2015 IEEE 27th International Conference on Tools with Artificial Intelligence

A 3D frontier-based exploration tool for MAVs

Cheng Zhu[1 2], Rong Ding[1 2], Mengxiang Lin[1 3*],Yuanyuan Wu[1 2]
[1]*State Key Laboratory of Software Development Environment*
[2]*School of Computer Science and Technology*
[3]*School of Mechanical Engineering and Automation*
Beihang University, Beijing, China
zcdoyle@hotmail.com, dingr@buaa.edu.cn, linmx@buaa.edu.cn, qiqipchy@buaa.edu.cn

***Abstract*—This paper presents a 3D frontier-based exploration tool named 3D-FBET. Our tool runs onboard the MAV equipped with a 3D sensor. The 3D map of the environment explored is constructed incrementally from two consecutive point clouds obtained. Considering the computation and memory limitations of MAVs, the OctoMap is utilized to represent 3D models. A novel approach is designed to extract the 3D frontiers from the OctoMap. Different from existing extraction method, only state-changed space in the 3D map is processed in each iteration. We implement our approach on top of the well-known robot operating system (ROS) and demonstrate the effectiveness of our tool in real scenarios.**

***Keywords*-Autonomous indoor exploration; 3D frontier-based method; MAVs;**

I. Introduction

The advances in construct and control make micro aerial vehicles (MAVs) an ideal choice for indoor autonomous exploration. Compared to a ground robot, the 6 DOF motions of a MAV enable it to detect the environment in almost any way needed, ensuring the full coverage of detection. Moreover, the movement of MAVs is not restricted by the 2D terrain. Their flying ability allows them explore the entire space more efficiently.

Vision-based autonomous indoor exploration has been well-studied for ground robots. However, as discussed in [1], the extension of traditional 2D strategy to 3D introduces several challenges. The large amounts of map data bring us high computational payload in the process of frontier extraction. In addition, incomplete free space observations result in poor exploration performance.

The main objective of this paper is to develop a tool that is capable of autonomous exploration in GPS-denied indoor environments. Our solution is based on vision and the target flying platforms are equipped with 3D sensors. Due to the limited payload carrying capabilities and on-board computational resources, the tradeoff between detecting capacity and system resource consumption is weighed carefully in design. Specifically, frontier-based exploration strategy is used to search and explore complex unknown space in a 3D world. The map built during exploration is based on 3D grid structure provided by OctoMap. To improve the efficiency of frontier extraction in 3D, a real time frontier detection strategy is used and only state-changed space in the 3D map is detected in each iteration. A tool named 3D-FBET is implemented in the framework of robot operating system (ROS) and is publicly available. The primary contribution of this paper is threefold: (1) to propose a 3D frontier-based indoor exploration approach for MAVs; (2) to implement our approach on an open-source tool; (3) to validate our tool on two experiments.

In the rest of this paper, we discuss the related work in Section 2. The details of our approach are presented in Section 3 and the implementation and experimental results are described in Section 4. Section 5 is a summary of the paper.

* Corresponding author. E-mail addresses: linmx@buaa.edu.cn.

II. Related Work

Recently, different autonomous indoor 3D exploration approaches have been proposed. In some approaches, 2D exploration tools are used to three dimensions. Fraundorfer et al. present a vision-based autonomous mapping and exploration system with a quadrotor MAV [2]. Bachrach et al. present a solution for enabling a quadrotor helicopter to autonomously explore and map unstructured indoor environments [3]. However, both of them use the 2D frontier-based exploration algorithm and set a fixed altitude for the MAV during exploration. The Next-Best-View (NBV) method [4] is applied for autonomous objects searching and exploration in 3D environments [5,6]. Dornhege et al. introduce a novel approach based on the NBV method to determine a minimal sequence of sensor viewpoints and evaluate their system on a mobile platform [5]. Ivan et al. present a 3D exploration strategy for a ground robot equipped with a 3D laser scanner and experiment in large unknown spaces [6]. However, the need for significant computational resources limits the application of the NBV method in MAVs. To address the issue, a stochastic differential equation-based exploration algorithm is proposed to enable the exploration in indoor environments with a payload constrained MAV [7].

