Analysis of data structures and exploration techniques applied to large 3D marine structures using UAS*

Margarida Faria[1] Ivan Maza[2] and Antidio Viguria[1]

***Abstract*— This paper is focused on the analysis and comparison of different data structures for 3D space representation in autonomous exploration of large marine structures with UAS. The classical and widely used frontier exploration approach is applied: the frontier cells, which are the locations in the world representation map that are explored and unoccupied but has unexplored space in its vicinity, are of particular interest as they yield the highest information gain. Thus, the data structures have been compared from the point of view of their performance to be applied in the frontier exploration approach. The same algorithm has been run for the different data structures under different scenarios, both with synthetic and real datasets gathered with an UAS. The results are analyzed in detail taking into account the amount of iterations required and the number of computed frontier cells.**

***Index Terms*— Structure Inspection, 3D Space Representation, UAS Applications**

I. INTRODUCTION

The increasing need of UAS usage for remote off-shore monitoring activities has raised challenges for marine environment protection and sustainable management. The European Strategy for Marine and Maritime Research states the need to protect the vulnerable natural environment and marine resources in a sustainable manner. The use of UAS provides increased endurance and flexibility while reducing environmental impact, risk for human operators and total cost of operations. This study has been carried out in MarineUAS [1] framework, an European Union funded doctoral program which strategically strengthens research training on Unmanned Aerial Systems for Marine and Coastal Monitoring.

Several constraints come with this type of scenario. The large size of the area, as it can be clearly seen in Fig. 1, is one such constrint. The structures that need to be inspected are several orders of magnitude bigger than the UAS. In some cases, there is no "a priori" 3D available map usable by the UAS for the navigation. Examples of offshore structures that would benefit from regular systematic and autonomous inspection are petro-stations and windmill farms. They have several structural features that need to be explored in vertical, calling for a full 3D exploration. This exploration must be done in an reduced amount of time, as it may require the activity of the facility to be suspended. In addition, exploring underneath the petro-station in Fig. 1 also requires operating in GPS-denied areas.

Any work done off shore is extremely expensive and has inherently rougher conditions. Reduction in inspection time will lower its cost. Enabling a systematic and autonomous inspection will shift to machines the work at which they excel: repetition without deviation, taking into account many factors. Also by simplifying the inspection, they will get done more often. Additionally as humans need to do less hands on work, they became less exposed to risky situations.

Fig. 1. Inspecting under this offshore petro-station offers an example of inspection that cannot rely on GPS localization.

The first step to achieve autonomous exploration is to have a clear understanding of one of the core components of the architecture of any robot that intends exploration: the data structure used for world representation. The characteristics of real world scenarios need to be taken into account. These scenarios will strain the memory capacity of the system. In real world scenarios, the free space is usually grouped together. Particularly in offshore structures, it is also likely to occupy most of the space. And the occupied space is also likely to be group together. Like the pillars in an offshore platform, the windmill's tower or its blades.

The characteristics of the features used for autonomous navigation should be also taken into account. The driving goal of the UAS is to explore a given area. Many options are available, but this work is focused on the classical and widely used frontier exploration approach. A frontier cell is a location in the world representation map that is explored and unoccupied but has unexplored space in its vicinity. These locations are of particular interest as they yield the highest information gain, and the UAS will only be able to complete its map if it samples all the locations.

*This project has received funding from the European Unions Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 642153.

[1]Margarida Faria and Antidio Viguria are with the Center for Advanced Aerospace Technologies, Sevilla, Spain mfaria@catec.aero and aviguria@catec.aero

[2]Ivan Maza is the Robotics, Vision and Control Group, University of Seville, Avda. de los Descubrimientos s/n, 41092, Sevilla, Spain imaza@us.es

[1]http://marineuas.eu

However, in real world applications the inspection is done with a concrete purpose. Common objectives are updating a 3D map of the structure for later inspection, getting images of particular points in the structures, transporting tools, etc. Each one trimming down the amount of combinations of waypoint sequences that compose useful and efficient plans.

This paper is organized as follows. Section II reviews related work and Section III characterizes readily available data structures. Section IV describes the algorithm used by the implemented system to process the world representation on the different data structures under analysis. Section V presents series of simulations done is several environments, whereas in Section VI is detailed the application to experimental data gathered by an UAS. Finally, conclusions and future work are discussed in Section VII.

II. RELATED WORK

As it has been metioned above, the driving goal of the UAS is to explore a given area and this work is focused on the classical and widely used frontier exploration approach. Then, in this paper the data structures will be examined from the point of view of their suitability to identify the locations that yield the greater information gain. These frontier locations satisfy two conditions: they are in free space and are connected to unexplored space.

