2016 IEEE International Symposium on
Safety, Security, and Rescue Robotics (SSRR)
EPFL, Lausanne, Switzerland, October 23-27, 2016

Towards autonomous 3D exploration using surface frontiers

P.G.C.N. Senarathne
ST Engineering - NTU Corporate Lab
Nanyang Technological University
Singapore
Email: senarathne@ntu.edu.sg

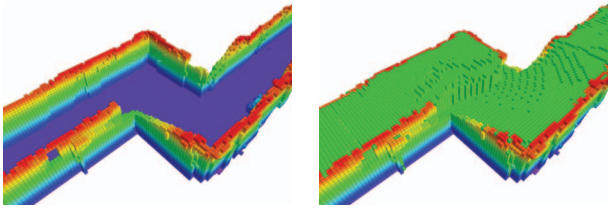
Danwei Wang
School of Electrical and Electronic Engineering
Nanyang Technological University
Singapore
Email: edwwang@ntu.edu.sg

***Abstract*—Exploring environments to generate 3D maps is a core component of any autonomous robot expected to operate in search and rescue or security related missions. Considering that information about surface voxels in a 3D grid map is sufficient for many tasks, following a mapping mission, such as motion planning for searching or rescue, this paper presents an exploration strategy guided by surface frontiers rather than free-space frontiers to actively map the object surfaces in the environment. Surface frontiers in a 3D grid map are surface boundary voxels neighboring unmapped space. Detecting surface frontiers from a 3D occupancy grid is discussed in detail and a strategy to sample view configurations using the detected surface frontiers for exploration is provided. Exploration experiments conducted in both simulated and real indoor environments validate that surface frontiers are a valid and an efficient alternative for free-space frontiers for autonomous 3D exploration.**

I. Introduction

Exploring an unknown environment to generate a map is one of the fundamental tasks of an autonomous robot operating in search and rescue and security missions. These generated maps serve as ground truths for human mission planners, rescuers, security personnel and even for other autonomous robots to execute more complex missions. Autonomous robot exploration to generate 2D maps has been studied extensively in the research literature and frontier based exploration [1] has been one of the most widely used yet simple strategies when implementing 2D exploration algorithms. While 2D frontier based exploration strategies can be employed to generate/recover floor plans as 2D maps, generating complete 3D maps is more desirable and advantageous especially when robots are deployed as initial responders for search and rescue or security missions [2].

With the availability of sensors capable of 3D environment perception and 3D occupancy mapping algorithms [3], the concept of frontiers has been successfully extended to 3D environments [4]–[6]. By default, a frontier voxel in 3D is defined as a free voxel that is neighboring unmapped voxels. However the requirement to process free voxels that span the entire empty space of an environment, as visualized in Fig. 1, to detect *free-space frontiers* could be inefficient if the environment being explored has considerably large free space. In addition, free-space 3D frontier based exploration algorithms, similar to their 2D counterparts, operate by guiding the robot to successively reduce the unmapped voxels and in doing so eventually uncovers the topography of the environment albeit indirectly. While this approach works in general when the combination of the environment and sensor acts in harmony to reveal objects at each successive scan for example in narrow indoor environments, environments that have more open spaces relative to sensor's field of view (FOV) results in this strategy guiding robots to initially map more open spaces rather than more informative object surfaces thus requiring additional controls to be enforced for the robot's view selection strategy [7], [8]. Further this *free-space* frontier based approach introduces more challenges to its localization and mapping as the sensor must perceive surface/obstacle features consistently to generate accurate estimates [9].



(a) (b)

Fig. 1. Visualization of a typical 3D occupancy voxel map. (a) surface voxels only (b) surface + free space voxels. All the free voxels representing the empty space (shown in green color) is processed to extract free-space frontiers, while only the surface voxels need to be processed to extract surface frontiers.

Based on the above mentioned insights the research presented in this article takes an alternative approach to 3D exploration by actively trying to seek the expansion of mapped surfaces rather than trying to indirectly map the topography of the environment through reduction of unmapped voxels. This approach is also justified by considering the fact that almost all the state of the art 3D robot motion planning algorithms rely on obstacle (i.e. mapped surface) information to generate paths that are collision free rather than on free space information [10]. It can also be trivially understood that actively expanding

978-1-5090-4349-1/16/$31.00 ©2016 IEEE

mapped surfaces ultimately results in mapping all the visible surfaces in the environment.

The proposed approach processes only the surface voxels of the 3D occupancy grid map to detect surface frontier voxels. Surface frontier voxels are the boundary voxels of mapped surfaces that are neighboring unmapped space. The detected surface frontier voxels are extracted as contours, and are sampled at a given interval along extracted contours to generate a representative set of the detected surface frontiers. The extracted surface frontier representatives are used to generate a set of valid view configurations for a robot operating in 2D plane with a limited sensor FOV and the best view according to a given utility criteria is selected as the next view of the robot. Experiments are conducted in two simulated environments to analyze the efficiency of surface frontier generation algorithm and the exploration algorithm based on it and in a real world environment to validate the concept of surface frontier based exploration. Details on surface frontier detection is given in Sec. III and next best view generation for exploration is provided in Sec. IV. Details on the experiments, results and analysis are given in Sec. V and the article concludes in Sec. VI.