Our work in this paper is closely related to the following work. Keidar et al. present two frontier detection algorithms (WFD and FFD) [9] to improve the efficiency of frontier-based approach. An open-source tool based on the WFD al-

1082-3409/15 $31.00 © 2015 IEEE
DOI 10.1109/ICTAI.2015.60

gorithm was developed for 2D exploration [10]. Makarenko et al. advocate an integrated approach to the selection of the frontiers [11] and define the cost function with reference to the theory of information entropy [12]. Pravitra et al. present a compact exploration strategy designed to be implemented onboard indoor MAV equipped with a 2D laser range scanner [13]. The exploration was achieved by augmenting a frontier based guidance strategy with wall-following logic. Different from the research above, the goal of our work is to extend frontier-based approach to three-dimensional cases.

III. Approach

A. Overview

The goal of exploration is to gain as much information as possible about an unknown environment in a reasonable time [8]. According to the frontier-based strategy, the entire space is classified into three states: unknown, free and occupied. The unknown space is a territory that has not yet acquired sensory information. Initially, all regions are unknown. The free space is not occupied by obstacles, which means MAVs can move free in it. The occupied space is the region which contains obstacles. Frontiers are regions on the boundary between the free space and the unknown space. Based on different exploration strategies, a MAV moves from one frontier to another until no frontier exists.

Like many 3D applications, the map is a key component of 3D exploration. This motivates our choice for using OctoMap to model the 3D environment explored. OctoMap is a volumetric representation of space developed for robotic applications. It is not only efficient with respect to memory consumption but also with respect to access times. On the other hand, 3D exploration suffers more from the limitation of the field of view of sensors compared with 2D. Therefore, a real time frontier detection strategy [9] is used in our tool to alleviate the problem. That is, frontiers are detected and evaluated continuously during flight. However, continuous frontier detection demands a high efficiency of frontier extraction. As a result, a state-changed based frontier extract algorithm is developed in this paper.

Following the general framework of frontier-based exploration, 3D-FBET proceeds in three phases from a high level. First, the three-dimensional map is built from point clouds observed by the sensor. Second, the frontiers from the OctoMap are extracted and clustered. Finally, the candidate frontiers are evaluated by a cost function and the goal frontier with the highest cost is selected. The details of each phrase are presented in the following sections.

B. Mapping

3D-FBET takes a frame of 3D point cloud as input, which describes the detected obstacles within the field of view (FOV) of sensors. The aim of mapping is to incrementally build the 3D map by fusing the 3D point clouds generated by the sensor. The SLAM algorithm is used to get a 3D point cloud model from two consecutive frames of point clouds. Specifically, an accurate transform matrix for translation and rotation is generated after a series of refinement and denoising processes such as visual odometry, ICP (Iterative Closest Point) and EKF (Extended Kalman Filter).

Instead of building a 3D occupancy grid map, we utilize the OctoMap as the data structure of 3D map. An OctoMap is a hierarchical data structure for spatial subdivision in 3D [14]. In the OctoMap, a node represents a cubic volume in the 3D environment, which we call a cell and its resolution is changeable. The compressed representation of a 3D map results in savings in storage costs and improved query performance. During the 3D map building process, each cell in the OctoMap is labeled as unknown, free, or occupied according to the state of space it represents.

C. Frontier extraction

Frontier extraction involves how to get a candidate frontier cell from a set of frontier cells. A frontier cell is considered as a free cell that has at least one unknown cell as its neighbor. A candidate frontier cell is the representative of the set of frontier cells.

Unlike other frontier extraction approach [9], we extract frontiers incrementally. When a new frame of point cloud comes, the global map is rebuilt and the states (i.e. unknown, free and occupied) of cells in the global map are updated with the new-coming point cloud. Instead of the entire space, only the state-changed space is checked in frontier cells calculation . The algorithm 1 describes frontier calculation. The set of state-changed cells is obtained directly from the state-check method of octree in octmap. For each cell in the state-changed cells, the free cell with at least one unknown neighbor is found out and pushed into the queue $frontier\_cells$.

