2018 IEEE International Conference on Robotics and Automation (ICRA)
May 21-25, 2018, Brisbane, Australia

Optimized environment exploration for autonomous underwater vehicles

Eduard Vidal[1], Juan David Hernández[1], Klemen Istenič[1], and Marc Carreras[1]

***Abstract*— Achieving full autonomous robotic environment exploration in the underwater domain is very challenging, mainly due to noisy acoustic sensors, high localization error, control disturbances of the water and lack of accurate underwater maps. In this work we present a robotic exploration algorithm for underwater vehicles that does not rely on prior information about the environment. Our method has been greatly influenced by many robotic exploration, view planning and path planning algorithms. The proposed method constitutes a significant improvement over our previous work [1]: Firstly, we refine our exploration approach to improve robustness; Secondly, we propose an alternative map representation based on the quadtree data structure that allows different relevant queries to be performed efficiently, reducing the computational cost of the viewpoint generation process; Thirdly, we present an algorithm that is capable of generating consistent maps even when noisy sonar data is used. The aforementioned contributions have increased the reliability of the algorithm, allowing new real experiments performed in artificial structures but also in more challenging natural environments, from which we provide a 3D reconstruction to show that with this algorithm full optical coverage is obtained.**

I. INTRODUCTION

One of the areas where autonomous underwater vehicles (AUVs) stand out over other available technologies, such as remotely operated vehicles (ROVs) or manned submersible vehicles (MSVs), is in inspection and mapping of complex underwater reliefs. In situations where the vehicle's navigation system is accurate and there are no unexpected obstacles in the environment, following a preplanned trajectory is usually sufficient to obtain good quality data. However, in challenging scenarios where these conditions are not met, the vehicle must be guided by an online path planner to adapt the exploration trajectory to the relief perceived in situ.

Different algorithms have been proposed to automate the exploration of a scene. These algorithms can be classified into three main categories:

- View planning (VP) algorithms, which focus on determining a suitable set of viewpoints that satisfy a particular reconstruction or exploration criteria [2].
- Frontier-based (FB) algorithms, which focus on the boundaries between different parts of a map to extract information about what to do next [3].
- Coverage path planning (CPP) algorithms, which compute a path that passes over all points of a surface of interest while avoiding obstacles, by having usually a prior map of the scene [4].

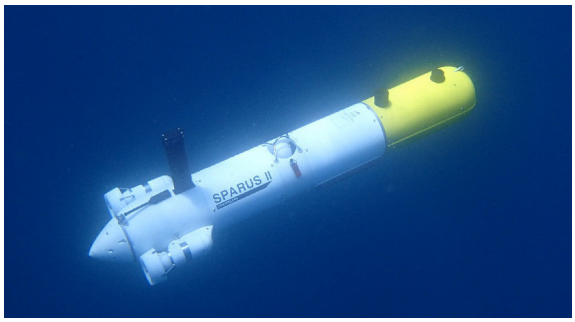

Fig. 1. Sparus II AUV, a torpedo-shaped underwater robot with partial hovering capabilities used to validate our approach.

In this paper we present a method to automate the exploration of unknown underwater environments. Our method has been inspired by VP and FB approaches, and by our previous publication [1]. The method does not require prior information about the environment. First, information about the environment is incorporated into a 2-dimensional (2D) grid map, where each cell is labeled according to six possible states. Then, the map is used to generate viewpoints that promote further exploration of the environment. Those viewpoints are generated using the frontiers between different labels. Among all possible viewpoints, the best one is chosen and a path is generated to steer the robot towards the selected viewpoint. This procedure is repeated until the exploration finishes.

In this publication not only do we refine our exploration approach to increase robustness, but we also tackle the following two important problems. The first one is to improve the map representation so that viewpoints can be generated more efficiently. This is necessary because of the lower computational resources available in underwater robots (and because of the online nature of the algorithm), and can also be interesting in other robotic fields. The second one is to improve the robustness of the algorithm when noise is present, specifically against false negative measurements in acoustic exteroceptive sensors. Although this is also domain specific, our method can be useful to anyone using a sensor that suffers from this type of noise. As shown in the results section, solving these two problems truly enables the algorithm to perform as intended in the underwater domain.

Work on this paper has been supported by the EXCELLABUST, ROBOCADEMY and ARCHROV Projects under the Grant agreements H2020-TWINN-2015, CSA, ID: 691980, FP7-PEOPLE-2013-ITN-608096 and DPI2014-57746-C3-3-R respectively, and by the Spanish Government FPU14/05493 PhD grant (to E. Vidal).