II. Literature Review

The simplest approach for autonomous 3D mapping is to employ 2D frontier based exploration with 3D mapping [11]. This however eventually leads to unmapped volumes and can later be filled by a dedicated 3D exploration strategy as proposed in [12]. However this two phase strategy requires robot having to make two passes in the environment which is inefficient and is not suitable for time critical missions.

The concept of frontiers has been successfully extended to 3D to generate free-space frontiers for exploring 3D environments [4]–[6] and search and rescue missions [2]. Free-space frontiers are free voxels in the 3D grid map that are adjacent to unmapped voxels. Given the increased computational effort in detecting free-space frontiers and generating view poses based on these frontiers, several alternative approaches for 3D exploration are also proposed. [13] uses virtual particles to represent the free space and applies stochastic disturbances to reveal the free-space boundary. A receding horizon next-best-view strategy where view pose sequences are sampled using RRT is presented in [14]. In [15] a potential based approach is employed for 3D exploration without explicitly generating view configurations. The method proposed in this paper attempts to reduce the computational effort by substituting free-space frontiers with surface frontiers.

Frontier patches defined based on mapped surface information and views sampled by ray-casting back to reachable space [16] is the closest approach to our proposed method albeit used with a different 3D mapping technique. The surface normal based void patches presented in [17] inspired the use of surface projection vectors in the proposed method.

III. Surface Frontier Detection

This section describes the process of extracting surface frontiers from a 3D occupancy grid map and generating the surface frontier representatives. A surface frontier voxel in a 3D occupancy grid map is defined as a boundary voxel of a mapped surface where at least one of the six faces of the voxel is exposed to the unmapped space. Therefore a boundary voxel is a voxel on the geometric boundary of a mapped surface. For single voxel thin surfaces, boundary voxels can be detected easily by adapting the method proposed in [18]. In this method, given the set of surface voxels $\mathcal{V}_s$, any voxel $v_q \in \mathcal{V}_s$ is classified as a boundary voxel if the largest angle generated by any two ordered neighboring voxels v_m, v_n with v_q exceeds a given threshold where v_m, v_n are neighboring voxels of v_q projected onto the plane defined by the surface normal $\boldsymbol{n_q}$ of v_q. However sensor noise, localization uncertainty and discretization of the space results in mapped surfaces to be several voxels thick as shown in Fig. 1(a) and results in this method not being able to detect all the valid surface boundary voxels. The method proposed in this section alleviates this issue by first detecting all 3D edges in the voxel map and then processing them to extract the boundary voxels and in turn surface frontier voxels. Following the convention used in 2D frontiers [1] and free space frontiers [5], surface frontier voxels are grouped before used to generate view configurations. This is done by extracting surface frontier voxels into contours and sampling voxels at some distance interval along the contours to generate surface frontier representatives. Fig. 2 indicates the major steps in the proposed surface frontier detection method and Fig. 3 illustrates the corresponding outputs of these steps. The following subsections describe these operations in more detail.

A. Edge detection in voxel map

The input to the algorithm is the 3D occupancy grid map. First the surface normals are computed by considering the center point of each voxel as a point on the surface. These computed surface normals are utilized through out the rest of the operations. All the boundary edges of single voxel thin surfaces can be detected using the method adapted from [18]. Any query voxel that fails this initial test is subjected to a secondary test based on its surface normal and the surface normals of its neighborhood surface patch. The set of voxels belonging to the neighborhood surface patch of a voxel v_q is denoted by $\mathcal{N}_{q,\theta_T}$ and is defined as follows.

$$\mathcal{N}_{q,\theta_T} = \{v_i \in \mathcal{N}_{q,26} \text{ s.t.} ||cos^{-1}(\frac{\boldsymbol{n_q}.\boldsymbol{\delta_i}}{|\boldsymbol{n_q}||\boldsymbol{\delta_i}|}) - \pi| - \frac{\pi}{2}| \leq \theta_T\}$$

where, $\mathcal{N}_{q,26}$ is the set of 26 neighboring voxels of v_q and $\boldsymbol{\delta_i} = \boldsymbol{v_i} - \boldsymbol{v_q}$. θ_T defines the maximum absolute angle a voxel in $\mathcal{N}_{q,\theta_T}$ makes with the plane defined by $\boldsymbol{n_q}$. A voxel v_q is detected to be a surface edge voxel if its surface normal $\boldsymbol{n_q}$ on average has a large deviation to neighboring voxels's surface normals and this is tested as follows.

