame ^l p with Equation 2;
    if ^l p inside of awareness map's range then
        PC_sample[ ] ← ^l p;
    end
end
foreach ^l p ∈ PC_sample do
    L_o[ ] ← ^l p;
    set the relevant voxel to occupied;
    apply raycasting (Algorithm 1), add non-occupied
    measurement to L_n[ ] ;
end
return L_o[ ]/L_n[ ] ;
```

The second step is to update the occupancy state with the point cloud and apply the raycasting. The occupied list and non-occupied list will be filled during the second step and then sent to the global map thread. Since the input point cloud may consist of too many points, the first step is to randomly downsample the point cloud and apply the range filter that filters out the points outside the awareness map range. For each point in the downsampled point cloud, we find the located voxel, set it as occupied voxel and add it to the occupied measurement list. Then, we apply the ray-casting from the center of the awareness map to the voxel. The voxels lying on the ray are set to non-occupied and are pushed into the non-occupied measurement list. The workflow of the awareness map update is shown in Algorithm 2.

C. LOCAL-GLOBAL MAP UPDATE WORKFLOW

The local-global map thread received the occupied/non-occupied measurements and the robot pose from the awareness map thread ${}^{w}_{B}\boldsymbol{T}$. The first step is to determine whether it is necessary to update the region of local area. If ${}^{w}_{B}\boldsymbol{T}$ exceeds the center submap of the local map, the algorithm will find its closest neighborhood submaps, and move the center of the local map to this submap. Then, the algorithm will allocate the memory for the new submaps inside the new local map area and recycle the memory from the submaps in the inactive area. The occupied/non-occupied measurements information is then used to update the occupancy probability (represented using log-odds value) on the local map (Equation 8 and Table 1). The local-global map thread contains two independent modules to generate the project 2D map and local ESDFs (Figure 6). The projected 2D map projects all of the occupied voxels to the ground level. Such a map can be used for global planning. The ESDFs generator extracts a batch of voxels from the global map. For every voxel in the batch, we calculate its distance to all other occupied voxels. The lowest values is then selected as the distance. This map can be used for optimization-based trajectory generation.

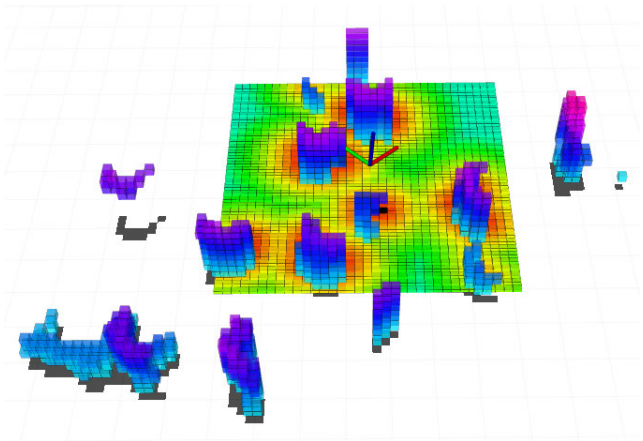


FIGURE 6. Local map, projected 2d grid map, and ESDFs (slide at $z = 0.3$).