Unmanned Ground Vehicles (UGVs) are particularly well suited to reduce the search space to 2D, due to their locomotion. Many applications use probabilistic occupancy grids to tackle the task of exploring unknown (or partially unknown) spaces in a 2D search space. Reference [1] explains the concept of frontier cells over a regular grid in the context of probabilistic occupancy. This concept is not new, it was presented in [2] but due to its simplicity it is still in use today. In [3] an UGV is directed to the nearest frontier region, leaving the task of path planning to a lower level of the architecture with purely reactive obstacle avoidance. In [4], an UGV also travels to the nearest unexplored cell but creates roadmaps, i.e. Voronoi diagrams generated from the occupancy grid.

Many approaches ([3], [5], [6], [7]) encode the status of the cells as free, unknown and occupied either explicitly or by probability thresholds. Here each cells encodes a somewhat different approach to status: free, warning, travel and far. In [8] this approach is extended to multiple vehicles. Each frontier cell is scored according to an heuristic combination of occupancy probability and travel distance. With this combination, the path finding problem is resolved with steepest descent of the heuristic function. In [9] the concept of frontier is combined with a topological map: these edges are calculated as equidistant points to obstacles, the robot then travels this edges marking them as explored. The process continues for as long as there are unexplored edges. Reference [3] is an example of applying this approach to UAS by setting a safe altitude. The frontier cells found at this altitude are then clustered into labeled regions, disregarding the small and inaccessible frontiers. The remaining ones are considered as goals for the UAS, finding the next goal by applying a vector field histogram.

In 3D space there are some applications of probabilistic occupancy grids to solve the next best view problem for robotic arms. In [7], to distinguish between unknown and unoccupied cells a ray casting algorithm is used to extrapolate free regions from the sensor location and occupied points. Holes in sensor measurements are extrapolated with Markov Random Fields. Integrating all this information, the frontier cells are scored relatively to information gain to be later selected as best view. One example where the mission objectives are heavily taken into account is [6]. From the probabilistic grid, a mesh is created as a tool to find void regions. Unknown cells are set at the center of ellipsoids, that expand while maintaining a minimum fitting quality. The ellipsoids are then combined with frontiers and scored according to neighboring voids. The heuristic function maximizes the information gain taking into account the priority each region has for the mission. Knowledge about the area critical for obstacle avoidance has the highest priority, followed by the regions affected by the robot's tools.

Another structure used for 3D space is the octree. One of its defining characteristics is its multi-resolution quality. In [5] the list of frontier cells is compressed as clusters with an union-finding algorithm. The unknown spaces are handled as macro regions through ellipsoid expansion. Finally the clusters are combined with the ellipsoids to score frontiers according to unknown region dimension. Another paper using the octree as its underlaying structure is [10]. However, the configuration space is searched by means of a Rapidly Exploring Random Tree, grown iteratively through safe configurations in the direction of the frontier. In [11] the information of the known space is stored in a map structure similar to an elevation map, although for other purposes the associated point cloud is stored in different data structures. The search for areas with greater information gain is done through sample generation, to address the issue of search space explosion in 3D. From a sample of known points of the environment, other points are generated and added to the pool. The model dynamics of the expansion of the molecules of a perfect gas is used to generate these points. Then, change rate between particle expansions is evaluated to find frontiers in regions.

III. DATA STRUCTURES FOR 3D SPACE REPRESENTATION

In this paper, data structures readily available as off the shelf libraries have been considered and evaluated. The amount of data translated into point clouds from most sensors is extremely high. Thus, it is crucial to select data structures that more than just compress data but also arrange information in a useful and efficient manner.

One of the first approaches used for world representation is the regular grid. The world is discretized into spaces of the same dimension. This approach is very simple and can be applied without a dedicated library. The random access has a complexity of $\mathcal{O}(1)$. Any kind of information can

be stored per cell, although with more information comes greater memory usage.

In order to make search more efficient, trees (with all their multitude of implementations and variations) are another option. As it has been mentioned above, an approach that lends itself particularly well to 3D space is the octree. Two notable implementations of this structure are available: the octomap library [12] and the pcl library. The latter offers several structures and this paper is focused on OctreePointCloudOccupancy [13]. Both offer random access with $\mathcal{O}(1)$ complexity and multi resolution queries. However, they differ in important details. The pcl library has a great focus on the compression needed for streaming, while the octomap library targets navigation and exploration. In the pcl library new measurements are added by summing points, whereas the octomap library integrates them in a probabilistic manner. The concept of unknown scpace is also slightly different in each implementation. In the pcl library it is simply either occupied or free. In the octomap library only location with information is created thus encoding implicitly unknown space. Both pcl and octomap libraries concern themselves with efficiency: the former focuses on read/write efficiency, and for this reason goes so far as to include a double buffered version of the structure. In the latter, the focus lies on memory efficiency with (almost) lossless compression regarding occupancy. For this reason, each node stores only the occupancy probability and one child pointer - forfeiting voxel size, coordinates and the full children array.

Another approach is to create meshes from the point cloud. A popular algorithm is the Delaunay triangulation. Both pcl and CGAL [14] libraries provide this implementation. Finally, it is worth mentioning the approach of working with samples. Although not readily provided by a library, the search space is greatly reduced.