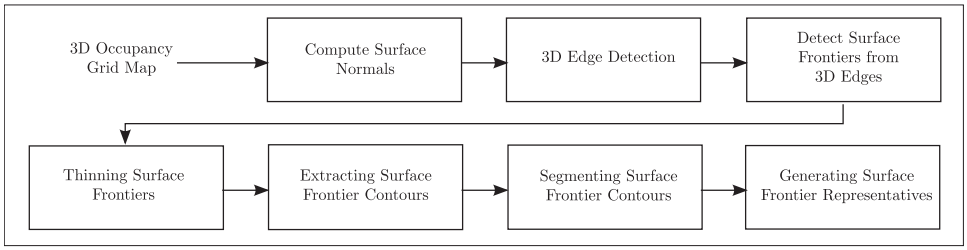


Fig. 2. Process of extracting surface frontier representatives from a 3D occupancy grid map

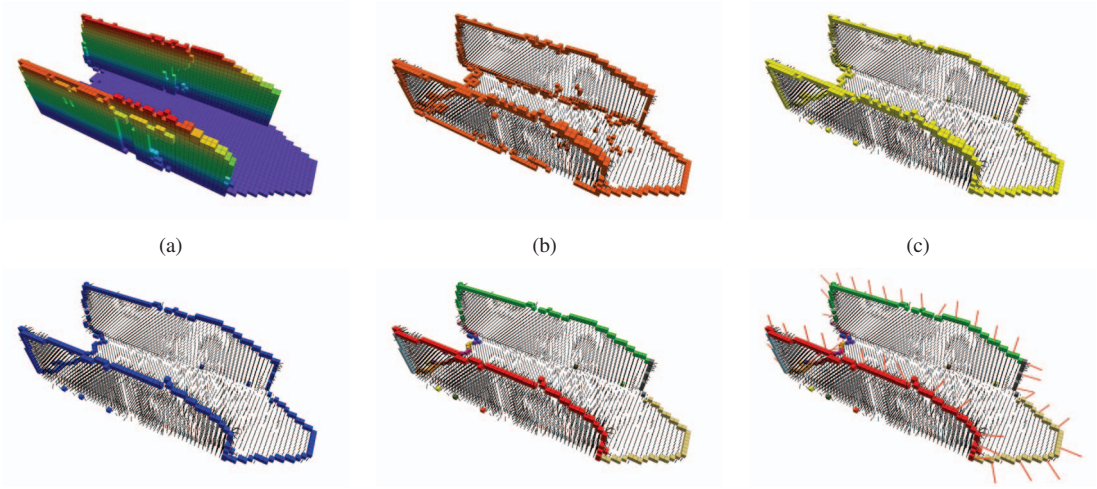


Fig. 3. Visualization of the outputs at different stages of the surface frontier extraction process. (a) The input 3D occupancy grid map generated in a typical indoor hallway. (b) The computed surface normals and detected 3D edge voxels. (c) Detected surface frontier voxels. (d) Thinned surface frontier voxels. (e) Surface frontier contours segmented at corners. (f) Surface frontier representatives sampled from the contours. The red arrows indicate the direction to which surface is projected to expand

$$\sum_{v_i \in \mathcal{N}_{q,\theta_T}} |\boldsymbol{n}_i \times \boldsymbol{n}_q| > |\mathcal{N}_{q,\theta_T}|\kappa$$

where $\boldsymbol{n}_i$ is the surface normal of v_i and κ is the threshold parameter. An alternative but similar approach to 3D edge detection is to detect edges based on the change of curvature at each surface voxel.

Fig. 3(b) shows detected 3D edges together with the computed surface normals from an input 3D occupancy grid map shown in Fig. 3(a). It can be noted that due to irregularities in mapping, some spurious 3D edge voxels are present in the output and requires an additional step of filtering.

B. Filtering invalid boundary voxels

Once 3D edges are extracted, these are tested to check for their validity to be surface frontiers. This process also requires filtering out of spurious 3D edge voxels. This is done by first checking if a detected edge voxel has all its six faces being adjacent to mapped voxels (either free or surface). Any voxel whose all six faces adjacent to mapped voxels are ignored. Remaining edge voxels are checked if they are part of a surface boundary or not. This is done for each v_q by tracing voxels from v_q along the mapped surface in the direction of two initial vectors $\boldsymbol{u_1^0}, \boldsymbol{u_2^0}$ that are opposite to each other and orthogonal to the detected edge and $\boldsymbol{n_q}$. At each step i of the surface voxel tracing operation, $\boldsymbol{u_1^{i-1}}$ and $\boldsymbol{u_2^{i-1}}$ vectors are projected to the tangent surfaces at the traced surface voxels to generate $\boldsymbol{u_1^i}$ and $\boldsymbol{u_2^i}$ respectively. These two vectors are then used to trace the next two voxels on the surface. The voxel tracing is terminated if both $\boldsymbol{u_1^i}$ and $\boldsymbol{u_2^i}$ are tracing voxels approximately in the same direction indicating v_q is at a surface boundary. If voxels can be traced along the surface for $d_{thickness}$ distance in both directions without termination, v_q is classified as not to be in a surface boundary and are filtered out. $d_{thickness}$ is a parameter that depends on the thickness of mapped surfaces and can be set according researcher's preference and experience. We have used a value of $0.3m$ during the experiments for detection of surfaces boundaries from $0.3m$ thick surfaces. The remaining edge voxels after executing these tests are output as the detected surface frontier voxels. Fig. 3(c) illustrates the voxels detected as surface frontiers from the input set of 3D edge voxels.