VI. EXPERIMENT

A. EXPERIMENT PLATFORM

We first evaluated the mapping framework on the simulation platform (Figure 7a). The simulator is developed based on the

Algorithm 3 Local-global map thread workflow

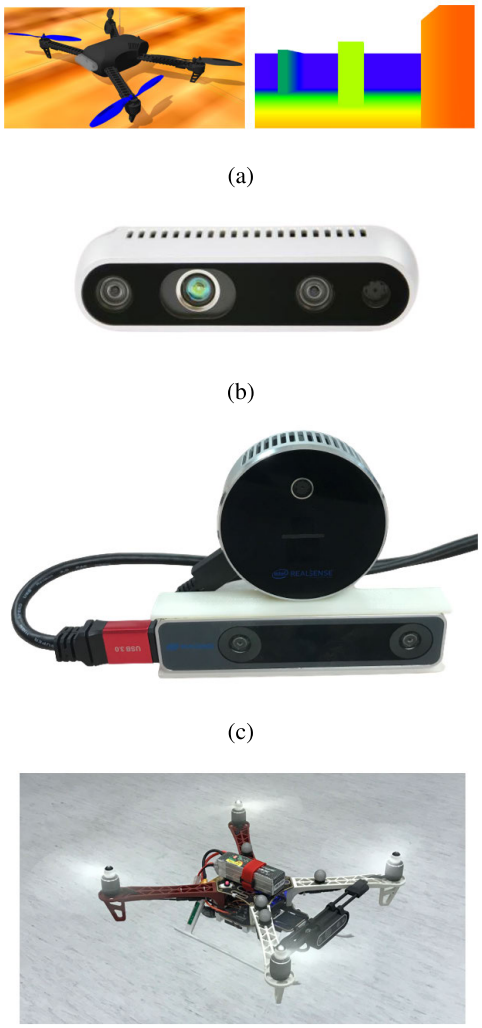
Input: UAV pose ${}^{w}_{B}\boldsymbol{T}$, Occupied/Non-occupied list $L_o[\,]/L_n[\,]$

if ${}^{w}_{B}\boldsymbol{T}$ *exceed the range of center submap* **then**
  switch the localmap to the new region, allocate memory for new submaps, and recycle submap in inactive area (see IV-C).
end
foreach ${}^{l}p \in L_o[\,]$ **do**
  update relevant voxels in local map with occupied measurement.
end
foreach ${}^{l}p \in L_n[\,]$ **do**
  update relevant voxels in local map with non-occupied measurement.
end
if *enable 2d grid map output* **then**
  project the localmap to the 2d plane and generate the 2d grid map.
end
if *enable ESDFs output* **then**
  calculate ESDFs inside the defined batch.
end

ROS-GAZEBO-PX4 toolchain. A depth camera is attached to a UAV. The user can command the UAV to move in the simulation world [25]. We note that we did not add noise in this simulator. This means that the input data (point cloud and UAV pose) for the mapping framework are highly accurate. Then, we evaluated the mapping framework with two sensor suites. Figure 7b shows the D435 camera. The camera can only output the point cloud. The pose information is collected from the motion capture system or using the pose estimator. Figure 7c shows another sensor suite. In this suite, a D515 camera and a T265 sensor are installed on a 3d printing framework. The D515 camera can output the high-quality point cloud, and the T265 sensor can output the real-time pose of the sensor suite. In Figure 7d, a D435 camera is installed on a UAV platform. The UAV is equipped with resource-limited embedded computer and we achieve autonomous navigation in the indoor environment. All of the navigation algorithms, including this mapping kit are executed on the on-board computer.

B. DYNAMIC PERFORMANCE OF THE MAPPING KIT

To test the dynamic performance of the mapping, we placed the D435 depth camera toward a wall, and then a person walked across the sensor. As shown in Figure 8, when the person started to walk into the mapping area, the mapping framework reacted to this event, and people appeared on the map. When the person walked away, the unoccupied measurement from raycasting kept updating the map, and the previously occupied voxels transformed into unoccupied voxels.



(a) (b) (c) (d)

FIGURE 7. Verification of the mapping framework using different platforms. (a) Simulation platform; (b) Realsene d435 depth camera, (c) sensor setup of a Realsense l515 camera and Realsense t265 camera, (d) UAV platform with a mounted Realsene d435 depth camera.

C. MAPPING IN THE INDOOR ENVIRONMENT

We conduct the simulation on the ROS-GAZEBO-PX4 toolchain. A depth camera is attached to a quadcopter. We manually command the quadcopter to perform exploration in the simulation world. The simulation world is a $30\,m \times 20\,m$ room. We add obstacles such as walls, cylinders, boxes, and spheres to the simulation world. We map the environment with the resolutions of 0.1 m cubes. The bird's eye view and side view of the reconstructed map and the ground truth are shown in Figure 9 and a good agreement between them can be observed.

