A Frontier-Void-Based Approach for Autonomous Exploration in 3D

Article *in* Advanced Robotics · November 2011

DOI: 10.1109/SSRR.2011.6106778

2 authors:

Alexander Kleiner
iRobot Corporation
102 PUBLICATIONS **1,416** CITATIONS
SEE PROFILE

Christian Dornhege
University of Freiburg
49 PUBLICATIONS **700** CITATIONS
SEE PROFILE

A Frontier-Void-Based Approach for Autonomous Exploration in 3D

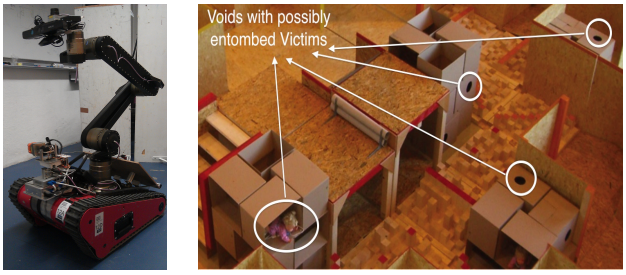
Christian Dornhege and Alexander Kleiner
Institut für Informatik
University of Freiburg
79110 Freiburg, Germany
{*dornhege, kleiner*}*@informatik.uni-freiburg.de*

***Abstract*— We consider the problem of an autonomous robot searching for objects in unknown 3d space. Similar to the well known frontier-based exploration in 2d, the problem is to determine a minimal sequence of sensor viewpoints until the entire search space has been explored. We introduce a novel approach that combines the two concepts of voids, which are unexplored volumes in 3d, and frontiers, which are regions on the boundary between voids and explored space. Our approach has been evaluated on a mobile platform equipped with a manipulator searching for victims in a simulated USAR setup. First results indicate the real-world capability and search efficiency of the proposed method.**

I. INTRODUCTION

We consider the problem of an autonomous robot searching for objects in unknown 3d space. Autonomous search is a fundamental problem in robotics that has many application areas ranging from household robots searching for objects in the house up to search and rescue robots localizing victims in debris after an earthquake. Particularly in urban search and rescue (USAR), victims can be entombed within complex and heavily confined 3d structures. State-of-the-art test methods for autonomous rescue robots, such as those proposed by NIST [Jacoff *et al.*, 2003], are simulating such situations by artificially generated rough terrain and victims hidden in crates only accessible through confined openings (see Figure 1 (b)). In order to clear the so called *red arena*, void spaces have to be explored in order to localize any entombed victim within the least amount of time.

The central question in autonomous exploration is „given what you know about the world, where should you move to gain as much new information as possible?" [Yamauchi, 1997]. The key idea behind the well known frontier-based exploration in 2d is to gain the most new information by moving to the boundary between open space and uncharted territory, denoted as frontiers. We extend this idea by a novel approach that combines frontiers in 3d with the concept of voids. Voids are unexplored volumes in 3d that are automatically generated after successively registering observations from a 3d sensor and extracting all areas that are occluded or enclosed by obstacles. Extracted voids are combined with nearby frontiers, e.g., possible openings, in order to determine adequate sensor viewpoints for observing the interior. By intersecting all generated viewpoint vectors, locations with high visibility, i.e., locations from which many of the void spaces are simultaneously observable, are determined. According to their computed visibility score, these locations are then visited sequentially by the sensor until the entire space has been explored. Note that by combining void spaces with frontier cells the space of valid sensor configurations for observing the scene gets significantly reduced.



(a) (b)

Fig. 1. The long-term vision behind frontier-void-based exploration: Efficient search for entombed victims in confined structures by mobile autonomous robots.

For efficient computations in 3d *octomaps* are utilized, which tessellate 3d space into equally sized cubes that are stored in a hierarchical 3d grid structure [Wurm *et al.*, 2010]. By exploiting the hierarchical representation, efficient ray tracing operations in 3d and neighbor queries are possible.