C. Thinning surface frontiers

While contour extraction can be performed on the extracted frontier voxels, thinning the boundaries using morphological operations allows for extraction of single voxel thick contours

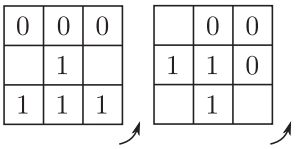


Fig. 4. Structuring elements used for thinning. Arrows indicate applying successive 90° rotations. 1s correspond to frontier voxels and 0s to non-frontier voxels. The $1, 0$ pattern in each structuring element or its three other rotated versions must match the 2D projected grid's $1, 0$ pattern, ignoring values in unmarked cells.

more accurately. Even though 3D structuring elements can be used for thinning the contours [19], similar results can be obtained at an improved efficiency using 2D skeletonization operations by considering that surface frontier contours are locally 2D elements. In order to perform 2D skeletonization operations on the detected frontier voxels, each detected frontier voxel's neighborhood is first projected onto a local 3×3 2D grid patch defined by each voxel's surface normal. The size of a cell in this 2D grid patch is identical to the size of a voxel. Each cell is marked with 1 if a frontier voxel is projected on to it and 0 otherwise. Thinning of the detected frontier voxel contours is done by removing any frontier voxel whose associated local 2D grid patch matches either of the structural elements depicted in Fig. 4 or their 6 other rotated copies [20]. For a structuring element to match a given 2D grid patch, the $1, 0$ pattern in it must match the $1, 0$ pattern in the 2D grid patch, while ignoring the values corresponding to unmarked cells. The remaining frontier voxels at the end of this operation can be chained to generate single voxel thick (i.e. thin) 8-connected contours. Fig. 3(d) shows the remaining frontier voxels after thinning is performed on the detected frontier voxels from Fig. 3(c).

D. Extracting surface frontier contours

Once the thinning operation is complete, a set of single voxel thick contours can be generated by a simple contour tracking algorithm. However, due to the imperfect nature of map data, a single geometric surface boundary may get represented using several voxel contours. To alleviate such occurrences of multiple small contours representing a single surface boundary, the extracted voxel contours are merged based on a nearest neighbor criteria of contour corners.

E. Segmenting surface frontier contours

Even though the extracted contours can be used directly to sample a set of representatives, segmenting these contours based on their curvatures allows to isolate different surface boundaries from each other. For example surface frontiers of a ground surface and a wall attached to the ground surface can be represented using a single contour. However, segmenting it to two contours at the base of the wall connecting to the ground surface allows to treat the ground surface's frontiers separately from wall's frontiers and vice-versa. To perform this segmentation first the local curvature of the contour at each surface voxel is computed. The curvature is approximated using the angle between two vectors defined using two neighboring discrete contour elements of some length [21]. Non-maximum suppression is performed on these curvature values to detect only the valid corners. Spurious corners detected due to sudden voxel changes are filtered out by checking the angle between the two line segments starting at the two neighboring candidate corner voxels and merging at the corner voxel in question.

F. Generating surface frontier representatives

Each segmented surface frontier contour is sampled at regular intervals to extract surface frontier representatives. The interval length is a mission dependent parameter and can be set considering the type of environment, sensors and map resolution used. We've used $0.5m$. Each representative voxel contains a vector approximating the gradient of the surface frontier contour. This gradient and surface normal at the representative voxel is used to compute another vector $\boldsymbol{s_f}$ for each representative f that corresponds to the direction to which the surface is projected to expand. This *Surface-Projection* vector $\boldsymbol{s_f}$ is orthogonal to both surface frontier gradient and the surface normal at the representative voxel f. Surface-Projection vectors are utilized in the view sampling phase to generate properly oriented sensing views for the robot considering its limited sensor FOV as described in Sec. IV-A.

IV. Next best view generation

Once a set of surface frontier representatives are generated from the 3D occupancy grid map, valid view configurations for the robot must be generated and the best view configuration based on a utility criteria is selected as the next best view of the robot. The experiments conducted has restricted the robot's motion to 2D surfaces for ease of development of navigation algorithms, however the presented view generation strategy can be adapted to robots with full 3D motion with minor changes.

A. Sampling a view configuration

Alg. 1 summarizes the steps for view configuration generation. The input to the algorithm is the set of surface frontier representatives $\mathcal{F}$ sorted in ascending order of euclidean distance from the robot's sensor's current position. Since exhaustive search for views in the free configuration space (C_{free}) is intractable [16], a maximum of N_q number of view configurations are sampled and evaluated. To sample a view configuration associated with an $f \in \mathcal{F}$, first a candidate sample point w is generated in the workspace. For a robot operating in a 2D plane, this is done by first projecting f to the 2D plane at the sensor's height and generating a vector on this plane from the projected point with a length l that makes an angle θ with the vector $-\boldsymbol{s_{f,xy}}$. $-\boldsymbol{s_{f,xy}}$ is the 2D projection of negative of $\boldsymbol{s_f}$ associated with frontier representative f. l is selected randomly between minimum and maximum sensor range and θ is also selected randomly between $(-\theta_b, \theta_b)$ where $0 < \theta_b \leq \pi/2$. The end point of the vector is output as w. If w is very close to an obstacle (line. 9) or if f is not

visible from w given sensor's FOV specification (line. 13) it is ignored. A maximum of N_w workspace samples are generated per f and the number of visibility tests per f are limited to $N_{w\_test} < N_w$ to reduce the use of computationally expensive ray casting operations. If w is valid, the union of f and its neighboring representatives that are not occluded by objects when viewed from w, denoted by $\mathcal{N}_{f,w,visible}$, are used to generate the view configuration $q = (w, \boldsymbol{o_{w,xy}})$ where $\boldsymbol{o_{w,xy}}$, the yaw vector, is generated as follows.