We then conduct the real-world experiment in our lab. We place boxes and artificial obstacles in the Lab. In this experiment, we choose to use the real-sense d435 as the perception sensor. The pose information of the sensor is provided by the motion capture system. We held the sensor and

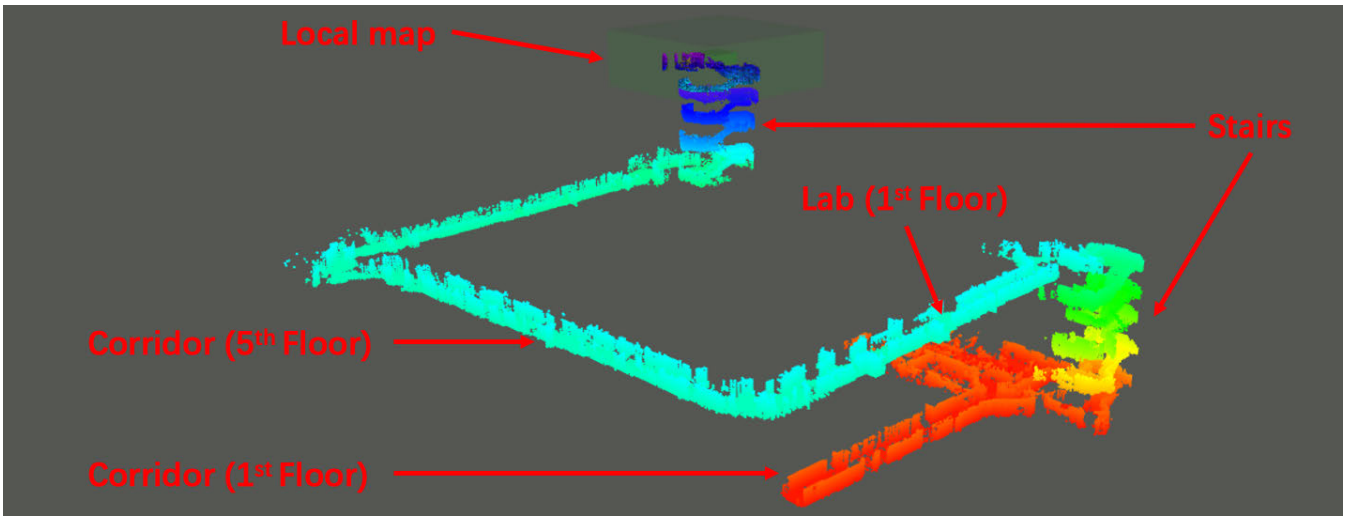


FIGURE 11. Large-scale mapping inside the teaching building. The length of a single corridor is 60 meters and the height of the building is 31 meters. The green region in the map represent the active mapping area (local map).

walk around the test field. The reconstructed map is shown in Figure 10. The voxel size of this map is 0.1 m.

D. LARGE-SCALE MAPPING

We design this experiment to evaluate the performance of this mapping framework on the balance between the memory overhead and the size of the mapping area. In this experiment, we hold the T265+L515 sensor suite (Figure 7c) and walk inside the teaching building. Figure 11 shows the result. The voxel size of this map is 0.15 m. The total length of the traveled path is 246 m. The mapping started from the corridor on the first floor. Then, we mapped the laboratory and went up to the fifth floor. Then, we walk through 3 corridors to another side of the building. Finally, we went up to the top floor of the building. The full video of this experiment can be found in the supplementary materials.

Figure 12 plots the memory usage of the mapping framework in this experiment. We observe that initially, the algorithm allocates 192 MB for every voxel in the awareness map and local map. Then, with the expansion of the mapping area, the memory usage increase incrementally. The total memory consumption is 258 MB. The map is divided into active (local map) and nonactive mapping areas. The dynamic memory allocation mechanism keep recycling the memory from non-occupied voxels in the nonactive mapping area. Therefore, the increasing trend of memory usage is not proportional to the map's size, but rather to the occupied voxels in the map. The system can describe a large mapping scenario with a limited overhead of memory.