We show experimentally the performance of our method on a mobile robot platform equipped with a manipulator searching for victims in a simulated USAR setup (see Figure 1 (a)). First results indicate the real-world capability and search efficiency of our approach.

The reminder of this paper is organized as followed. In Section II related work is discussed. In Section III the problem is formally stated, and in Section IV the presented approach is introduced. Experimental results are presented in Section V, and we finally conclude in Section VI.

II. RELATED WORK

Searching and exploring unknown space is a general type of problem that has been considered in a variety of different areas, such as finding the next best view for acquiring 3d models of objects, the art gallery problem, robot exploration, and pursuit evasion.

Traditional next best view (NBV) algorithms compute a sequence of viewpoints until an entire scene or the surface of an object has been observed by a sensor [Banta *et al.*, 1995; Gonzalez-Banos *et al.*, 2000]. Banta *et al.* [1995] were using ray-tracing on a 3D model of objects to determine

the next best view locations revealing the largest amount of unknown scene information. Although closely related to the problem of exploring 3d environments, NBV algorithms are not necessarily suitable for robot exploration [Gonzalez-Banos *et al.*, 2000]. Whereas sensors mounted on mobile robots are constrained by lower degrees of freedom, sensors in NBV algorithms are typically assumed to move freely around objects without any constraints. The calculation of viewpoints, i.e., solving the question where to place a sensor at maximal visibility, is similar to the art gallery problem. In the art gallery problem [Shermer, 1992] the task is to find an optimal placement of guards on a polygonal representation of 2d environments in that the entire space is observed by the guards.

Nüchter *et al.* [2003] proposed a method for planning the next scan pose of a robot for digitalizing 3D environments. They compute a polygon representation from 3D range scans with detected lines (obstacles) and unseen lines (free space connecting detected lines). From this polygon potential next-best-view locations are sampled and weighted according to the information gain computed from the number of polygon intersections with a virtual laser scan simulated by ray tracing. The next position approached by the robot is selected according to the location with maximal information gain and minimal travel distance. Their approach has been extended from a 2d representation towards 2.5d elevation maps [Joho *et al.*, 2007].

In pursuit-evasion the problem is to find trajectories of the searchers (pursuers) in order to detect an evader moving arbitrarily through the environment. Also here the problem is to determine a set of locations from which large portions of the environment are visible. Besides 2d environments, 2.5d environments represented by elevation maps have been considered [Kolling *et al.*, 2010].

III. PROBLEM FORMULATION

In this section the exploration task is formally described. We first describe the model of the searcher and then the structure of the search space. Then we formulate the search problem based on these two definitions.

We consider mobile robot platforms equipped with a 3d sensor. The 3d sensor generates at each cycle a set of n 3D points $\{\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_n}\}$ with $\mathbf{p_i} = (x_i, y_i, z_i)^T$ representing detected obstacles within the sensor's field of view (FOV). The state of the sensor, and thus the searcher, is uniquely determined in $\Re^3$ by the 6d pose $(x, y, z, \phi, \theta, \psi)^T$, where $(x, y, z)^T$ denotes the translational part (position) and $(\phi, \theta, \psi)^T$ the rotational part (orientation) of the sensor.

Let $\mathcal{X}$ be the space of 6d poses, i.e., $\mathcal{X} \cong \mathbb{R}^3 \times SO(3)$ [LaValle, 2006] and $\mathcal{C} \subset \mathcal{X}$ the set of all possible configurations of the sensor with respect to the kinematic motion constraints of the robot platform. In general, $\mathcal{C}$ depends on the degrees of freedoms (DOFs) of the robot platform. Without loss of generality we assume the existence of an inverse kinematic function $IK(\mathbf{q})$ with $\mathbf{q} \in \mathcal{X}$ that generates the set of valid configurations C. Furthermore, let $\mathcal{C}_{free} \subset \mathcal{C}$ be the set of collision free configurations in $\mathcal{C}$, i.e., the set of configurations that can be taken by the sensor without colliding into obstacles. For computing C_{free} we assume the existence of collision detection function $\gamma : \mathcal{C} \rightarrow \{TRUE, FALSE\}$ that returns $FALSE$ if $\mathbf{q} \in \mathcal{C}_{free}$ and $TRUE$ otherwise. Note that for experiments presented in this paper a mobile robot platform equipped with a sensor mounted on a 5-DOF manipulator has been used. Also note that the set of valid configurations can be precomputed for efficient access during the search using capability maps [Zacharias *et al.*, 2007].