$$\boldsymbol{o_{w,xy}} = \frac{\sum_{f' \in \mathcal{N}_{f,w,visible}} \omega_{f'} \boldsymbol{v_{wf',xy}}}{\sum_{f' \in \mathcal{N}_{f,w,visible}} \omega_{f'}}$$

$$\omega_{f'} = e^{(\frac{-\,\mathrm{eucl}(f,f')^2}{2\sigma^2})}$$

$\boldsymbol{v_{wf',xy}}$ is the 2D projection of the vector from w to frontier f' and $\omega_{f'}$ is the weight associated with it. $\mathrm{eucl}(f, f')$ is the euclidean distance between frontier representative f and $f' \in \mathcal{N}_{f,w,visible}$. σ can be varied depending on how much influence the neighboring representatives are expected to have on the yaw vector. If q is not in C_{free} this candidate view configuration is ignored. Large values for N_q, N_w and N_{w_test} lead to more complete exploration at the expense of execution time/computational power and vice-versa. Values of $50, 100$ and 10 respectively provided acceptable results during our experiments.

Algorithm 1 Sample view configurations

Require: $\mathcal{F}$: sorted in ascending euclidean distance to robot
Require: *SensorSpec* : Sensor's FOV specification
1: $Q = \phi$
2: $c_q = 0$
3: **while** c_q == N_q **do**
4: get next $f \in \mathcal{F}$
5: $c_w = 0$, $c_{w\_test} = 0$
6: **while** $c_w < N_w$ **and** $c_{w\_test} < N_{w\_test}$ **do**
7: sample a point w in workspace using f and *SensorSpec*
8: $c_w = c_w + 1$
9: **if** w is very close to an obstacle **then**
10: go to line. 6
11: **end if**
12: $c_{w\_test} = c_{w\_test} + 1$
13: **if** f is not visible from w **then**
14: go to line. 6
15: **end if**
16: extract $\mathcal{N}_{f,w,visible}$
17: generate view q using w and $\mathcal{N}_{f,w,visible}$
18: **if** q is in C_{free} **then**
19: $Q = Q \cup \{q\}$
20: $c_q = c_q + 1$
21: go to line. 3
22: **end if**
23: **end while**
24: **end while**
25: **return** Q

B. Computing the utility of view pose

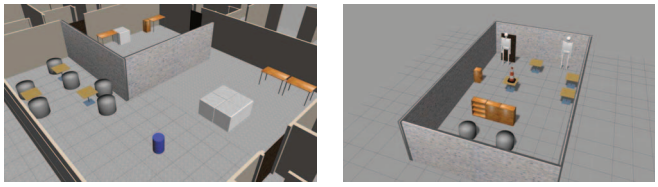
The utility $U(q)$ of a sampled view configuration q is computed using $U(q) = \gamma I(q) - (1-\gamma)C(q)$ [4] with $\gamma = 0.35$ where $I(q)$ is the normalized information gain at q and $C(q)$ is the normalized cost of reaching q from robot's current pose. Since the proposed method guides exploration using surface frontiers, the normalized information gain at q is computed as the normalized summation of visible surface frontier lengths as given in Eq. 1.

$$I(q) = \frac{\sum_{f \in \mathcal{F}_{q,visible}} length(f)}{\max_q I(q)} \quad (1)$$

$\mathcal{F}_{q,visible}$ is the set of all the non-occluded $f \in \mathcal{F}$ that are visible from q with respect to sensor's FOV and $length(f)$ is the length of the surface frontier segment represented by f. The cost of q is set to the normalized planned shortest distance to q from robot's current pose.

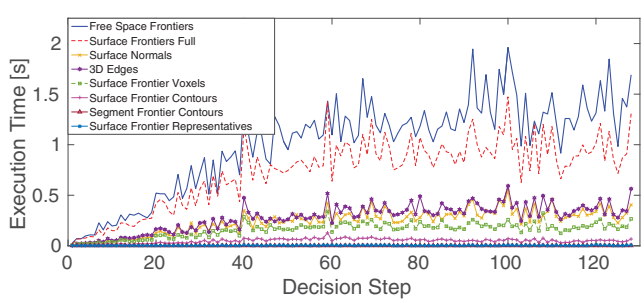
V. Experimental Results

Experiments are conducted in two simulated environments (Fig. 5) to analyze a) the efficiency of the proposed surface frontier generation algorithm and b) the efficiency of the exploration algorithm based on it. A real world experiment is also conducted to validate the proposed exploration algorithm. For ease of developing the navigation algorithm using existing software libraries from Robot Operating System (ROS) [22], the robot is restricted to traverse on a 2D plane for all experiments. However the proposed view planning algorithm can be easily adapted for full 3D navigation and will be the subject of future work. Simulations are conducted in Gazebo [23] with a simulated Pioneer3-AT robot and a Husky A200 robot is used for the real world experiment. For both simulated and real settings, 3D range measurements are generated using an RGB-D sensor with maximum range of 3.5m and horizontal and vertical FOVs of $58°$ and $45°$ respectively. For all experiments the pose estimates are generated using a popular 2D SLAM solution, GMapping [24], using a 2D LiDAR and the 3D map is generated and maintained using OctoMap [3] library. The resolution of the 3D map is set to $0.1m$ per voxel.