E. AUTONOMOUS UAV NAVIGATION WITH THE MAPPING FRAMEWORK

Finally, we integrated this mapping kit into our UAV navigation system. The UAV navigation system's hardware consists of a D435 camera and a LattePande^{TM} 864 embedded computer (Figure 7d). In the software part, the mapping kit works together with our previously developed localization and path-planning kits to achieve the click-and-fly level autonomy in an unknown environment. That is to say, the UAV has no prior knowledge of the environment setup and needs to sense the surrounding and reacts to it accordingly.

TABLE 2. Processing time of mapping framework in different computation platform.

	Computer1	Computer2
Awareness Map Update	11.9 ms	22.1ms
Local-global Map Update	5.2ms	7.7 ms
Projected 2D Map	<1ms	<1ms
Local ESDFs	32.1ms	39.1ms

Note: Computer 1: i5-8250u CPU, 8GB RAM; Computer 2: m3-8100y, 8GB RAM (Latte Panda embedded computer)

The localization kit we used is the FLVIS [26], a stereo visual-inertial SLAM. It adopts the feed-forward and feedback loops to fuse the IMU and camera data and achieves high accuracy pose estimation in the resources limited computation platform. The output from this localization module and the raw point cloud measurement from the D435 camera were feed into the mapping framework as the input.

We used the Fuxi Planner as our path planning kit, which searches and plans the obstacle awareness and trajectory based on the reconstructed map. The planner composes two parallel running planner. The global planner works on the 2D grid map to find the shortest 2D path with the improved Jump Point Search algorithm, and output the local setpoint to the local planner. The local planner works directly on the awareness map to avoid the potential collision and plans a kinematically feasible trajectory.