IV. ALGORITHMS IMPLEMENTED

To assess the impact of search space explosion while searching for frontier cells, a framework was created to run all the different combinations between data structures and search space under the same exact conditions. The diagonal directions were disregarded as all the frontier regions were identified without them.

Each data structure needs to provide a function that returns its neighbors and a set of functions to implement iteration. Namely initialization, iteration and end condition. These will be called from the same exact implementation of Algorithm 1. The algorithm searches for frontier cells from the initial iteration condition until the end condition. The evaluation made for each cell selects locations that meet the following requirements: it must be in known space, it must be unoccupied and it must have at least one neighbor that is unexplored. Again, the process of finding the neighbors is specific to each data structure. The dimension flexibility is given by the bounding box set at the beginning of the iteration and by adjusting the number of directions a neighbor should be created in. In all cases, the neighbors are in adjacent cells.

The algorithms have been implemented in C++ using the Robot Operating System (ROS) [15] as middleware. The open source implementation is freely available in the form of a self-contained Robot Operating System (ROS) unit tests. It was released under the MIT-license and can be obtained from https://github.com/margaridaCF/dataStructureAnalysis. More data structures can be integrated in a straightforward way, the only requirement is to provide the four varying functions: obtain the neighbors, iteration initialization, end condition and next cell.

Algorithm 1 Generic procedure to find frontier cells. The aspects that change between the data structures are finding the neighbors of a cell, initialization of iteration, iteration and end condition for iterator. For each cell, first it is determined if the cell is explored and in free space. When these conditions are met, its neighbors are evaluated. If there is at least one neighbor that is in unknown space, the cell will be classified as a frontier.

Input: $dimensions$, $variation\_spec$
Output: $frontier\_cells$

```
cell = variation_spec.initIteration(min, max)
while cell != variation_spec.endIteration() do
  if isExplored(cell)&& !isOccupied(cell) then
    frontier = false
    neighbors =
    variation_spec.getNeighbors(dimensions)
    for all neighbors : n do
      if !isExplored(n) then
        frontier = isExplored(n) || frontier
      end if
    end for
    if frontier then
      frontier_cells.add(cell)
    end if
  end if
  cell = variation_spec.getNextCell()
end while
return frontier_cells
```

A. Regular Grid

The regular grid makes a discretization of the continuous space into cells that always have the same dimensions. This classical approach is frequently still used due to its simplicity.

The full iteration of such a grid will always require a number of iterations given by multiplying the length, width and height of the 3D space considered. This implementation is based on a simple regular increment of the coordinates. The neighbors are always at the same distance and their computation algorithm is shown in Algorithm 2.

B. Sparse Grid

The sparse grid extends the concept of the regular grid by grouping same value regions. Its tree-like approach divides the space in different sizes, creating a high resolution cell

Algorithm 2 Regular grid neighbor calculation. The dimensions of the search space will determine if neighbors above and below the cell will be add to the output list. The calculation of the distance to the neighbor varies accordingly to the data structure. Here it is always the same as it refers to a regular grid. In a sparse grid, the size of the current voxel takes the place of the grid resolution.

Input: $current$, $dimensions$
Output: $neighbors$
1: **if** $dimensions$ == 2D **then**
2: $directions$ = $\{up, down, left, right\}$
3: **else if** $dimensions$ == 3D **then**
4: $directions$ = $\{up, down, left, right, up, down\}$
5: **end if**
6: **for all** $directions : d$ **do**
7: $neighbors$.add ($current$ + ($d \times grid\_resolution$))
8: **end for**
9: **return** $neighbors$

only to accommodate known points extracted from the point cloud. In this particular implementation, information is added not only for detected obstacle location but also to the free space that can be extrapolated from them. By transversing the grid passing only through a known leafs, all the cells unexplored will be skipped. This is quite useful since by definition, any unknown cell can never be a frontier cell.

The state of each location is stored in log-odds notation to enable probabilistic fusion of each new point cloud. The compression is nearly lossless, and there is only a trimming of the maximum and minimum values. Additionally, macro regions with the same state will be analyzed only once. This will also reduce the amount of cells that need to be examined to find frontiers, as the maximum size of each cell is always lower than the sensor limit. It is the regular grid equivalent of analyzing several cells at the same time.

The algorithm to advance to the next leaf location is detailed in [12]. To calculate the neighbors, a similar algorithm to Algorithm 2 is applied, being the main difference to take into account that the coordinates of each cell refer to the center of the location. In this work, the library under analysis is the Octomap library.

V. SIMULATION RESULTS

A. Simulation Environment

Five datasets were used to run the tests under simulation. They were all generated using ROS nodes on a Gazebo simulator. The sensor used emulates a VLP-16 LIDAR: it is omnidirectional, dividing the 360 degrees in 1500 samples and stacking it 16 times. The sensor was mounted on the front of a model of a quadrotor (see Fig. 2) while transversing the scenarios.