[1]E. Vidal, J.D. Hernández, K. Istenič, and M. Carreras are members of the Underwater Robotics Research Center (CIRS), University of Girona, Spain. J.D. Hernández current affiliation is: Department of Computer Science, Rice University, Houston, USA. eduard.vidalgarcia@udg.edu, juandhv@rice.edu, klemen.istenic@udg.edu and marc.carreras@udg.edu.

The remainder of this paper is organized as follows. Section II briefly overviews relevant related work on robotic exploration, view planning and path planning. Section III describes the proposed online exploration method for autonomous inspection of unexplored underwater structures and Section IV provides a detailed description of the contributions related to increase the consistency of the map. Section V reports obtained results under simulated and real experiments using the Sparus II AUV (see Figure 1). Finally, in Section VI conclusions are presented and directions for further research are discussed.

II. RELATED WORK

Over the last three decades there have been a significant amount of contributions to the robotic exploration and VP fields. This section briefly reviews the work of important authors.

In [5] and [6], VP techniques were applied to ship hull inspection, but a prior rough map was required to plan the path to explore the propellers and rudders. A similar approach was presented by Bircher et al. [7] for structural inspection using unmanned aerial vehicles (UAVs). The same authors have recently proposed a method that uses a rapidly-exploring random tree (RRT) to perform exploration without a prior map [8] [9]. Williams et al. [10] proposed a VP target reinspection method for AUVs equipped with a synthetic aperture sonar (SAS). Galceran et al. [11] presented a 2.5-dimensional (2.5D) approach for inspection of complex underwater structures. In this approach, a nominal path is planned in advance using a prior map of the scene. Then, during the mission execution, the path is reshaped online to deal with the navigation drift and inaccuracies from the initial map. Blaer et al. [12] presented a VP approach for 3-dimensional (3D) site modeling with unmanned ground vehicles (UGVs). In this case, a minimal set of covering views is planned in 2D using a prior map of the scene, and then the resulting model is improved in a second stage in which further 3D views are planned. In [13] Connolly proposed the next-best-view (NBV) methodology to autonomously plan views to reconstruct a 3D object. Latombe et al. applied NBV strategies to robotic exploration [14]. In the same line, Vasquez-Gomez et al. [15] presented a NBV algorithm to model arbitrary objects in 3D, and in [16] the same authors refined their method by adding uncertainties. Their method does not use prior knowledge about the object's shape, but it does require information about its position and size. Isler et al. [17] have also developed a NBV approach with uncertainties for active volumetric 3D reconstruction. Finally, Burgard et al. explored FB methods and even extended them to work with multiple robots in [18].

III. VIEW PLANNING EXPLORATION ALGORITHM

The proposed algorithm operates according to the hierarchical/deliberative robotic paradigm: first incorporate the data from the sensors into a map (sense), then use the map to generate viewpoints that promote further exploration of the environment, and also generate a safe path to reach the generated viewpoint (plan), and finally steer the robot so that it follows the generated path (act).

A. Map Representation (Sense)

The proposed view planning (VP) approach plans the next-best-view (NBV) at each iteration so that full coverage of the scene is achieved with data from the following sensors:

- *A profiling sonar* that acquires 2D exteroceptive data of the surroundings. Since a full 360^o scan usually takes several seconds to be completed, in our approach the scanning sector has been limited to 120^o. It is reoriented online so that it points towards the target cell, while always covering the front part of the vehicle for safety purposes.
- *An optical camera* that gathers images of the scene. There is no need to have online feedback from the camera, since only its field of view (FOV) is taken into account in the planning stage.

The information provided by both sensors is used to incrementally build a representation of the surroundings. The world is represented with a 2D grid map, where each cell can be labeled as:

- *Unknown* cells, representing areas that require further inspection.
- *Empty* cells, reflecting areas that have been inside the sonar FOV, but did not have sufficient sonar detections to be marked as occupied.
- *Occupied* cells, reflecting those regions which had sufficient detections.
- *Viewed* cells, reflecting occupied space that has been inside the camera FOV.
- *Range candidates*, being unknown cells next to empty and occupied cells.
- *Camera candidates*, being occupied cells next to empty and viewed cells.

Initially, all cells are unknown. Figure 2 depicts all possible labels in a simulated case, where also the sonar and camera FOV can be observed.

B. View Planning and Path generation (Plan)

Two tasks are performed during the VP process:

- *Generate sonar and camera viewpoints*: sonar and camera target cells, identified during the map generation process, reflect key locations in the map that must be inspected in order to continue exploration. For this reason, viewpoints are located at a user-defined distance from the target cells along the surface's normal direction.
- *Select the best viewpoint*: the distance between each viewpoint and the current robot configuration is computed as described in our previous work [1]. The computed distance reflects not only how far the viewpoint is with respect to the robot position, but also the required orientation changes.