To verify the capability of the UAV navigation system, we place some artificial obstacles in the test field. We command the UAV to take off and fly to the target destination point behind the obstacle. The estimated pose and the point cloud from D435 are fed into the mapping kit to reconstruct the surroundings. Based on the reconstructed map, the path

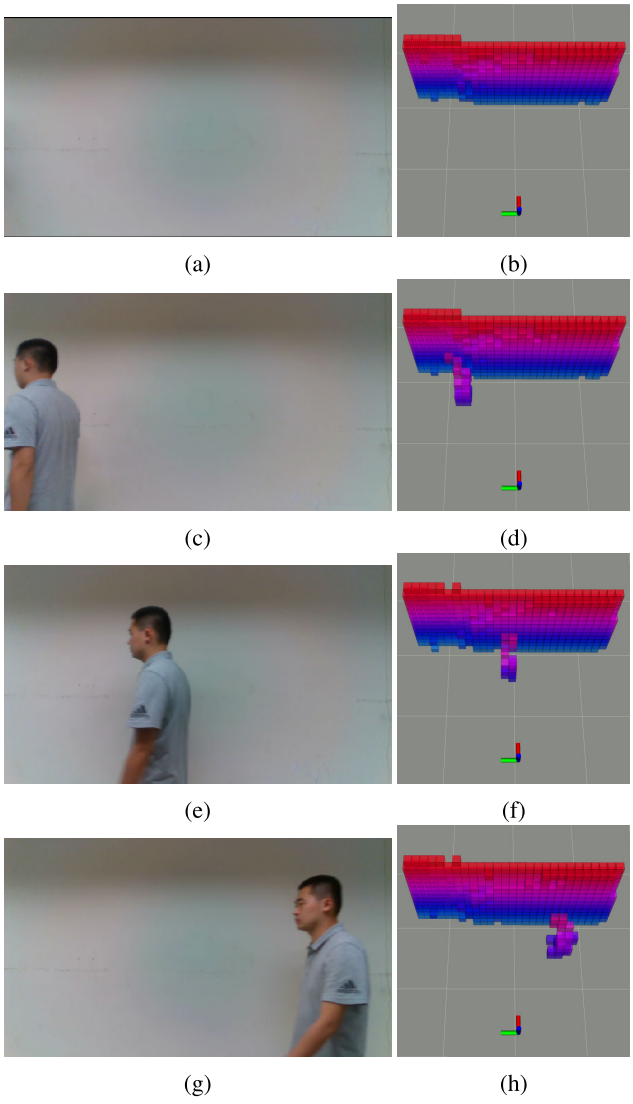


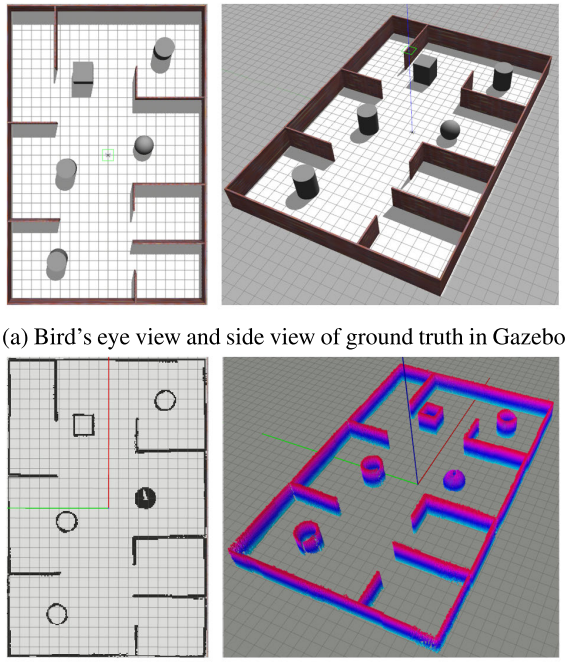
FIGURE 8. A person walked across a wall from right to left. (a)-(c)-(e)-(f) are the camera views, (b)-(d)-(f)-(h) are the reconstructed maps.

planning kit generates the obstacle awareness trajectory to the destination and forwards it to the UAV (Figure 13). In this experiment, the projected 2D grid map, local map, and awareness map are used for path planning. The full video of this experiment can be found in the supplementary materials.

F. PERFORMANCE OF THE MAPPING FRAMEWORK

To evaluate the processing speed of our mapping framework, we test the mapping framework in both a personal computer and resources-limited embedded computer platform using the building dataset (See VI-D). The configuration used for running the dataset is as follows:

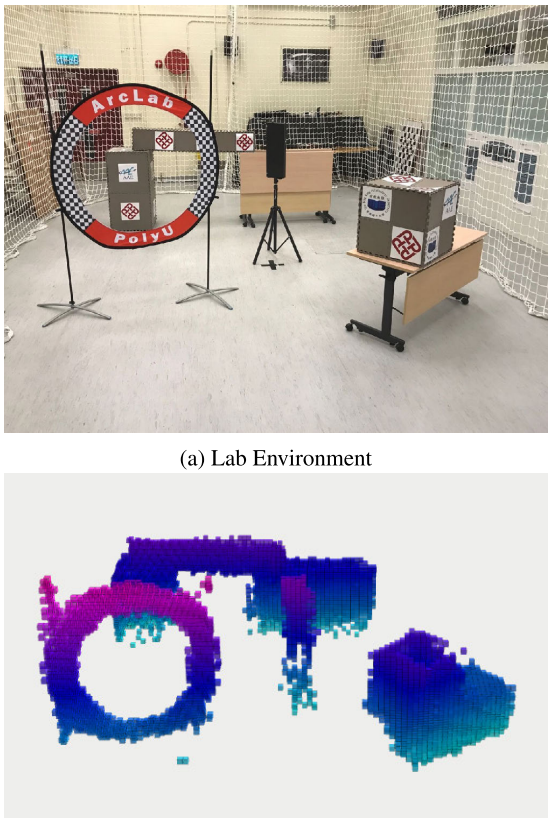
- The awareness map consists of 450000 voxels (100, 180 and 25 discretization steps in ρ, ϕ and z directions, respectively).
- The size of the voxel is 0.2 m and the size of local map is 19 m*19 m*7 m.