The five scenarios shown in Fig. 3 have been considered. Three of them are the same as the ones used in [16]: an enclosed space with only one opening (room), a circular corridor with only one opening to the inner area and a Z shaped corridor with openings at both ends. In these scenarios the UAS fully explores the enclosed area, never crossing an opening. Each of the openings is 0.8 meters long. The dimensions of the simulated worlds are detailed in Table V-A. For a scale closer to the targeted application, there is a scenario that emulates the lower part of an offshore petro-station. In this scenario, the ground is simulating the sea level and the floor of the station is above the UAS - too far to be sensed. From this scenario two datasets were extracted, one where the UAS inspects two pillars and the second one were the UAS inspects 3 pillars. The comparison of two information states in the same scenario will bring insight into the effect of unknown space in the analysis of the world.

(in m)	Room	Corridor	Z	2 pillars	4 pillars	Pillars
Width	24	24	22	77.8	180.4	13
Length	8	14	13.8	77.2	90	10

TABLE I
DIMENSIONS OF THE SCENARIOS USED FOR THE SIMULATIONS IN METERS.

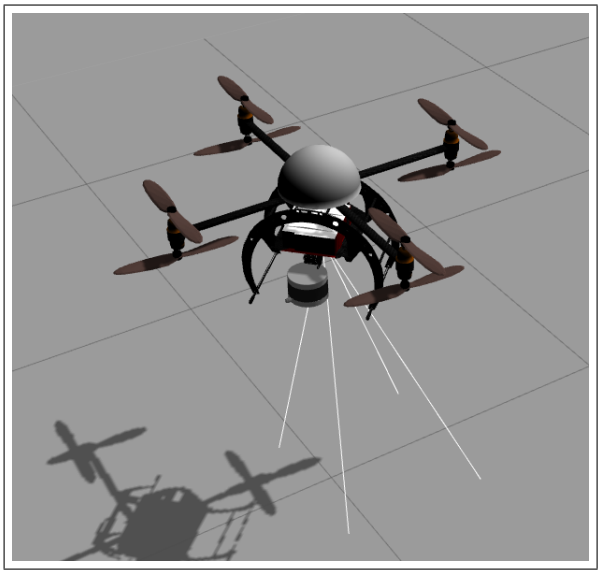


Fig. 2. The UAS model used in simulation with its VLP-16 LIDAR in front.

B. Results analysis

In all the scenarios, both in 2D and 3D, the frontier regions are roughly the same. No region goes undetected in any case. This result is illustrated in Fig. 4.

The same regions are detected, however the number of cells needed to represent them is smaller. This difference tends to be small using a LIDAR as the edge of the sensor is often times irregular due to the increased angular interval between rays. However, when in large scenarios (as 2 pillars or 3 pillars), it starts to gain more and more relevance. Also, the geometrical disposition of the voxels and how neighbors are computed explains this result. In both grids, the direction in which to test is applied to a reference point

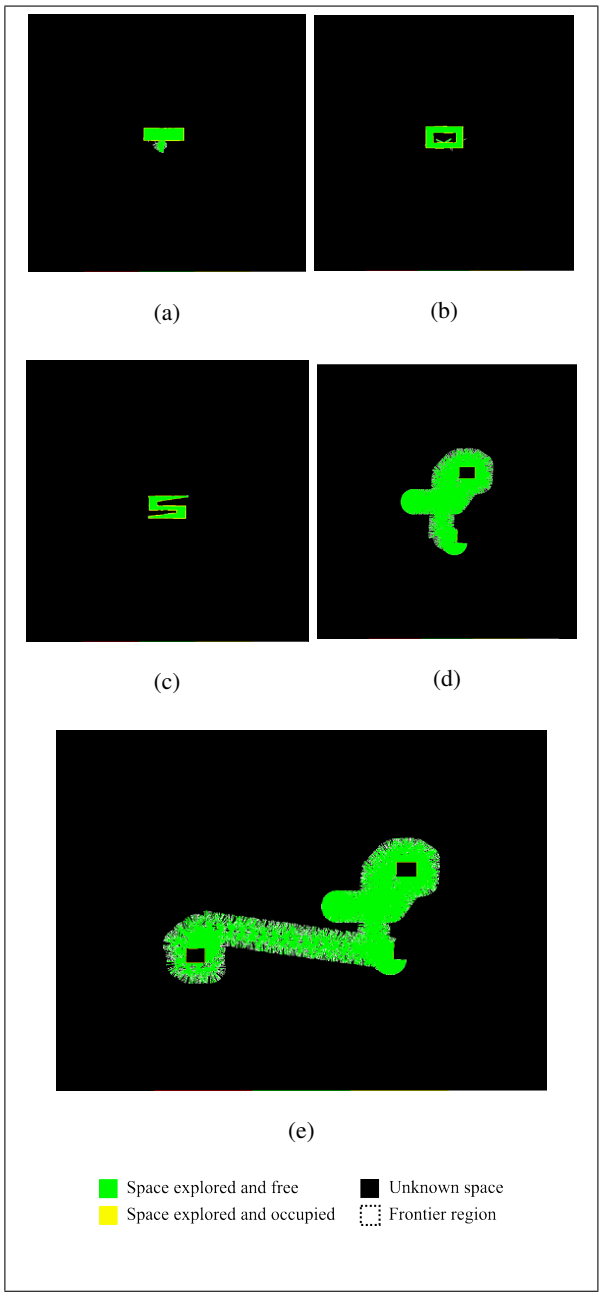


Fig. 3. The different used scenarios (a) Room. (b) Circular corridor. (c) Z corridor. (d) Offshore petro-oil structure, two pillars sensed (e) Same structure one additional pillar sensed.

of the voxel. When a larger explored voxel is adjacent to the same unknown voxel, its whole volume is considered a frontier. However, in a regular grid the added volume is always the size of the grid resolution. Computed results are shown in Fig. II.