The search space $\mathcal{S}$ can be of arbitrary shape, we only assume that its boundary $\delta\mathcal{S}$ is known but nothing is known about its structure. Similar to the well known concept of occupancy grids in 2d, our 3d search reagin $\mathcal{S}$ is tessellated into equally sized cubes that are stored in a hierarchical 3d grid structure [Wurm *et al.*, 2010]. The size of the cubes is typically chosen according to the size of the target to be searched.

Let the detection set $D(\mathbf{q}) \subset \mathcal{S}$ be the set of all locations in $\mathcal{S}$ that are visible by a sensor with configuration $\mathbf{q} \in \mathcal{C}_{free}$. The problem is to visit a sequence $\mathbf{q}_1, \mathbf{q}_1, \mathbf{q}_3, \ldots$ of configurations until either the target has been found or the entire search space $\mathcal{S}$ has been explored, i.e., $\mathcal{S} \setminus \bigcup_{i=1}^{m} D(\mathbf{q}_i)$ is the empty set. Since our goal is to reduce the total search time, the sequence of configurations has to be as short as possible. Note that this problem is related to the next best view (NBV) problem [Banta *et al.*, 1995] and the art gallery problem [Shermer, 1992].

IV. FRONTIER-VOID-BASED EXPLORATION

In this section the procedure for computing the next best viewpoints based on extracted frontiers and voids is described. This is an iterative procedure where at each cycle of the iteration the following steps are performed: (1) to capture a 3d point cloud from the environment (2) to register (align) the point cloud with previously acquired scans (3) to integrate the point cloud into the hierarchical octomap data structure (4) to extract the frontier cells (see Section IV-A) (5) to extract the voids (see Section IV-B) (6) to determine and score next best view locations (see Section IV-C) (7) to plan and execute a trajectory for reaching the next best view location.

A. Frontier cell computation

In this section we describe the process of extracting frontier cells from the incrementally build octomap data structure. Similar to the occupancy grid classification in 2d frontier-based exploration, all 3d cells in the octomap are classified into *occupied*, *free*, and *unknown*. Whereas occupied cells are regions covered with points from the point cloud and free cells are regions without points, unknown cells are regions that have never been covered by the sensor's field of view.

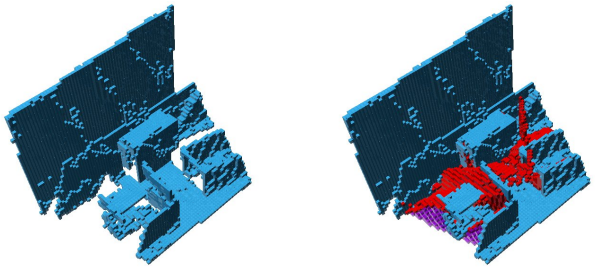


Fig. 2. This figure shows the pointcloud data integrated in an octomap structure (left) and computed frontiers (red) and voids (violet).

The set of frontier cells $\mathcal{F}$ consists of free cells that are neighboring any unknown cell. Note that occupied cells neighboring unknown cells are not belonging to $\mathcal{F}$. For the sake of efficient computation, a queue storing each cell from the octomap that has been updated during the last scan integration, is maintained. By this, frontiers can be computed incrementally, i.e., only modified cells and their neighbors are updated at each cycle. After each update, frontier cells are clustered by an union-find algorithm [Tarjan and van Leeuwen, 1984] forming the frontier cluster set $\mathcal{FC}$.