(a) Bird's eye view and side view of ground truth in Gazebo

(b) Reconstructed map with the voxel size of 0.1m

FIGURE 9. Ground truth and the reconstruct map.



(a) Lab Environment

(b) Reconstructed map with the voxel size of 0.1m

FIGURE 10. Reconstruct the lab environment.

- All of the features and functions are enabled, including the raycasting, ESDFs generator and 2D grid map generator.

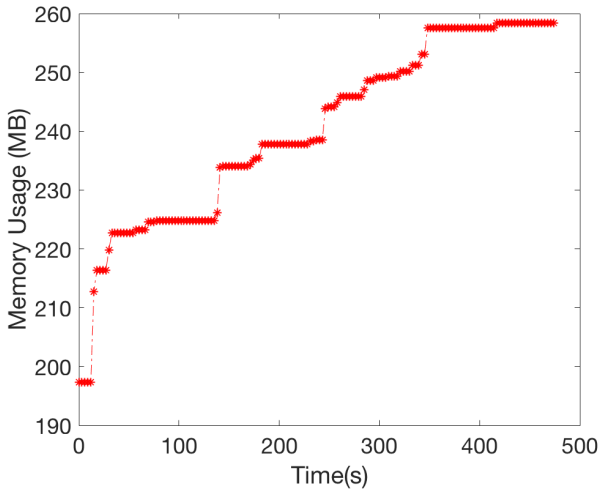
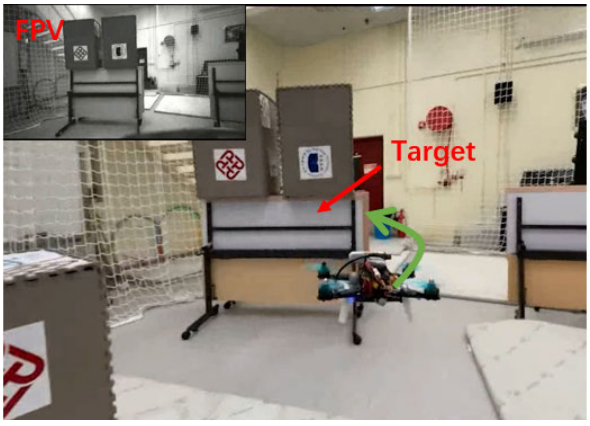
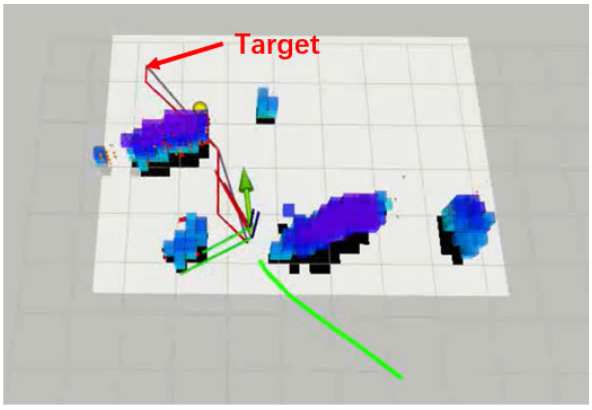


FIGURE 12. Memory usage of the mapping framework in large-scale mapping.



(a) Third-person view and onboard-camera view (upper-left inset) of UAV navigation. The target destination is behind the obstacle.



(b) Multilayer mapping framework in UAV navigation. Green curve is the UAV path, green arrow is the velocity vector, the purple and red lines are the planned paths.

FIGURE 13. UAV navigation in the unknown environment.

Table 2 shows the performance of our mapping kit. The mapping framework can satisfy the real-time requirement for UAV navigation (replan the trajectory 10 times per second) even with the limited computational resources.

TABLE 3. Comparison of features and performance of our mapping kit with Octomap and Voxblox.

	MLMapping	Octomap	Voxblox
Occupancy map update time	17.1 ms	22.1ms	39.2ms
ESDFs support	Yes	No	YES
ESDFs update time	49.2 ms	N.A.	39.2ms
Large scale mapping	Yes	Yes	No
Memory Overhead	259MB	209MB	N.A.