For each scenario four combinations are run: regular grid in 2D, regular grid in 3D, sparse grid in 2D and sparse grid in 3D. Only the portion of the world above the ground was considered, more accurately between zero and one meter altitude. This interval was chosen for study to make the comparisons with the 2D space within the same magnitudes. By comparing the execution time of each run depicted in Fig. 5, it is clear that the sparse grid needs less time to find the frontier cells.

The number of iterations needed to process the whole structure is always lower in a sparse grid. More specifically, the number of iterations is one order of magnitude higher for regular grids in the 2D case and two orders of magnitude higher in the 3D case, as it can be seen in Fig. 6. The linear growth in a logarithmic scale shows the exponential progression of the iterations amount. The granularity afforded by the hierarchical nature of the structure certainly predisposes this progression since the worst case scenario of the sparse grid is a regular grid. However the disparity of the results can not be explained only by its hierarchical nature.

Another factor to be taken into account is the amount of unknown cells. As it can be seen in Fig. 7, the unknown space is what composes the vast majority of the surveyed area at that moment, opening the possibility of restricting the analyzed cells to the known cells.

The size of the space represented affects the number of iterations and therefore the execution time. The results of the scenarios with two and four pillars reflect it. Here the world is exactly the same - only the amount of information changes. However, in the scenarios Room, Corridor and Z, the increase in space does not correlate to the iterations amount or execution time. As the world configuration in each scenario changes drastically, the distribution of the sizes of the voxels generation for its representation changes with it.

VI. EXPERIMENTAL RESULTS

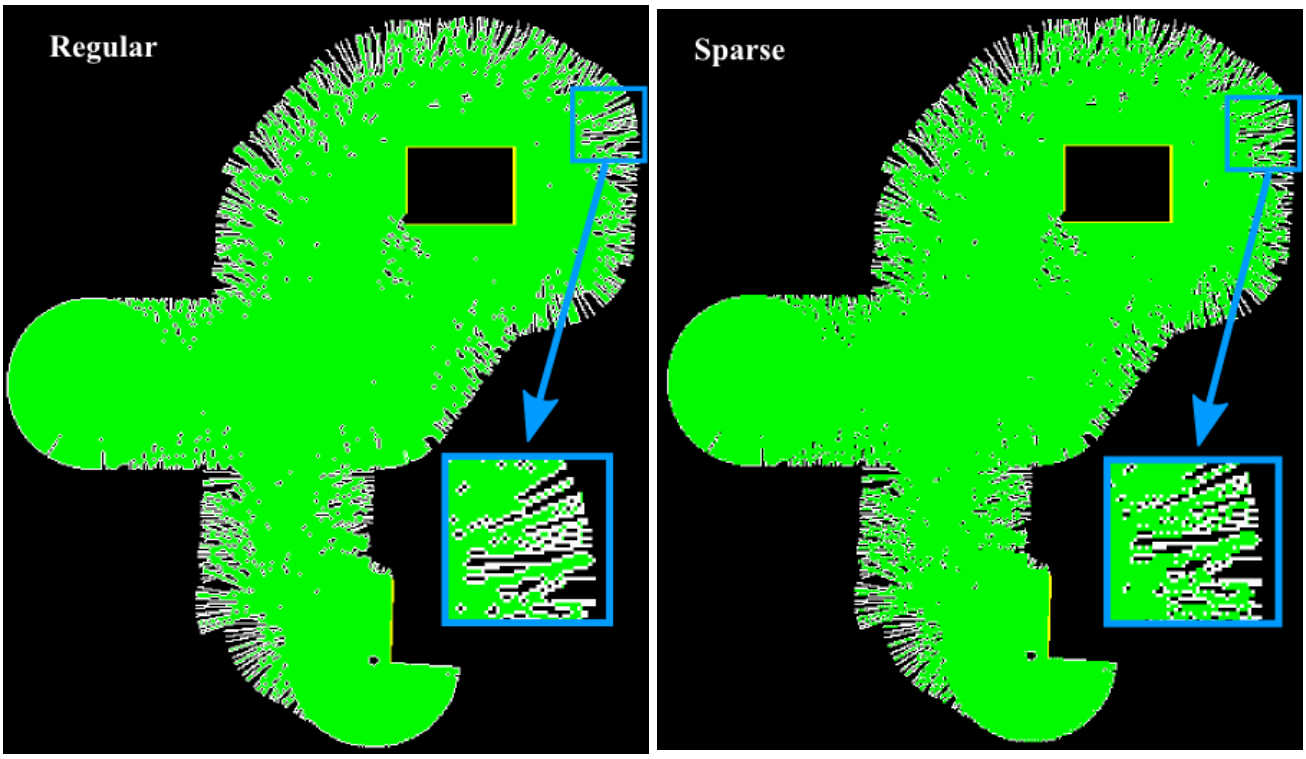
The algorithms have been applied to a dataset captured in a single flight by a real UAS, although in a more confined space.