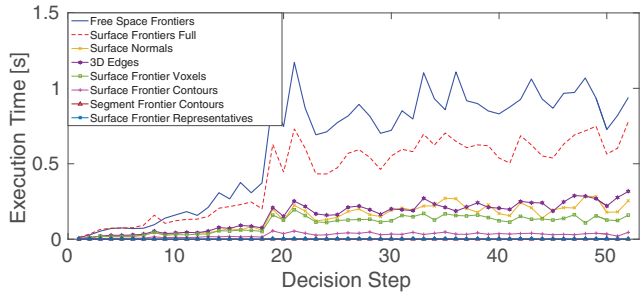


(a) Office-space environment ($16m \times 15m$) (b) Workshop environment ($12m \times 8m$)

Fig. 5. The simulated environments used for experiments



(a) Office-space environment



(b) Workshop environment

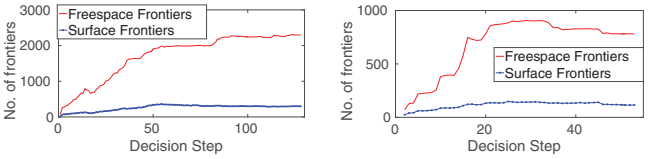
Fig. 6. Comparison of execution times

A. Computational efficiency

The surface frontier detection algorithm is compared with a free-space frontier detection algorithm for efficiency. Free-space frontiers are generated by processing the free voxels and clustering the detected frontier voxels as in [5] into patches with radius of $0.25m$. In order to compare the execution times between these two algorithm in identical conditions, two exploration data sets, one for each simulated environment, are used. [1] For each data set, exploration decision steps are indexed and the two frontier generation algorithms are executed at these decision steps and the elapsed times are recorded. Fig. 6 compares the execution times of the two algorithms along the decision steps for the two simulated environments.

It is evident from both graphs that the surface frontier detection requires less execution time and the difference in execution time with free-space method increases as the exploration mission progresses. This is due to the addition of more and more free voxels to the map which in turn increases the execution time of free-space frontier generation. The graphs also plot the execution times of different stages of the surface frontier detection algorithm. It can be observed that surface frontier contour extraction related stages demonstrate almost constant execution times while the execution times of initial operations executed to detect the 3D surface frontier voxels increase over time with the addition of new surface voxels. This also confirms that the proposed method's execution time is proportional to the number of mapped surface voxels.

[1] Any 3D exploration algorithm can be used to generate the data set as the objective here is to input the same 3D map to frontier generation algorithms.



(a) (b)

Fig. 7. Comparison of number of generated frontiers

	Avg. Time [s]		Avg. Distance [m]	
	SF	FF	SF	FF
Office Space	1204.4	2881.0	131.34	193.07
Workshop Room	470.6	721.6	49.28	54.99

TABLE I
COMPARISON OF EFFICIENCY BETWEEN SURFACE FRONTIER AND FREESPACE FRONTIER BASED EXPLORATION. (SF = SURFACE FRONTIER, FF = FREESPACE FRONTIER)

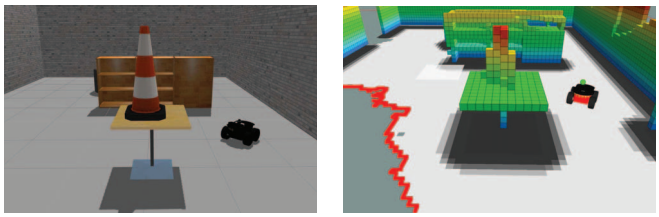
Fig. 7 compares the number of frontiers generated by both methods along the decision steps and they indicate that the proposed method generates significantly smaller number of frontiers as expected making surface frontier based view configuration generation more efficient.

B. Exploration efficiency

The efficiency of the proposed surface frontier based exploration algorithm is compared with a free-space frontier based exploration algorithm. The free-space frontier based exploration algorithm samples view configurations using a similar approach to the proposed algorithm and computes the information gain using sparse ray-casting. For each simulated environment five missions are conducted using each exploration algorithm. Tab. I compares the average time consumed in seconds and the distance traveled in meters by the robot to complete the exploration missions. It indicates that the proposed surface frontier based exploration algorithm is more efficient in exploring the two environments and confirms the utility of the surface frontier based exploration as a valid alternative 3D exploration algorithm.