B. Void Extraction

Similar to frontier cells, void cells are extracted from the octomap in a sequential manner, i.e., only cells modified by an incoming 3d scan are updated during each cycle. The set of void cells $\mathcal{V}$ contains all unknown cells that are located within the convex hull of the accumulated point cloud represented by the octomap. Extracted frontiers and voids can be seen in Figure 2. The convex hull of each sensor update is efficiently computed by using the *QHull* library [Barber *et al.*, 1996].

We are utilizing ellipsoids to build clusters of void cells since they naturally model cylindrical, and spheroidal distributions. The symmetric positive definite covariance matrix for a set of n 3D points $\{\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_n}\}$ with $\mathbf{p_i} = (x_i, y_i, z_i)^T$ is given by:

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{p_i} - \mu)^T(\mathbf{p_i} - \mu), \tag{1}$$

where μ denotes the centroid of the point cloud given by $\mu = 1/n\sum_{i=1}^{n}\mathbf{p_i}$. Matrix Σ can be decomposed into principle components given by the ordered eigen values $\lambda_1, \lambda_2, \lambda_3$, with $\lambda_1 > \lambda_2 > \lambda_3$, and corresponding eigen vectors $\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}$. The goal is to maximize the fitting between the ellipsoids and their corresponding void cells. The quality of such a fitting can be computed by the ratio between the volume of the point cloud covered by the void and the volume of the ellipsoid representing the void:

$$S_i = \frac{N_i R^3}{\frac{4}{3}\pi\lambda_{i1}\lambda_{i2}\lambda_{i3}}, \tag{2}$$

where N_i denotes the number of void cells inside the ellipsoid, R the resolution of the point cloud, i.e., the edge length of a void cell, and λ_{i1}, λ_{i2}, and λ_{i3} the eigen values of the ith void ellipsoid.

Motivated by the work from Pauling et. al [Pauling *et al.*, 2009] voids are extracted from the set $\mathcal{V}$ by randomly sampling starting locations that are then successively expanded by a region growing approach until the score in Equation 2 surpasses a certain threshold value. The procedure terminates after all void cells have been assigned to a cluster. The set of void clusters is denoted by $\mathcal{VC}$, where each cluster $v_i \in \mathcal{VC}$ is described by the tuple $v_i = (\mu_i, \mathbf{e_{i1}}, \mathbf{e_{i2}}, \mathbf{e_{i3}}, \lambda_{i1}, \lambda_{i2}, \lambda_{i3})$, see also Figure 3.

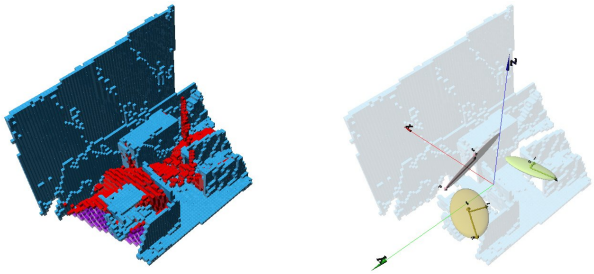


Fig. 3. On the left frontier and void cells are shown. The right side shows the result of the clustering the void cells as three ellipsoids.

The set of void clusters $\mathcal{VC}$ and the set of frontier cells $\mathcal{F}$ are combined in the final *frontier-void* set $\mathcal{FV}$, where each frontier-void $fv \in \mathcal{FV}$ is defined by the tuple $fv_i = (v_i \in \mathcal{V}, F_i \subset \mathcal{FC}, u_i \in \Re^+)$, where u_i describes a positive utility value. In $\mathcal{FV}$ frontier clusters are associated with a void if they neighbor the void cluster. Utility u_i is computed according to the expectation of the void volume that will be discovered, which is simply the volume of the void.