Usually, $frontier\_cells$ contains too many cells to be evaluated considering the computational cost. So we gather the continuous cells into several clusters and select their geometric central cell as the representative for this candidate frontier. The algorithm 2 presents details of frontier clustering. Specifically, the depth-first algorithm is used to gather the neighbors of each cell in $frontier\_cells$ to a cluster.

It is worth noting that a queue of the history frontiers is maintained during frontier extraction. The members of the history queue are checked first in each cycle and the cells that are no longer a frontier would be deleted from the history frontier queue.

D. Frontier Evaluation

In the presence of multiple candidate frontier cells, a decision should be made by a criterion that selects the desired waypoint from the set of candidate frontier cells while maximizing (or minimizing) some evaluation function. Several evaluation criteria have been proposed based on different optimization goals. Take the situation shown in Fig.

Algorithm 1 Frontier calculation

Input: m_octree :the current octree
Output: $frontier\_cells$:a vector containing frontier cells

$frontier\_cells = \phi$
$neighbors = \phi$
m_octree.GetChangeCells($statechanged\_cells$)
for all $cell$ such that $cell \in statechanged\_cells$ **do**
  $point =$ GetCellPoint($cell$)
  $key = m\_octree$.CoordToKey($point$)
  if key is occupied **then**
    $neighbors =$GetNeighbor(key)
    for all nei such that $nei \in neighbors$ **do**
      if nei is unknown **then**
        $flag = true$
      end if
    end for
    if $flag$ is $true$ **then**
      add key to $frontier\_cells$
    end if
  end if
end for

Algorithm 2 Frontier clustering

Input: $frontier\_cells$:a vector containing frontier cells
Output: $candidate\_cells$:a vector containing candidate frontier cells

$candidate\_cells = \phi$
$temp\_queue = \phi$
while $frontier\_cells$ is not empty **do**
  $f_cell =$POP($frontier\_cells$)
  add f_cell to $temp\_queue$
  while $temp\_queue$ is not empty **do**
    $temp_cell =$POP($temp\_queue$)
    $neighbors =$GetNeighbor($temp\_cell$)
    for all nei such that $nei \in neighbors$ **do**
      if nei is in $frontier\_cells$ **then**
        add nei to $temp\_queue$ //check the unique when adding to the $temp\_queue$
        cancel nei from $frontier\_cells$
      end if
    end for
    add $temp\_cell$ to $cluster$
  end while
  add $cluster$ to $cluster\_gather$
end while
for all $cluster$ such that $cluster \in cluster\_gather$ **do**
  $centercell =$FindCenter($cluster$)
  add $centercell$ to $candidate\_cells$
end for

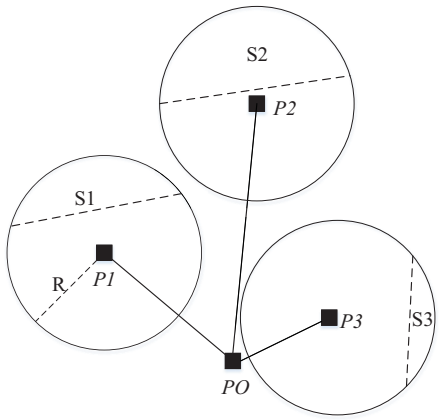

Figure 1. Demonstration of evaluation function.

1 as an example. Suppose that PO is the current location of the MAV, and $P1$, $P2$ and $P3$ are the candidate frontier cells. Let R be the range of the sensor. For a candidate frontier cell Pi $(i = 1, 2, 3)$, Si is the unknown space detected from the current location PO. According to the cost of travel criterion, $P3$ is the desired frontier since it is the closest frontier to PO. In contrast, $P2$ would be selected by the expected information gain criterion.