Figure 3 depicts an example of the viewpoints generated for a given configuration of cells. For visualization purposes,

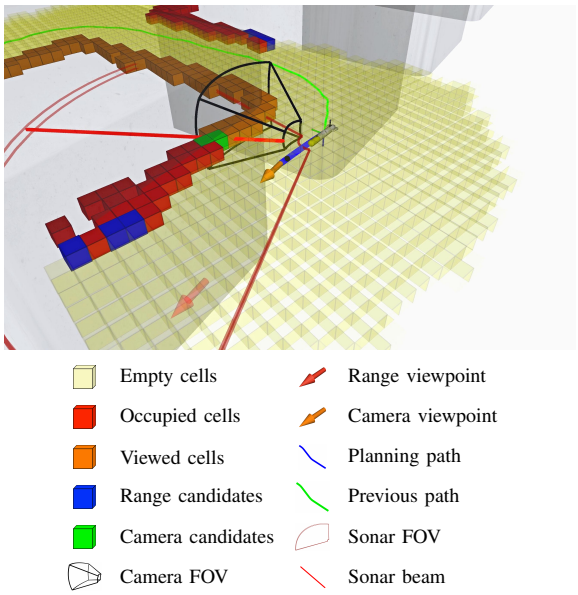

Fig. 2. Example showing the robot performing an inspection of a structure. All possible cell labels are present in a single image. FOVs of both sensors are also shown.

the best sonar and camera viewpoints are displayed, but only the closest one is used to guide the robot. No preference is given to a particular type of viewpoint.

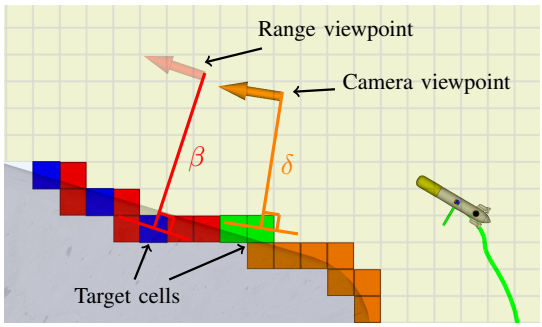

Fig. 3. Example of selected sonar and camera viewpoints. Their corresponding target cells are also highlighted. Viewpoints are placed at a user configurable distance (β and δ) along the perpendicular direction of the estimated surface.

Once the NBV has been selected, a start-to-goal planner is used to generate a path that allows the robot to safely navigate from its current position to the selected viewpoint. We use the asymptotic optimal rapidly-exploring random tree (RRT*) [19] path planner (from the open motion planning library (OMPL) [20]). Different components of the path planner have been reimplemented to adapt its behavior to our needs. The configuration *sampler* has been modified to reuse the last obtained solution, thus ensuring that new solutions are at least as good as the previous one. This characteristic was proposed by Hernández et al. [21] and was also used in our previous work. Figure 4 reflects this behavior. Finally, paths are evaluated according to the integral of a cost function that reflects the amount of occupied cells around a specific configuration. This allows to combine our two main path planning objectives: minimize path length while keeping a safe distance to the obstacles.

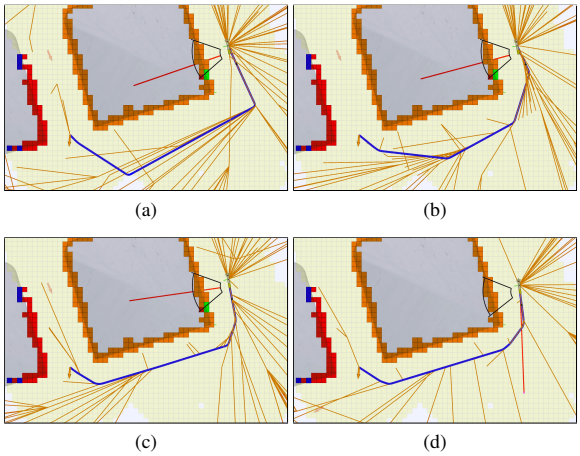

Fig. 4. The path generated using the online RRT* path planner converges fast to a solution that minimizes the path length while keeping a safe distance to the obstacles: (a) initial solution, (b) the solution improves but it is still inefficient, (c) the solution converges to the optimal solution, and (d) the solution has converged.