Test conducted on a i5-8250u CPU, 8GB RAM computer. The grid size for all mapping kits is 0.1m. The Voxblox update the ESDFs and Occupancy map in a single thread and the update times are same. As Voxblox does not support large scale mapping, we didn't evaluate it's memory overhead and use the subset of the dataset to test it processing speed.

We also compared the features and performance of our mapping kit with two other widely used mapping kits: Octomap [5] and Voxblox [21]. We list the results in Table 3. The proposed mapping kit support various map representation with relatively high processing speed and affordable memory consumption.

VII. CONCLUSION

In this article, we present a novel open-source three-dimensional mapping framework. Our approach uses a three-layer awareness-local-global map design and achieves a fast raycasting-based visibility check in the local map. By adopting the probabilistic occupancy state, our framework can represent volumetric 3D models robustly. The map visualization part is decoupled from the map maintenance thread and can run on another computer separately. This design can decrease the computational load of the on-board computer. We evaluated our approach with the simulation and the real-world data sets. The results demonstrate that our approach can model the environment in the embedded computation platform in real time. We also integrated our mapping framework into the UAV navigation system and achieved click-and-fly level autonomy in the indoor environment without prior knowledge. The open-source framework is provided to the research community. The verification data sets are also available online for to enable the reproduction and comparison of our experimental results.

ACKNOWLEDGMENT

This work was performed under the project of "Resilient Urban PNT Infrastructure to Support Safety of UAV Remote Sensing in Urban Regions." The authors want to acknowledge Dr. Peng Lu from The University of Hong Kong for his contribution in the path planning kit.

REFERENCES

[1] H. Bavle, P. D. L. Puente, J. P. How, and P. Campoy, "VPS-SLAM: Visual planar semantic SLAM for aerial robotic systems," *IEEE Access*, vol. 8, pp. 60704–60718, 2020.
[2] N. Gageik, P. Benz, and S. Montenegro, "Obstacle detection and collision avoidance for a UAV with complementary low-cost sensors," *IEEE Access*, vol. 3, pp. 599–609, 2015.
[3] D. Meagher, "Geometric modeling using octree encoding," *Comput. Graph. Image Process.*, vol. 19, no. 2, pp. 129–147, Jun. 1982.
[4] P. Payeur, P. Hebert, D. Laurendeau, and C. M. Gosselin, "Probabilistic octree modeling of a 3D dynamic environment," in *Proc. Int. Conf. Robot. Autom.*, vol. 2, Apr. 1997, pp. 1289–1296.

[5] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auto. Robots*, vol. 34, no. 3, pp. 189–206, Apr. 2013.
[6] J. Ryde and J. J. Corso, "Fast voxel maps with counting Bloom filters," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 4413–4418.
[7] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proc. 23rd Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, 1996, pp. 303–312.
[8] X.-F. Han, J. S. Jin, M.-J. Wang, W. Jiang, L. Gao, and L. Xiao, "A review of algorithms for filtering the 3D point cloud," *Signal Process., Image Commun.*, vol. 57, pp. 103–112, Sep. 2017.
[9] F. Gao, W. Wu, W. Gao, and S. Shen, "Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments," *J. Field Robot.*, vol. 36, no. 4, pp. 710–733, Jun. 2019.
[10] Y. Lin, F. Gao, T. Qin, W. Gao, T. Liu, W. Wu, Z. Yang, and S. Shen, "Autonomous aerial navigation using monocular visual-inertial fusion," *J. Field Robot.*, vol. 35, no. 1, pp. 23–51, Jan. 2018.
[11] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.
[12] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, Jun. 1989.
[13] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using voxel hashing," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 1–11, Nov. 2013.
[14] H. Qin, Z. Meng, W. Meng, X. Chen, H. Sun, F. Lin, and M. H. Ang, "Autonomous exploration and mapping system using heterogeneous UAVs and UGVs in GPS-denied environments," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1339–1350, Feb. 2019.