Our evaluation function takes the two factors into account. As a result, the optimal goal frontier is the cell in which we can gain more new information about the unknown environment, while the cost of moving to it should be as low as possible. For our example, $P1$ is the desired waypoint weighing the benefits of both. The definition of evaluation function is as follows:

$$c_f = w_1 \times \frac{N_{unknown}}{N_{all}} - w_2 \times \parallel P_f - P_O \parallel \qquad (1)$$

where c_f is the cost of the candidate frontier cell, P_f represents the coordinate of the candidate frontier cell and P_O represents the current position, w_1 and w_2 are the relative weight and could be adjusted according to the environment and demands. $N_{unknown}$ and N_{all} are the number of the unknown cells and all cells in the spherical space of around each candidate frontier cell (the radius is R).

In principle, our exploration strategy works in depth-first order. The frontiers in the new-coming state-changed space are evaluated until no new frontiers are extracted. After that, the closest frontier cell in the history queue is selected as the goal frontier and the exploration continues. In fact, the goal frontier would be adjusted in real time during flight. It may disappear when the MAV moves to it. However, it gives the direction where the MAV should flight.

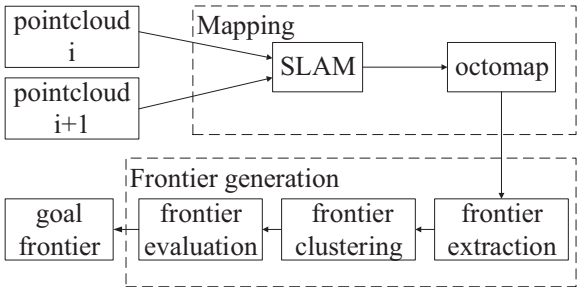

Figure 2. 3D-FBET framework.

IV. Implementation and Experiment

A. Implementation details

The 3D-FBET is implemented based on the well-known robot operating system (ROS), which ensures that our tool can be integrated to many existing systems in ROS, such as 3D SLAM and 3D navigation. The block diagram of the 3D-FBET architecture is shown in Fig. 2. The interaction between modules is done by topic communication method in ROS.

The SLAM module gets current point cloud from the sensor and incrementally building the 3D model. We use the RGBDSLAM in ROS and remap the point cloud topic to the topic requested by the OctoMap module.

The OctoMap module handles the 3D model and builds the OctoMap incrementally. The octomap_server libraries in ROS is used to build the OctoMap data structure of the environment explored. Two functions are developed based on the query function of the octomap_server: (1) to return the neighbors of a cell according to a certain coordinate, including the unknown, free and occupied. (2) to check and return the state-changed cells after each observation.

The frontier generation module extracts the frontier cells from the OctoMap, clusters the continuous cells into a candidate frontier, and selects the candidate frontier with the highest cost as the goal frontier. The goal frontier cell and its three-dimensional coordinate are returned for navigation and planning. We also publish the occupied cells, free cells, frontier cells, goal frontier cell on respective topic in ROS for visualization.

Our tool is publicly available, which can be found in the website of github: https://github.com/zcdoyle/fbet.git. The software is organized according to the ROS build system of stacks and packages.

B. Experimental results

The embedded quad-core CPU EPIA-P910 is applied for our hardware experiment platform. The sensor used is the xtion RGBD 3D sensor from ASUS which provides RGBD data at 30Hz. 3D-FBET builds the OctoMap and generates the goal frontier at 0.5∼10Hz depending on the number of the state-changed cells.