Algorithm 1 *Compute_Next_Best_View*

```
// Compute utility vectors
UV ← ∅
for all fv_i = (v_i, F_i, u_i) ∈ FV do
  for all c ∈ corners(v_i) do
    for all f_i ∈ F_i do
      vp ← f_i
      dir = normalize(pos(f_i) − c)
      UV ← UV ∪ (vp, dir, u_i)
    end for
  end for
end for
// Accumulate utilities in C_free
for all uv = (vp, dir, u) ∈ UV do
  RT ← set of grid cells on the line segment of length
  s_r in direction dir originating from vp.
  for all c ∈ RT do
    util(c) ← util(c) + u
  end for
end for
```

C. Next Best View Computation

The computation of the next best view has the goal to identify the configuration of the sensor from which the maximal amount of void space can be observed. As shown by Algorithm 1 this is carried out by generating the set of utility vectors $\mathcal{UV}$ by considering each frontier cell and associated void cluster from the set of frontier-voids $fv_i \in \mathcal{FV}$. For each $uv \in \mathcal{UV}$ ray tracing into $\mathcal{C}_{free}$ is performed in order to update the expected utility value of possible viewpoints. Ray tracing is performed efficiently on the octomap and at each visited cell of the grid the utility value from the void from which the ray originates is accumulated. In Algorithm 1 s_r denotes the maximal sensor range, function $pos(.)$ returns the 3d position of a grid cell, and $normalize(.)$ is the unit vector. $corners(v_i)$ is a function returning the center and the set of extremas of a void cluster $v_i = (\mu_i, \mathbf{e_{i1}}, \mathbf{e_{i2}}, \mathbf{e_{i3}}, \lambda_{i1}, \lambda_{i2}, \lambda_{i3})$, i.e. $corners(v_i) = \mu_i \cup \bigcup_{j\in\{1,2,3\}} \mu_i \pm \lambda_{ij} \cdot \mathbf{e_{ij}}$.

The space of view points is then pruned by intersecting it with the sensor's workspace, i.e., locations that are not reachable by the sensor due to the kinematic motion model of the robot or due to obstacles in the way, are removed. This can efficiently be implemented by using capability maps [Zacharias *et al.*, 2007]. An example is given in Figure 4.

In the last step of the procedure the generated set of viewpoints is taken as a starting point for determining the next best view configuration of the sensor. For this purpose valid camera orientations (ϕ, θ, ψ) are sampled at the viewpoints c sorted by $util(c)$ for computing the set of valid sensor configurations $\mathcal{C}_{sens} \subset \mathcal{C}_{free}$. Each $c_i \in \mathcal{C}_{sens}$ is defined by the tuple $(x_i, y_i, z_i, \phi_i, \theta_i, \psi_i, U_i)$, where x_i, y_i, z_i denote the viewpoint location, $(\phi_i, \theta_i, \psi_i)$ a valid orientation at this viewpoint, and U_i denotes the expected observation utility for the configuration computed by raytracing the sensors field of view from the pose $(x_i, y_i, z_i, \phi_i, \theta_i, \psi_i)$. The computation stops after N valid poses have been computed in $\mathcal{C}_{sens}$. In our experiments we used $N = 50$. Finally, the next best sensor configuration is determined by:

$$c^* = \arg \max_{c_i \in \mathcal{C}_{sens}} U_i. \tag{3}$$

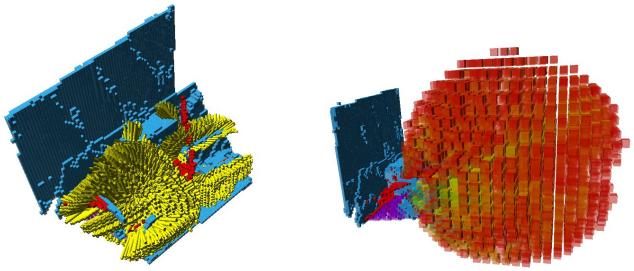


Fig. 4. The left image shows the computed utility vectors. On the right the accumulated utilities pruned by the workspace are shown from red (low), yellow (medium) to green (high).