C. Trajectory Tracking (Act)

Finally, a line of sight (LOS) controller [22] is used to follow the generated path. The controller sends velocity commands to the vehicle in order to keep the robot trajectory as close as possible to the generated path. In our approach, we consider that external perturbations, such as water currents, are handled at the control stage, which is out of the scope of our research.

IV. MAP OPTIMIZATION CONTRIBUTIONS

A. Map Representation

In our previous work [1] the map was stored as a 2D array of integers, where each label was associated to a particular integer value. In this representation, cell types could be accessed and modified in constant time. However, this approach requires a lot of memory for large maps, and other operations such as range queries, k-nearest neighbors queries, or iteration over a specific label require traversing all the cells in the map.

In order to support those operations, which are really useful to efficiently generate viewpoints, we evaluated different possibilities:

- Store each cell type in a set, providing O(log(n)) access time, or even in a hash table, providing arguably O(1) access time. With this approach, memory requirements are reduced because there is no need to store unknown cells. However, range and nearest neighbor queries are not supported.
- Use a library such as Octomap [23]. This approach has the benefit of using a well established and tested library. However, it is designed to store a probability for each cell, supporting only three states (Unknown, Empty and

Occupied). Furthermore, although it would be possible to instantiate an Octomap for each label, range and nearest neighbor queries are not natively supported.
- Design our own map representation, based on the quadtree (a quadtree is a tree data structure in which each node has exactly four children [24]. See Figure 5) and octree data structures, as in the Octomap library, but supporting all the operations that are required to optimize the viewpoint generation. This is the option that we chose.

As a result, in this work we propose the use of multiple quadtrees, each containing only the cells of a particular label. By storing each label in a different quadtree, all cells that have a particular label can be obtained by simply traversing the corresponding tree. This also allows to perform range and k-nearest queries for a specific cell label.

Bearing all this in mind, our approach uses separate quadtrees for the following labels:

- Empty cells.
- Occupied cells.
- Sonar candidates.
- Camera candidates.
- Occupied cells that have not been viewed. When no camera target cells are present, these are the cells that generate camera viewpoints.

By isolating the sonar and camera candidates in the map generation stage we avoid having to recompute all target cells during the view planning stage. This is a key aspect for improving the performance of our VP algorithm.

Furthermore, the start-go-goal planner can use range or k-nearest queries to efficiently determine the cost of a particular state without having to explore the complete neighborhood of that state.

B. Map Generation

To acquire occupancy data from the environment we use a scanning profiling sonar. This type of sensors are able to provide 2D exteroceptive data by measuring the intensity of the acoustic return over time after an initial pulse is emitted. Usually the sensor reports a range, which internally is obtained by isolating and thresholding the strongest peak of the acoustic return. However, sometimes the acoustic return is too weak, and the sensor reports empty space even though an obstacle is within the maximum range of the sensor.

The common approach to incorporate range measurements into a grid map is to consider all cells prior to the detection as empty space and the cell corresponding to the detection as occupied. This approach usually works great, but when false negative measurements are present, it can lead to empty cells appearing behind occupied cells, creating inconsistent maps (Figure 6). This is a problem for our algorithm because viewpoints could then be generated in unreachable locations.

To solve this problem, in [1] we used a region growing approach to compute all the reachable cells from the robot position. This was performed at each planning iteration, so

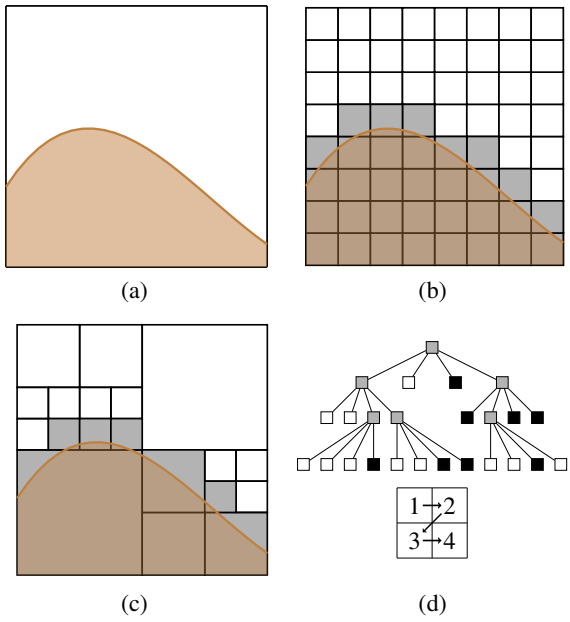

Fig. 5. Quadtree data structure example: (a) the structure being mapped, (b) a rasterized version of the structure, in which the space is represented using cells of the same size, (c) quadtree representation of the structure, in which the space is recursively subdivided, and (d) the corresponding tree.