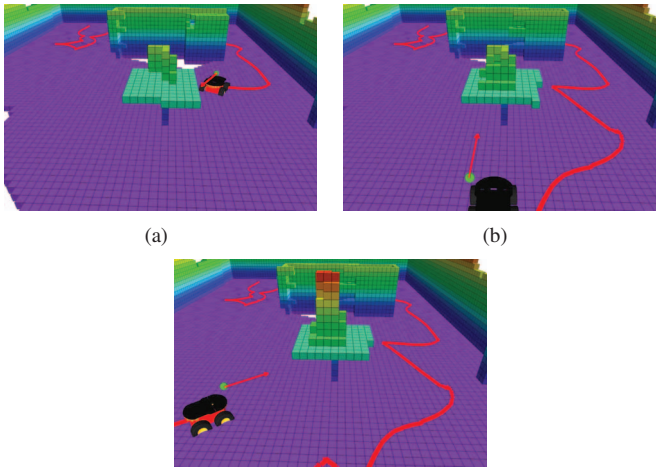
It is important to note our observation that Octomap's 3D mapping implementation by default maps free space voxels only along the rays from the sensor that are induced by the occupied voxels. Therefore it is unable to maintain free-space information in certain open areas in the environment, relative to sensor's maximum range, even though the sensor has not perceived any objects within its FOV. This results in generating incorrect free-space frontiers thus degrading the exploration performance of free-space frontier based method. This indicates that while free-space frontier based exploration is efficient when used in environment + sensor combinations that support accurate maintenance of free-space, it is inferior when used in open environments or with sensors that have shorter range and in conjunction with octree based 3D maps, further justifying the utility of surface-frontiers to guide 3D exploration missions.

Fig. 8(a) shows an interesting feature in the workshop environment that also helps to qualitatively illustrate the efficacy



(a) (b)

Fig. 8. (a) A table and a construction cone on top of it in the workshop environment. (b) 3D map of the scene together with its 2D projection. At robot's current pose the 2D frontiers (marked in red) already extend beyond the table area avoiding the mapping of the cone from its other side.



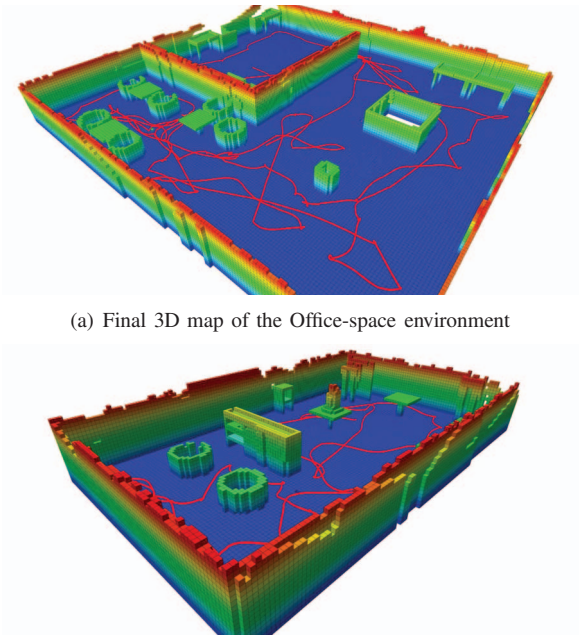
(a) (b) (c)

Fig. 9. Illustrating surface frontier based exploration in action to actively map an item on a table top from multiple view configurations. The red arrow indicates the direction to which the range sensor is aligned.

of surface frontier based exploration. If a 2D frontier based exploration algorithm with 3D mapping is employed to explore the workshop environment using the 2D projected map, the robot would not be able to completely map the construction-cone placed on top of the table. This is due to the generated 2D frontiers extending beyond the table area as indicated in Fig. 8(b), making it difficult to properly orient the sensor to map the other side of the cone, confirming observations in [12]. Whereas, the proposed approach which is based on surface frontiers guides the robot to map the cone in multiple angles until it is fully mapped as illustrated in Fig. 9. The final 3D maps of the simulated environments generated using the surface frontier based exploration algorithm are depicted in Fig. 10.

C. Real world validation

The ability to use surface frontiers to guide autonomous exploration missions is also validated in a real world experiment. Fig. 11 depicts the environment and the robot used for the experiment. Exploration is bounded to a $30m \times 10m$ area. The same localization and mapping tools used in the simulated experiments with identical parameters are used for the real test as well. Fig. 12 illustrates two instances of the



(a) Final 3D map of the Office-space environment

(b) Final 3D map of the Workshop environment

Fig. 10. The resultant 3D maps after exploring using surface frontiers. The robot's trajectory is marked in red.

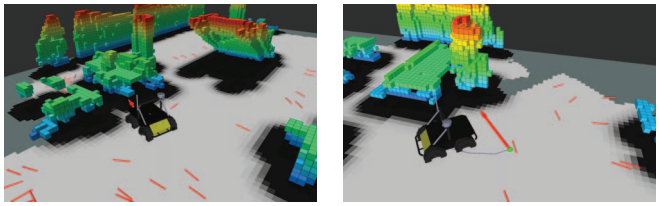
exploration process where the robot is oriented correctly to map object surfaces rather than simply mapping the free space. The small red arrows indicate the view configurations sampled from the free configuration space and the large red arrows indicate the selected next best view configurations for the robot. The proposed exploration algorithm was able to explore the given indoor environment to generate an informative 3D map as depicted in Fig. 13 and validates the utility of surface frontiers for 3D environment exploration.

(a) (b)

Fig. 11. (a) Environment and (b) Husky A200 robot used for experiments.