V. EXPERIMENTAL RESULTS

A. Experimental Platform

The robot is a Mesa Robotics matilda with a Schunk custom-built 5-DOF manipulator that has a work space radius of one meter. It is shown in the top left image in Figures 5 and 6. For 3d obvervations we use a Microsoft Kinect RGBD-camera on the sensorhead. Additionally a Hokuyo-URG 04-LX laser range scanner is mounted on the sensorhead.

B. Results

We conducted experiments in two different settings. The first setting displayed in Figure 5 has a computer and two boxes that are open on the top as well as some two-by-fours. The second setting shown in Figure 6 features two larger closed boxes and smaller boxes with small openings.

For both experiments the robot was positioned in an initial start position. Each execution of one view consists of: Integrating the scan into the world representation, computing the next best view configuration, and moving the sensor to the next best view position configuration. Views were executed until the algorithm reports that there are no more

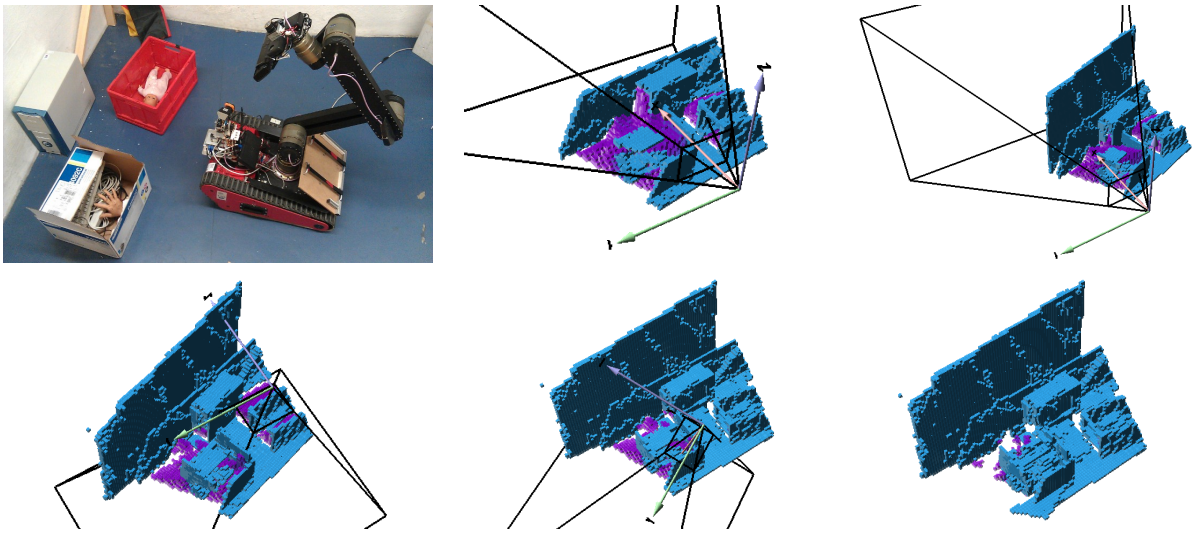


Fig. 5. The figure shows the first experiment. In the top left the experiment setting is displayed. The consecutive images show the best views chosen by the algorithm from the current world representation. The bottom left image shows the final result.