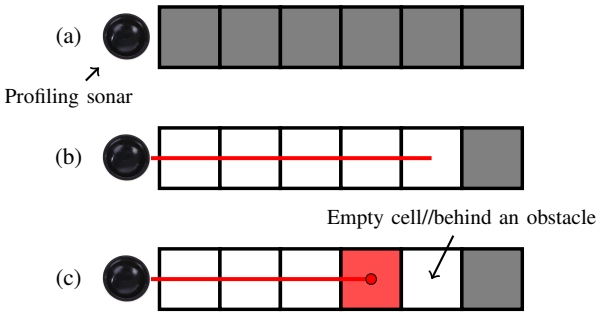

Fig. 6. Using the traditional approach, the map consistency is affected by false negative measurements: (a) initially the map contains unknown cells, (b) then a false negative measurement is incorporated, resulting in empty cells along all the beam until the maximum range of the sonar, and (c) finally a correct measurement is incorporated, generating an occupied cell but without deleting the empty cells behind it.

that all unreachable empty cells were reset to the unknown label. This approach had two drawbacks: it was highly inefficient and cells were only eliminated when completely isolated.

As an alternative, we evaluated the possibility of storing cell connectivity as a graph, and then use a fully dynamic connectivity algorithm. However, they are difficult to implement, and in fact we do not want to keep track of isolated empty cells: we just want to delete them as soon as we detect them.

In this paper we present a novel algorithm that is able to keep the coherency in the map and it is easy to implement, while being light on the computations.

The algorithm operates using the *hit and miss* approach [25], which uses the total amount of empty and occupied

detections of a cell to determine its state. After determining the proportion between them (Equation 1), a user defined threshold is used to classify the cell as empty or occupied (Equation 2 and lines 12 and 20 of Algorithm 1). From our experience, the hit and miss approach takes care of false positives just fine if a proper threshold is set, since it balances how many empty detections are required to compensate for an occupied detection.

$$\tau = \frac{\#occupied\ detections}{\#occupied\ detections + \#empty\ detections} \quad (1)$$

$$\text{label}(\tau) = \begin{cases} occupied & \text{if } \tau > threshold \\ empty & \text{otherwise} \end{cases} \quad (2)$$

Our contribution, however, lies in the fact that empty detections are only taken into account under certain conditions. Empty detection counters are initialized for each possible direction (north, south, east and west), and they store information about the direction from which the detection has been obtained. Furthermore, an incremental stamp is also stored for each cell, thus reflecting the order in which the state of each cell has changed. With this information at our disposal, the empty detections for a particular cell are computed according to Algorithm 2. Essentially, for each direction, the following conditions must be met in order to count their empty detections:

1) The neighbor in the direction that is being currently checked must be an empty cell. This condition has been imposed because it is impossible for a sonar beam to detect empty space behind occupied space.
2) The neighbor in the direction that is being currently checked must have changed its state before the evaluated cell. This means that the neighbor's stamp must be smaller. In other words, if the state of a cell c_2 is determined using the detections coming from another nearby empty cell c_1, the cell c_1 can not use empty detections coming from the cell c_2 when reevaluating its state in the future.

By setting the previous conditions, empty cells can only appear next to other empty cells, thus avoiding the formation of isolated empty regions in the map. The cell corresponding to the sonar position is always assigned the label *Empty* regardless of its counters (because it is impossible for the sensor to be placed on top of an obstacle), and this is what allows empty cells to appear in the map at the beginning of the map generation process.

During the map generation, the quadtree data structures are updated when the functions *setEmpty()*, *setOccupied()* or *setUnknown()* are called (lines 14, 22 and 25 of Algorithm 1). Inside these functions it is also determined if the evaluated cell is a range or camera candidate. In parallel, the state of all cells within the camera FOV is also updated in the corresponding quadtree.

This algorithm adds a constant computational cost per cell to check the map coherency. Then, the algorithm operates in linear time with respect to the amount of cells that have to be deleted.