A. Platform description

The platform used for data acquisition was the quadcopter shown in Fig. 8. It was equipped with two RGB-D cameras Asus Xtion Pro Live, one facing forwards and another facing backwards. The images captured by the camera were then combined and analyzed to generate the point cloud. This data was recorded in an indoor testbed with dimensions 15x15x5 meters at the Center for Advanced Aerospace Technologies (CATEC) located in Seville (Spain). Figure 9 shows the point cloud captured in the testbed.

B. Results analysis

The dataset captured in a real scenario was analyzed using the same source code (both 2D and 3D in each data structure). Again only the portion of the world above the ground between zero and one meter altitude was considered. The results are consistent with the ones observed with datasets captured in simulation. Concerning execution time, the octree keeps preforming significantly better as it can be seen in Fig. 10. To analyze efficiency from a platform independent point of view, the number of cells was evaluated. In Fig. 11 again it can be seen the high proportion of unknown cells which corroborates the explanation of the efficiency gain by the amount of skipped cells.



(a) (b)

Fig. 4. The frontier cells found in the two pillar scenario using the sparse grid data structure. (a) Shows the results using the regular grid. (b) Shows the results using the sparse grid.

TABLE II

FRONTIER REPRESENTATION FOR SIMULATION DATASET. THE UNITS OF SPACE ARE M^2 AND M^3 FOR TWO AND THREE DIMENSIONS RESPECTIVELY.

		2D			3D		
		Regular	Sparse	Difference	Regular	Sparse	Difference
Room	Frontier cells	236.0	221.0	15.0	50,903.0	47,460.0	3,443.0
	Space	9.4	9.7	-0.2	407.1	407.0	0.1
Corridor	Frontier cells	137.0	110.0	27.0	5,729.0	1,951.0	3,778.0
	Space	5.5	5.8	-0.4	45.8	22.8	23.1
Z	Frontier cells	67.0	61.0	6.0	24,210.0	20,873.0	3,337.0
	Space	2.7	2.9	-0.2	193.7	185.1	8.5
2 pillars	Frontier cells	9,000.0	6,809.0	2,191.0	845,091.0	790,015.0	55,076.0
	Space	360.0	386.2	-26.2	6,718.9	7,028.2	-309.3
3 pillars	Frontier cells	28,627.0	21,724.0	6,903.0	1,273,737.0	1,156,968.0	116,769.0
	Space	1,145.0	1,165.6	-20.7	10,067.7	11,047.3	-979.6

Another interesting result refers to the frontier space and the amount of cells needed to represent it. In Table III, although the frontier space represented by the octree deviates by around 1.3 (square and cubic meters respectively), the number of cells that represent it is reduced. It is hypothesized that this is not a variation that will greatly impact the further processing of the frontier cells for selecting a goal, but it is a reduction nonetheless.

TABLE III

FRONTIER REPRESENTATION FOR THE EXPERIMENTAL DATASET. THE UNITS OF SPACE ARE M^2 AND M^3 FOR TWO AND THREE DIMENSIONS RESPECTIVELY.

	2D			3D		
	Regular	Sparse	Change	Regular	Sparse	Change
Cells	1341	1067	274	10394	8652	1742
Space	13.4	12.0	1.4	10.4	11.7	-1.3

VII. CONCLUSIONS AND FUTURE WORK

In this paper, it has been analyzed the impact of the underlying data structure on processing the world representation to find frontier cells. In the analyzed octree implementation (octomap), some characteristics emerge as beneficial for efficiency. The baseline used was a regular grid with the same resolution as the octree. The number of iterations needed to find frontier cells was less in all cases by, at least, one order of magnitude. Grouping regions with the same status and skipping the unknown cells explain these results. The

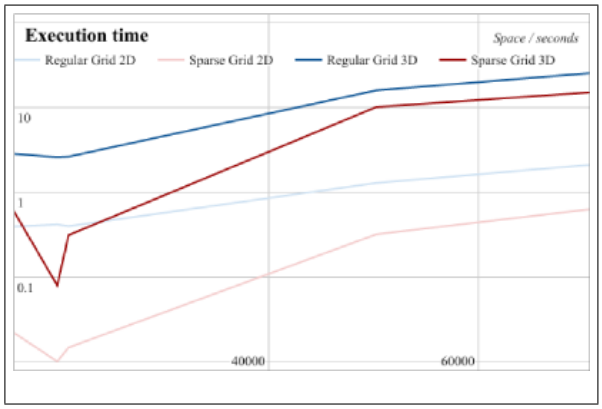


Fig. 5. Analysis of the influence of the search space explosion on the execution time. Each scenario appears first as 2D and then as 3D, using a logarithmic scale for both axis. The units of space are M^2 and M^3 for two and three dimensions respectively.