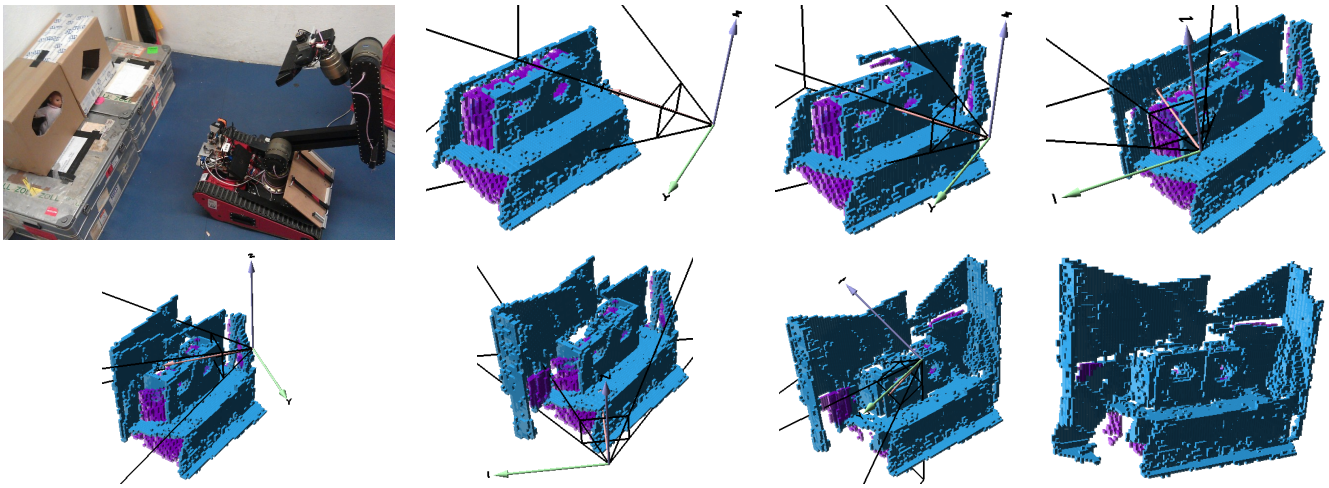


Fig. 6. The figure shows the second experiment. In the top left the experiment setting is displayed. The consecutive images show the best views chosen by the algorithm from the current world representation. The bottom left image shows the final result.

void cells that are reachable by the manipulator, i.e. the algorithm returns a utility of 0 for the best view.

The results of both experiments are shown in Table I and Table II. The integration time notes the time to integrate the scan into the octree and compute the frontier and void property incrementally. Search time gives the time to search for the next best view. The last two columns list the expected number of void cells to be seen by the view and the corresponding volume.

View #	Integration time (s)	Search Time (s)	Void Cells	Utility (dm^3)
1	3.07	22.91	1245	19.45
2	3.21	30.27	988	15.44
3	3.12	99.22	739	11.55
4	3.51	66.85	306	4.78
5	4.20	00.01	0	0.0

TABLE I

THIS TABLE SHOWS THE RESULTS FOR THE FIRST RUN. EACH ROW REPRESENTS ONE SCAN INTEGRATION AND NEXT BEST VIEW COMPUTATION.

View #	Integration time (s)	Search Time (s)	Void Cells	Utility (dm^3)
1	3.83	43.58	2292	35.81
2	3.25	53.32	392	6.13
3	2.99	53.28	167	2.61
4	2.79	67.90	120	1.88
5	3.87	68.53	1615	25.23
6	4.46	14.49	103	1.61
7	4.11	3.35	0	0.0

TABLE II

THIS TABLE SHOWS THE RESULTS FOR THE SECOND RUN. EACH ROW REPRESENTS ONE SCAN INTEGRATION AND NEXT BEST VIEW COMPUTATION.

VI. CONCLUSIONS

We introduced a novel approach for solving the problem of selecting next best view configurations for a 3d sensor carried by a mobile robot platform that is searching for objects in unknown 3d space. Our approach extends the well known 2d frontier-based exploration method towards 3d environments by introducing the concept of voids.