VI. Conclusion

This article presented a 3D environment exploration strategy based on the concept of surface frontiers. Surface frontier voxels are defined as the boundary voxels of mapped surfaces adjacent to unmapped space. Details on extracting surface frontier voxels from 3D occupancy grid maps with multiple voxel thick surfaces is provided and it was shown that generating surface frontiers is computationally less expensive than generating free-space based frontiers. Results from exploration



(a) (b)

Fig. 12. The selected next best views (large red arrows) orient the robot's sensor to map object surfaces more. The small red arrows are the sampled view configurations. All view configuration samples are generated in the free space (depicted in white color) identified by the collision map projected to 2D plane.

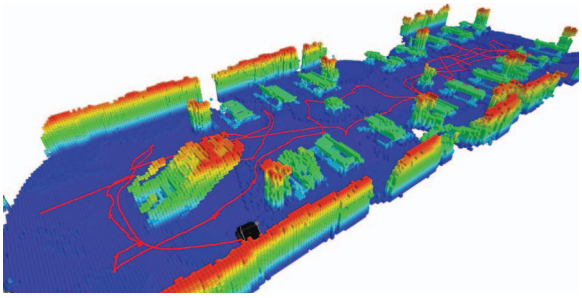


Fig. 13. The 3D map of the real environment generated after exploration. Robot's trajectory depicted by the red line.

missions conducted in both simulated and real environments, where the next best view for the robot is selected based on the view samples generated using surface frontiers, validate the utility of surface frontiers as a viable alternative to free-space frontiers to guide exploration missions.

Even though the exploration results are reported only for robots navigating in 2D planes, the proposed view generation strategy using surface frontiers can be adapted to robots navigating in 3D environments and will be the subject of future work. In addition utilizing surface projection vector information to generate more accurate information gain estimates warrants further research and will also be explored in the future.

ACKNOWLEDGMENT

This research is partially supported by the ST Engineering - NTU Corporate Lab through the NRF corporate lab@university scheme.

REFERENCES

[1] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings., 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Jul. 1997, pp. 146–151.

[2] C. Dornhege and A. Kleiner, "A frontier-void-based approach for autonomous exploration in 3D," *Advanced Robotics*, vol. 27, no. 6, pp. 459–468, 2013.

[3] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[4] C. Zhu, R. Ding, M. Lin, and Y. Wu, "A 3D Frontier-Based Exploration Tool for MAVs," in *Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on*. IEEE, 2015, pp. 348–352.

[5] C. Potthast and G. S. Sukhatme, "A probabilistic framework for next best view estimation in a cluttered environment," *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 148–164, 2014.

[6] I. Maurović, I. Petrović *et al.*, "Autonomous Exploration of Large Unknown Indoor Environments for Dense 3D Model Building," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 10 188–10 193, 2014.

[7] M. Nieuwenhuisen, D. Schulz, and S. Behnke, "Exploration Strategies for Building Compact Maps in Unbounded Environments," in *Intelligent Robotics and Applications*, ser. Lecture Notes in Computer Science, S. Jeschke, H. Liu, and D. Schilberg, Eds. Springer Berlin / Heidelberg, 2011, vol. 7101, pp. 33–43.

[8] P. G. C. N. Senarathne, D. Wang, and H. Wang, "A new gain function for compact exploration," in *Control Automation Robotics Vision (ICARCV), 2012 12th International Conference on*, Dec 2012, pp. 650–655.

[9] R. Grabowski, P. Khosla, and H. Choset, "Autonomous exploration via regions of interest," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 2. IEEE, 2003, pp. 1691–1696.

[10] H. M. Choset, *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[11] M. Al-khawaldah and A. Nüchter, "Multi-robot exploration and mapping with a rotating 3D scanner," *IFAC Proceedings Volumes*, vol. 45, no. 22, pp. 313–318, 2012.

[12] P. S. Blaer and P. K. Allen, "Data acquisition and view planning for 3-D modeling tasks," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 417–422.

[13] S. Shen, N. Michael, and V. Kumar, "Autonomous indoor 3d exploration with a micro-aerial vehicle," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 9–15.

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

[15] R. Shade and P. Newman, "Choosing where to go: Complete 3D exploration with stereo," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2806–2811.

[16] D. Joho, C. Stachniss, P. Pfaff, and W. Burgard, "Autonomous exploration for 3D map learning," in *Autonome Mobile Systeme 2007*. Springer, 2007, pp. 22–28.

[17] R. Pito, "A solution to the next best view problem for automated surface acquisition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 10, pp. 1016–1030, 1999.

[18] R. B. Rusu, "Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments," Ph.D. dissertation, Computer Science department, Technische Universitaet Muenchen, Germany, Oct. 2009.

[19] P. P. Jonker, "Morphological operations on 3D and 4D images: From shape primitive detection to skeletonization," in *International Conference on Discrete Geometry for Computer Imagery*. Springer, 2000, pp. 371–391.

[20] R. C. Gonzalez and R. E. Woods, *Digital image processing*. Upper Saddle River, N.J.: Prentice Hall, 2008.

[21] L. d. F. D. Costa and R. M. Cesar, Jr., *Shape Analysis and Classification: Theory and Practice*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 2000.

[22] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[23] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3. IEEE, pp. 2149–2154.

[24] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *Robotics, IEEE Transactions on*, vol. 23, no. 1, pp. 34–46, 2007.