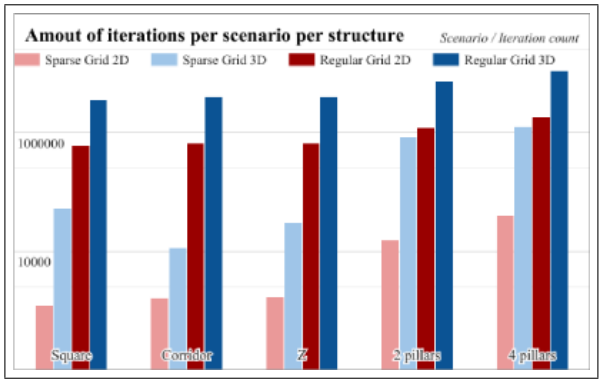


Fig. 6. A comparison of the iterations amount needed to transverse the world representation for each of the data structures, both in two dimensions and three dimensions using a logarithmic scale for the space axis. The units of space are M^2 and M^3 for two and three dimensions respectively.

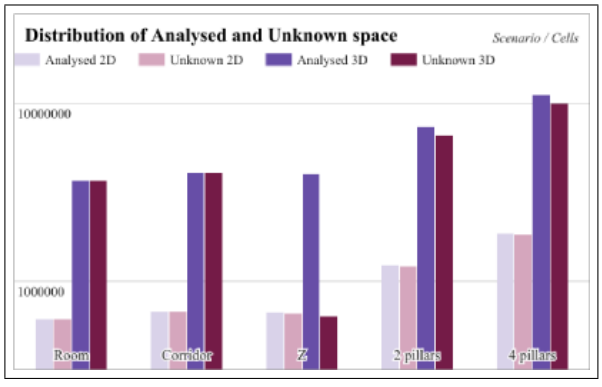


Fig. 7. The amount of space between unknown space during exploration is very large, in any scenario, in any dimension. This graph is using a logarithmic scale for the space axis. The units of space are M^2 and M^3 for two and three dimensions respectively.

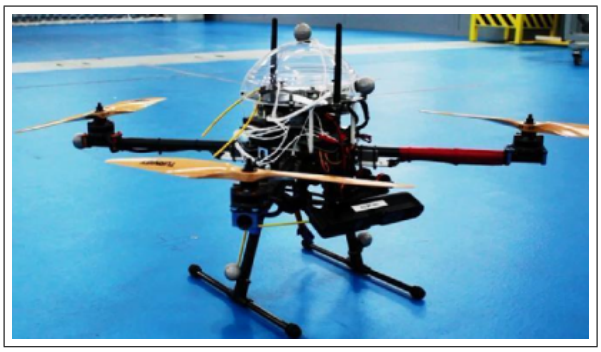


Fig. 8. UAS platform used for data acquisition.

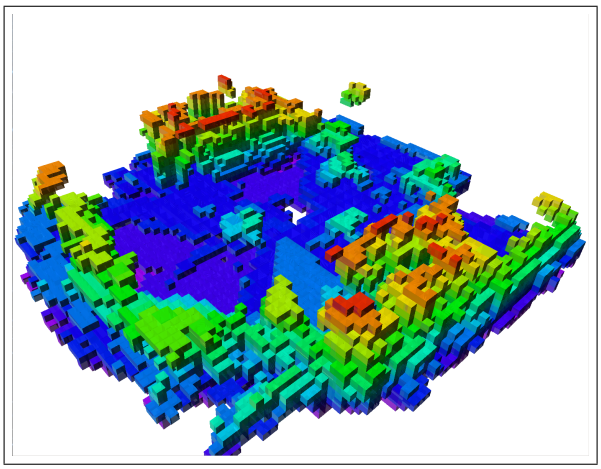


Fig. 9. Point cloud captured in an indoor testbed with the UAS shown in Fig. 8 at the Center for Advanced Aerospace Technologies (CATEC) located in Seville (Spain).

number of frontier cells is smaller for the sparse grid in all of the datasets, while covering a similar amount of area. This indicates that these cells better summarize the frontier regions, providing some degree of grouping. Although it will be beneficial for any cell processing, it is not a big enough difference as to allow to concluded it will create a sizable impact.

Three different future directions of research are of interest. On the one hand, it would be of great interest to integrate more data structures and to add more implementations of the octree such as the pcl cloud. On the other hand it would be valuable to compare entirely different data structures. One type of data structure of particular interest is a triangle mesh extracted with Delaunay triangulation. Again several implementations exist (CGAL, pcl), comparing them would bring a greater insight into their application. Another research direction is to evaluate the performance of the search algorithm throughout a mission. Such survey would permit a thorough comparison between increasing amounts of information in the same scenario, while using each different data structure. Finally, applying these results to UAS exploring very large areas in the real world.