[15] A. Annaiyan, M. A. Olivares-Mendez, and H. Voos, "Real-time graph-based SLAM in unknown environments using a small UAV," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2017, pp. 1118–1123.
[16] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online UAV replanning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 5332–5339.
[17] S. F. F. Gibson, "Using distance maps for accurate surface representation in sampled volumes," in *Proc. IEEE Symp. Volume Visualizat. (VVS)*, Oct. 1998, pp. 23–30.
[18] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones, "Adaptively sampled distance fields: A general representation of shape for computer graphics," in *Proc. 27th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, 2000, pp. 249–254.
[19] T.-T. Cao, K. Tang, A. Mohamed, and T.-S. Tan, "Parallel banding algorithm to compute exact distance transform with the GPU," in *Proc. ACM SIGGRAPH Symp. Interact. 3D Graph. Games*, 2010, pp. 83–90.
[20] B. Lau, C. Sprunk, and W. Burgard, "Improved updating of Euclidean distance maps and Voronoi diagrams," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 281–286.
[21] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D Euclidean signed distance fields for on-board MAV planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 1366–1373.
[22] L. Han, F. Gao, B. Zhou, and S. Shen, "FIESTA: Fast incremental Euclidean distance fields for online motion planning of aerial robots," 2019, *arXiv:1903.02144*. [Online]. Available: http://arxiv.org/abs/1903.02144
[23] J. Amanatides and A. Woo, "A fast voxel traversal algorithm for ray tracing," *Eurographics*, vol. 87, no. 3, pp. 3–10, 1987.
[24] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 2, Mar. 1985, pp. 116–121.
[25] S. Chen, H. Chen, W. Zhou, C.-Y. Wen, and B. Li, "End-to-end UAV simulation for visual SLAM and navigation," 2020, *arXiv:2012.00298*. [Online]. Available: http://arxiv.org/abs/2012.00298
[26] S. Chen, C.-Y. Wen, Y. Zou, and W. Chen, "Stereo visual inertial pose estimation based on feedforward-feedback loops," 2020, *arXiv:2007.02250*. [Online]. Available: http://arxiv.org/abs/2007.02250

SHENGYANG CHEN received the B.Eng. degree from Northwestern Polytechnical University, China, and the M.S. degree from the University of Siegen, Germany. He is currently pursuing the Ph.D. degree with the MAV/UAV Laboratory, Department of Mechanical Engineering, The Hong Kong Polytechnic University.

HAN CHEN (Graduate Student Member, IEEE) received the B.Eng. and M.S. degrees from the Beijing Institute of Technology, China, in 2016 and 2019, respectively. He is currently pursuing the Ph.D. degree with the MAV/UAV Laboratory and the ARC Laboratory, Interdisciplinary Division of Aeronautical and Aviation Engineering, The Hong Kong Polytechnic University.

CHING-WEI CHANG received the B.S. degree in mechanical engineering from Yuan-Ze University, Taoyuan, Taiwan, in 2015. He is currently pursuing the Ph.D. degree with the MAV/UAV Laboratory, Department of Mechanical Engineering, The Hong Kong Polytechnic University. He has worked as an Engineer and an experienced UAV Pilot at Taiwan UAV Company Ltd., from 2016 to 2017. His research interests include the development of UAV path planning, variable pitch mechanism and control, and fault-tolerant control.

CHIH-YUNG WEN received the B.S. degree from the Department of Mechanical Engineering, National Taiwan University, in 1986, and the M.S. and Ph.D. degrees from the Department of Aeronautics, California Institute of Technology (Caltech), USA, in 1989 and 1994, respectively. He joined the Department of Mechanical Engineering, The Hong Kong Polytechnic University, in 2012, as a Professor. In 2019, he became the Interim Head of the Interdisciplinary Division of Aeronautical and Aviation Engineering, The Hong Kong Polytechnic University. His current research interests include modeling and control of tail-sitter UAVs, visual-inertial odometry systems for UAVs, and AI object detection by UAVs.