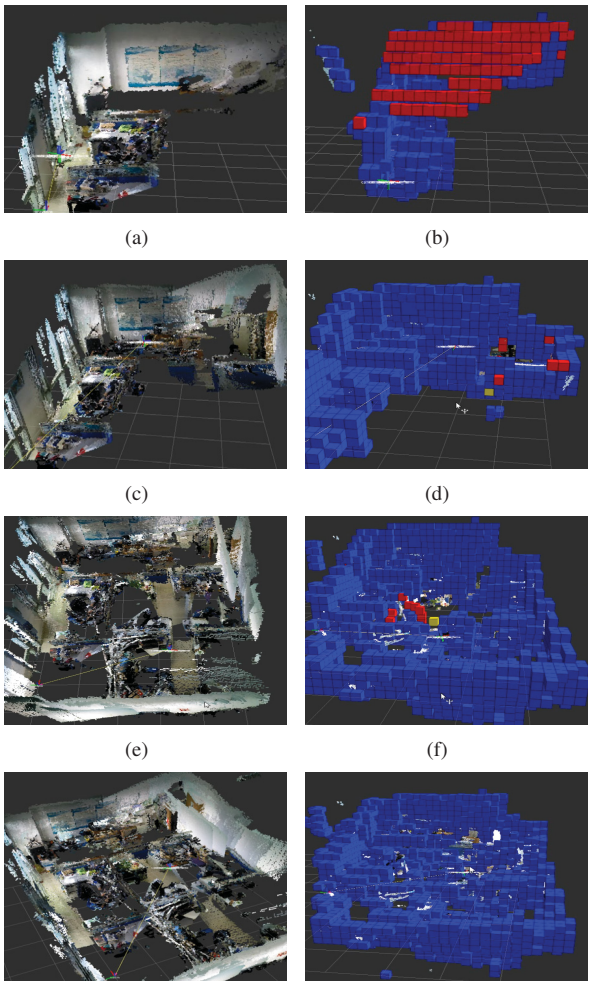

Figure 3. Office exploration process.

We conducted two experiments to demonstrate the performance of the 3D-FBET in the 3D indoor environments: (1) a 3D exploration in a cluttered office of $7m \times 6m$; and (2) a 3D exploration in an unstructured meeting room of $12.6m \times 7.8m$. The intermediate results of the exploration process are shown in Fig.3 and Fig.4, among which figure (a), (c), (e), (g) are the 3D model built by the SLAM algorithm, and figure (b), (d), (f), (h) are the OctoMap based on the 3D model. The blue cells, red cells and yellow cells represent the occupied cells, the frontier cells and the final goal frontier cell, respectively. As we can see, drift occurs near the right wall in Fig.3(g) generated by RGBDSLAM in ROS. Some optimization strategies are expected to solve the issue in our future work.

The total exploration time in each experiment is less than 3 minutes and 5 minutes respectively. The computation time of mapping and frontier generation is examined carefully. The average processing time of building the OctoMap from

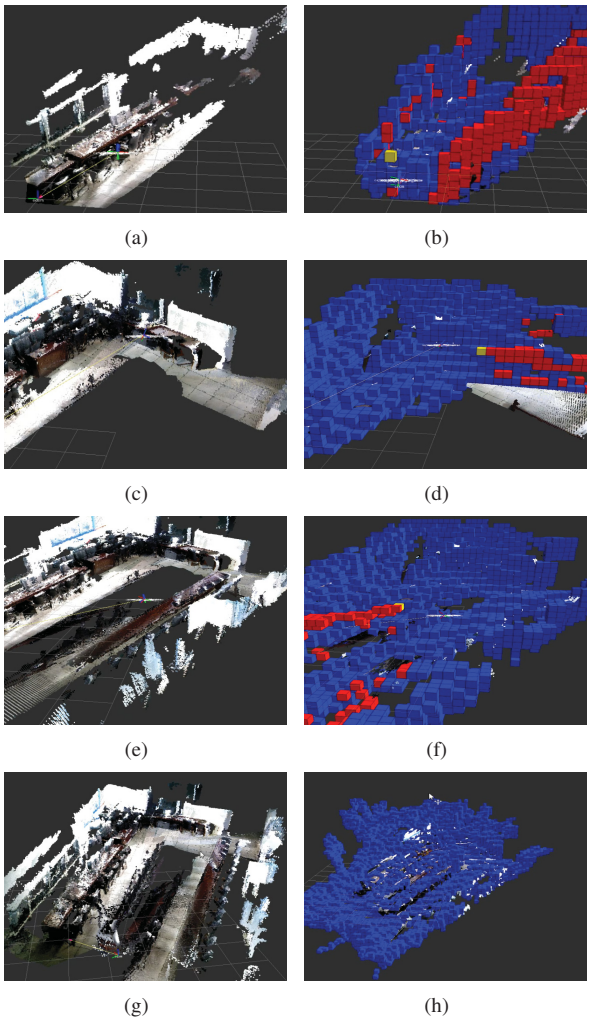

Figure 4. Meeting room exploration process.