REFERENCES

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*, ser. Intelligent robotics and autonomous agents. Cambridge (Mass.) (London):

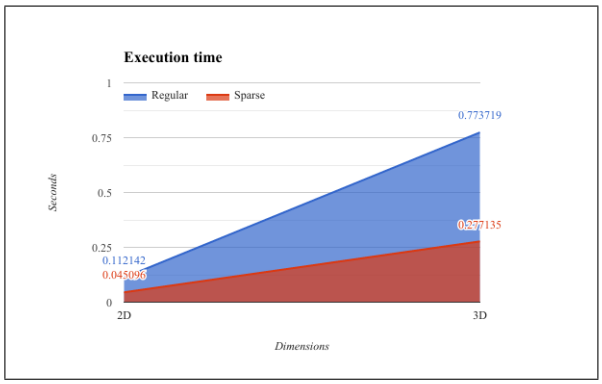


Fig. 10. Comparison of execution time between data structures and dimensions for experimental dataset. In the graph a logarithmic scale is used for the time axis.

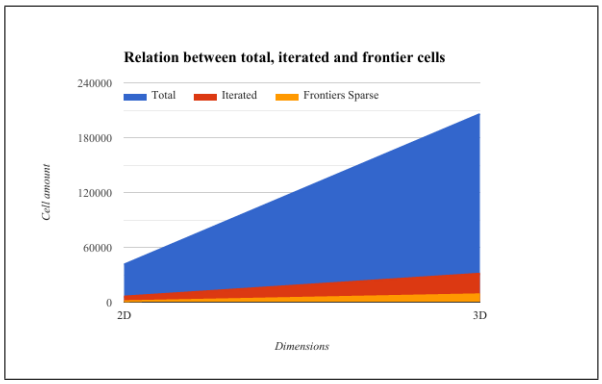


Fig. 11. Relation between total amount of cells, the cells iterated through with the sparse data structure and the amount of frontier cells found. The amount of unknown cells can be extrapolated as the blue area, since the values are not stacked. In the graph a logarithmic scale is used for the vertical axis.

The MIT Press, 2005.

[2] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*. IEEE Comput. Soc. Press, 1997, pp. 146–151. [Online]. Available: http://ieeexplore.ieee.org/document/613851/

[3] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys, "Vision-based autonomous mapping and exploration using a quadrotor MAV," *IEEE International Conference on Intelligent Robots and Systems*, pp. 4557–4564, 2012.

[4] L. Romero, E. Morales, and E. Sucar, "A Robust Exploration and Navigation Approach for Indoor Mobile Robots Merging Local and Global Strategies," *Advances in Artificial Intelligence: International Joint Conference 7th Ibero-American Conference on AI 15th Brazilian Symposium on AI IBERAMIA-SBIA 2000 Atibaia, SP, Brazil, November 19–22, 2000 Proceedings*, pp. 389–398, 2000. [Online]. Available: http://dx.doi.org/10.1007/3-540-44399-1_40

[5] C. Dornhege and A. Kleiner, "A frontier-void-based approach for autonomous exploration in 3d," in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*. IEEE, nov 2011, pp. 351–356. [Online]. Available: http://ieeexplore.ieee.org/document/6106778/

[6] G. Paul, S. Webb, D. Liu, and G. Dissanayake, "Autonomous robot manipulator-based exploration and mapping system for bridge maintenance," *Robotics and Autonomous Systems*, vol. 59, no. 7-8, pp. 543–554, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2011.04.001

[7] 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. [Online]. Available: http://dx.doi.org/10.1016/j.jvcir.2013.07.006

[8] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[9] H. Choset and K. Nagatani, "Topological simultaneous localization and mapping (SLAM): Toward exact localization without explicit localization," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 2, pp. 125–137, 2001.

[10] L. Freda, G. Oriolo, and F. Vecchioli, "Sensor-based exploration for general robotic systems," *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 2157–2164, 2008.

[11] S. Shen, N. Michael, and V. Kumar, "Autonomous multi-floor indoor navigation with a computationally constrained MAV," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 20–25, 2011.

[12] 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.

[13] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[14] S. Hornus, O. Devillers, and C. Jamin, "dD triangulations," in *CGAL User and Reference Manual*, 4.9 ed. CGAL Editorial Board, 2016. [Online]. Available: http://doc.cgal.org/4.9/Manual/packages.html#PkgTriangulationsSummary

[15] 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.

[16] S. Shen, N. Michael, and V. Kumar, "Stochastic differential equation-based exploration algorithm for autonomous indoor 3D exploration with a micro-aerial vehicle," *The International Journal of Robotics Research*, vol. 31, no. 12, pp. 1431–1444, oct 2012. [Online]. Available: http://ijr.sagepub.com/cgi/doi/10.1177/0278364912461676