Algorithm 1: updateMap

```
Input:
Beam origin, beam direction, range and maximum range.
Output:
Updated map.
1  begin
2      setEmpty(beam_origin_cell)
3      assignZeroStamp(beam_origin_cell)
4      foreach cell ∈ beam do
5          updateCounter(cell)
6          cells_to_check.append(cell)
7          foreach cell ∈ cells_to_check do
8              computeDetections(cell)
9              if dealing_with_empty_detection then
10                 if total_detections == 0 continue
11                 computeOccupancy(cell)
12                 if occupancy ≥ th continue
13                 if isEmpty(cell) continue
14                 setEmpty(cell)
15                 assignNewStamp(cell)
16                 cells_to_check.append(
                           nonempty_neighbors)
17             else
18                 if total_detections > 0 then
19                     computeOccupancy(cell)
20                     if occupancy < th continue
21                     if isOccupied(cell) continue
22                     setOccupied(cell)
23                 else
24                     if isUnknown(cell) continue
25                     setUnknown(cell)
26                 assignMaxStamp(cell)
27                 cells_to_check.append(
                           empty_neighbors)
```

Algorithm 2: computeDetections

```
Input:
Cell position.
Output:
empty_detections and total_detections.
1  begin
2      empty_detections = 0
3      foreach direction do
4          if isEmpty(neighbor_at_direction) and
             neighbor_at_direction.stamp ≤ cell.stamp
             then
5              empty_detections +=
                 cell.empty_detections_from_direction
6      total_detections =
         empty_detections + cell.occupied_detections
```

Finally, to compare our algorithm to the well-known method used in the Octomap library, Table I is presented.

TABLE I
QUALITATIVE COMPARISON BETWEEN OUR PRESENTED APPROACH AND THE OCTOMAP APPROACH

	Benefits	Drawbacks
Presented approach	The state of a cell not only depends on its measurements, but also on the state of its neighbors, providing coherency	Based on some heuristics and less on probabilities. More memory consumption and computations. Requires a threshold to be specified
Octomap	Less memory consumption, less computations and probability based	Uniform prior probability must be specified, and thresholds are also required. Each cell is considered as an independent measurement, uncorrelated to its neighbors

V. RESULTS

To validate our approach, experiments have been conducted in simulated and real environments using the Sparus II AUV (see Figure 1). This AUV is a compact torpedo-shaped robot with a reconfigurable payload area, in which the scanning profiling sonar and the cameras have been installed for the experiments presented in this work. It has partial hovering capabilities, meaning that *surge*, *heave* and *yaw* degrees of freedom (DOFs) are actuated, while *sway*, *roll* and *pitch* remain underactuated.

Most of the tests were carried out in a scenario located outside the harbor of St. Feliu de Guíxols (Figure 7), Girona, which consists of a series of breakwater concrete blocks. These blocks are designed to dissipate the force of the incoming waves, providing a challenging scenario for underwater robotics. The model of the complete harbor has also been used to simulate longer missions in different survey areas (Figure 10).

A. Simulated environments

The first experiment consisted in adding noise (false negatives) to the sonar data and compare the consistency of the generated map. Without the proposed algorithm (our previous approach) many cells are labeled as empty behind cells labeled as occupied, and this becomes a problem when viewpoints are generated in those areas. With the proposed algorithm those empty cells are eliminated and remain as unknown cells during the exploration. Figure 8 shows simulated results comparing both behaviors. An extra 10% of empty scans were added to simulate sonar noise.

The second experiment is related to the performance of the algorithm. The computational time of our previous approach and the proposed approach have been measured for different map sizes, and results are shown in Figure 9. Simulated missions were carried out in different parts of the harbor, as shown in Figure 10, producing 4 different maps of increasing size.

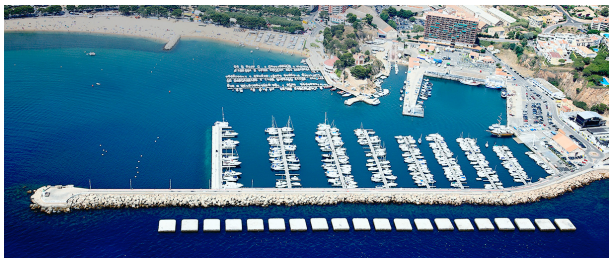

Fig. 7. Aerial view of the St. Feliu de Guíxols harbor. Breakwater concrete blocks appear at the bottom of the image.