Although the number of sensor configurations in 3d is significantly higher than in 2d, experimental results have shown that frontier-void-based exploration is capable to accomplish an exploration task within a moderate amount of time. Due to the computation of utility vectors from void-frontier combinations the search space of viewpoint configurations of the sensor was drastically reduced. As a comparison perspective consider that the robot's workspace discretized to $2.5cm$ contains $444\,925$ nodes. A rough angular resolution of 10 degrees will result in $444\,925 \cdot 36^3 \approx 2.08 \cdot 10^{10}$ configurations.

The hierarchical octomap structure allowed us to perform efficient ray tracing and neighbor query operations, which are typically expensive when working with 3d data.

However, the performance of our approach needs still to be improved in order to be applicable in real-time. Future improvements will deal with a more compact representation of the inverse kinematik of the robot, as well as a further exploitation of the hierarchical structure for accelerating the search procedure.

We also plan to evaluate the approach on different robot platforms, such as unmanned aerial vehicles (UAVs) and snake-like robots, as well as extending the empirical evaluation for various types of environments, such as represented by realistic point cloud data recorded at USAR training sites.

VII. ACKNOWLEDGMENTS

This work was supported by Deutsche Forschungsgemeinschaft in the Transregional Collaborative Research Center *SFB/TR8 Spatial Cognition* project R7-[PlanSpace].

REFERENCES

[Banta *et al.*, 1995] J. Banta, Y. Zhieng, X. Wang, G. Zhang, M. Smith, and M. Abidi. A "Best-Next-View" Algorithm for Three-Dimensional Scene Reconstruction Using Range Images. *Proc. SPIE (Intelligent Robots and Computer Vision XIV: Algorithms)*, 2588, 1995.

[Barber *et al.*, 1996] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE*, 22(4):469–483, 1996.

[Gonzalez-Banos *et al.*, 2000] H. Gonzalez-Banos, E. Mao, J. Latombe, T. Murali, and A. Efrat. Planning robot motion strategies for efficient model construction. In *Proc. of the 9th International Symposium on Robotics Research (ISRR)*, pages 345–352. Springer, Berlin, 2000.

[Jacoff *et al.*, 2003] A. Jacoff, B. Weiss, and E. Messina. Evolution of a performance metric for urban search and rescue robots. In *Performance Metrics for Intelligent Systems*, 2003.

[Joho *et al.*, 2007] D. Joho, C. Stachniss, P. Pfaff, and W. Burgard. Autonomous Exploration for 3D Map Learning. *Autonome Mobile Systeme 2007*, pages 22–28, 2007.

[Kolling *et al.*, 2010] A. Kolling, A. Kleiner, M. Lewis, and K. Sycara. Pursuit-evasion in 2.5d based on team-visibility. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4610–4616, 2010.

[LaValle, 2006] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[Nüchter *et al.*, 2003] A. Nüchter, H. Surmann, and J. Hertzberg. Planning robot motion for 3d digitalization of indoor environments. In *Proceedings of the 11th International Conference on Advanced Robotics (ICAR '03)*, pages 222–227, 2003.

[Pauling *et al.*, 2009] F. Pauling, M. Bosse, and R. Zlot. Automatic segmentation of 3d laser point clouds by ellipsoidal region growing. In *Australasian Conference on Robotics and Automation*, 2009.

[Shermer, 1992] T. Shermer. Recent results in art galleries. *Proceedings of the IEEE*, 80(9):1384–1399, 1992.

[Tarjan and van Leeuwen, 1984] R. E. Tarjan and J. van Leeuwen. Worst-case analysis of set union algorithms. *Journal of the ACM*, 31(2):245–281, 1984.

[Wurm *et al.*, 2010] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, Anchorage, AK, USA, May 2010. Software available at `http://octomap.sf.net/`.

[Yamauchi, 1997] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 146–151, Monterey, CA, 1997.

[Zacharias *et al.*, 2007] F. Zacharias, Ch. Borst, and G. Hirzinger. Capturing robot workspace structure: representing robot capabilities. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS2007)*, pages 4490–4495, 2007.