Table I
THE NUMBER OF FRONTIER CELLS AND CORRESPONDING COMPUTATION TIME.

n_{fron}	$time(sec)$	n_{fron}	$time(sec)$
6	0.000198522	78	0.00624603
11	0.000479162	86	0.0107626
22	0.000711354	96	0.0106371
35	0.00101569	117	0.0146942
45	0.00207981	133	0.016315
57	0.00671504	169	0.0194044
68	0.00245926	215	0.0155885

a frame of point cloud is about 0.5s. The average time of frontier clustering and evaluation is in the level of millisecond. The time spent on frontier extraction varies by the number of frontier cells. Table 1 shows the typical number of the extracted frontier cells and the corresponding computation time in our experiments.

V. CONCLUSION

An efficient autonomous exploration tool is essential for a variety of indoor applications of MAVs. As one of our efforts in the area, we extend the traditional frontier-based exploration to 3D and implement our approach in the framework of ROS in this paper. Preliminary experiments in real-world scenarios demonstrate the effectiveness and efficiency of our tool. Introduction of 3D autonomous navigation and more experiments on actual flight are left as future work.

REFERENCES

[1] Shen S, Michael N, Kumar V. Autonomous indoor 3D exploration with a micro-aerial vehicle[C]// Robotics and Automation (ICRA), 2012 IEEE International Conference on. IEEE, 2012:9 - 15.

[2] Fraundorfer F, Heng L, Honegger D, et al. Vision-Based Autonomous Mapping and Exploration Using a Quadrotor MAV[C]// Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on. IEEE, 2012:4557 - 4564.

[3] Bachrach A, He R, Roy N. Autonomous Flight in Unknown Indoor Environments[J]. International Journal of Micro Air Vehicles, 2009, 1(4):217-228.

[4] Banta J E, Zhien Y, Wang X Z, et al. A Best-Next-View Algorithm for Three-Dimensional Scene Reconstruction Using Range Images[C]// In Intel. Robotics and Comp. Vision XIV session of Intel. Sys. and Advamced Manufacturing Symp. SPIE. 1995:418 - 29.

[5] Dornhege C, Kleiner A. A frontier-void-based approach for autonomous exploration in 3d [C]// Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on. IEEE, 2011:351 - 356.

[6] Maurovic, Dakulovic I, Petrovic M, et al. Autonomous Exploration of Large Unknown Indoor Environments for Dense 3D Model Building[J]. World Congress of the International Federation of Automatic Controlcape Town South Africa, 2014, 19(1):10188 - 10193.

[7] Shen S, Michael N, Kumar V. Autonomous multi-floor indoor navigation with a computationally constrained MAV[C]// Proceedings - IEEE International Conference on Robotics and Automation. 2011:20 - 25.

[8] B. Yamauchi. A frontier-based approach for autonomous exploration[C]// In Proc. CIRA. IEEE Computer Society-Washington, DC, USA, 1997:146 - 151.

[9] Keidar M, Sadeh-Or E, Kaminka G A. Fast Frontier Detection for Robot Exploration[J]. International Journal of Robotics Research, 2012.

[10] Paul Bovbel. Implementation of frontier exploration for ROS. https://github.com/paulbovbel/frontier_exploration, 2013.

[11] Makarenko A, Williams S B, Bourgault F, et al. An experiment in integrated exploration[C]// Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on. IEEE, 2002:534 - 539.

[12] Cover T M, Thomas J A. Elements of Information Theory. New York:Wiley[J]. Cram101 Textbook Outlines to Accompany, 1991.

[13] Pravitra C, Chowdhary G, Johnson E. A compact exploration strategy for indoor flight vehicles[C]// Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on. IEEE, 2011:3572 - 3577.

[14] Hornung A, Kai M W, Bennewitz M, et al. OctoMap: an efficient probabilistic 3D mapping framework based on octrees[J]. Autonomous Robots, 2013, 34(3):189 - 206.