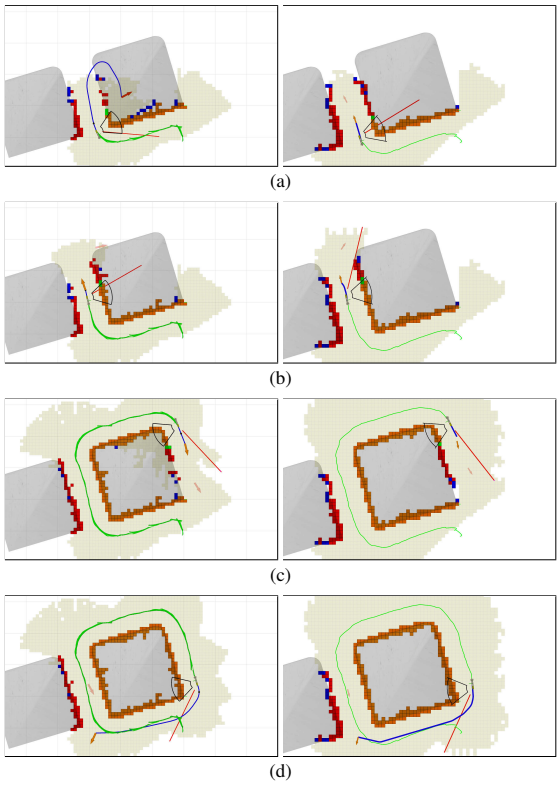

Fig. 8. Experiment showing the consistency of the generated map when noise is added. On the left, our previous approach, and on the right, the proposed approach: (a) some empty cells appear on top of the obstacle, (b) they disappear when they get unconnected from the rest of empty cells, (c) more empty cells appear, and (d) final reconstruction of the block.

Experimental results suggest that the computational time of our previous approach can be approximated to be proportional to the amount of cells in the map, which makes sense because many parts of the algorithm required complete traversal of all the cells in the map.

On the other hand, experimental results show that the proposed algorithm is able to compute the viewpoints almost in constant time. This is explained by the fact that the amount of operations performed in the view planning stage is proportional the the amount of occupied cells that have not been seen, which usually remain low during a mission.

This experiment is also important because it demonstrates that the vehicle successfully follows both concave and convex obstacles without colliding with the walls of the harbor.

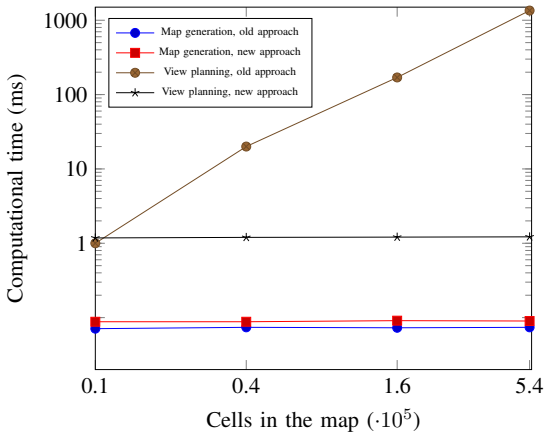

Fig. 9. Execution time comparison. The computational cost of the old approach is proportional to the amount of cells in the map, while in the proposed approach it is only proportional to the amount of cells that are occupied but not seen, which usually remains low.

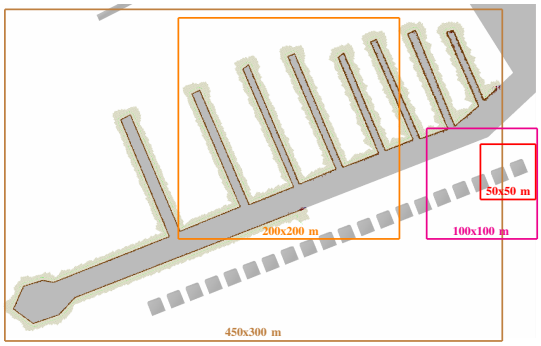

Fig. 10. To measure performance, different survey areas have been used. The map shown in the figure corresponds to the longest exploration (2.4 km), where the robot started outside of the harbor and entered to explore all the wharves. The cell size was set to 0.5x0.5 meters for all cases.

B. Real man-made environment

Real experiments were also conducted to validate our proposed algorithm. Figure 11 shows a mission where Sparus II AUV successfully mapped two breakwater blocks. The mission was performed at a depth of 1.75 meters and the robot navigated a total of 98 meters.

C. Real natural environment

Figure 13 shows a mission where Sparus II AUV successfully mapped a rock surrounded by water, located near the St. Feliu de Guíxols harbor (Figure 12). This natural environment is challenging due to its irregular shape and also because it is located next to the coast cliffs, providing narrow passages of difficult access.

This experiment was performed at a depth of 2.5 meters and the total robot displacement was 172 meters. It was performed in approximately 17 minutes.

During the execution of this mission, three unsynchronized GoPro Hero 4 Black edition cameras were recording images from the side of the vehicle. The cameras were attached at the front of the vehicle and oriented right, right-forward, and

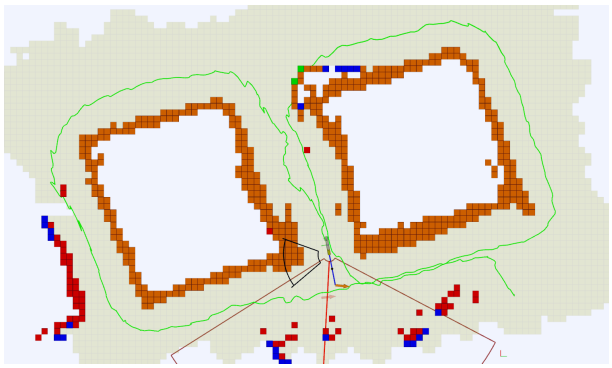

Fig. 11. Real experiment showing the inspection of two breakwater concrete blocks. The size of the blocks is approximately 12x12 meters. The inspection began in front of the block that appears on the right of the image.

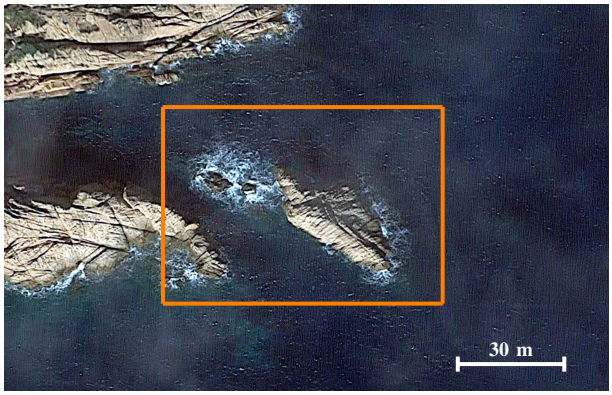

Fig. 12. Aerial top view of Punta del Molar. The explored rock has been highlighted inside the orange rectangle. The coast cliffs can be seen on the top and left sides of the image.

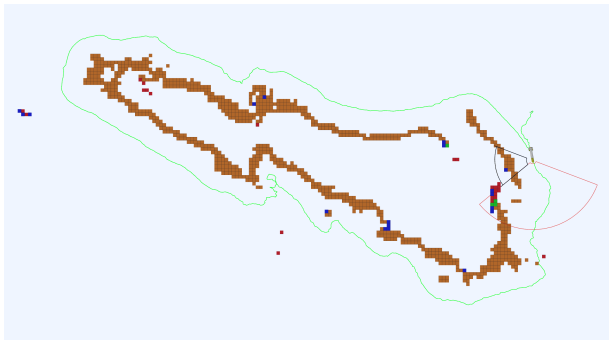

Fig. 13. Real experiment showing the inspection of a natural rock surrounded by water next to the harbor of St. Feliu de Guíxols. The rock is 60 meters long (please see Figure 12 for further reference). The inspection began approximately in the same place where the robot ended, and the robot explored the rock clockwise because the cameras were mounted on the right. Empty space cells are not represented.

right-backward to ensure that maximum 3D information was captured while maintaining spatial overlap.

With the acquired images, a reconstruction was made (Figure 14) using an optical 3D reconstruction procedure, as described in [26] (the reconstruction procedure is out of the scope of this work).

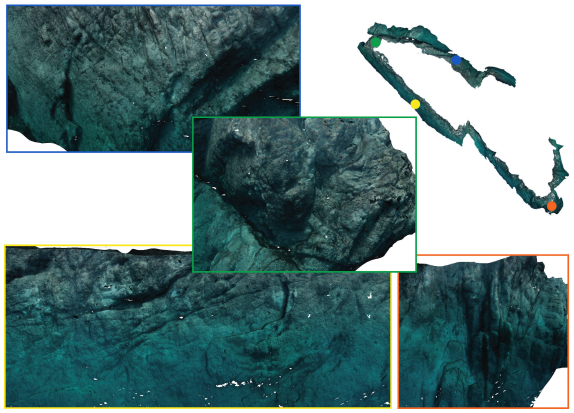

Fig. 14. Optical 3D reconstruction of the explored environment (top view with a few enlarged details).

VI. CONCLUSIONS AND FURTHER WORK

In this paper we have refined our view planning (VP) exploration algorithm and we have addressed two important problems: First, we have improved the map representation so that viewpoints can be generated more efficiently. This is particularly important in the underwater domain, as computational resources are scarce; Second, we have improved the robustness of the algorithm against sonar noise. Both improvements have allowed new and challenging experiments to be performed, not only in simulation but also in real man-made and natural scenarios, which present complex irregular shapes. Finally, we strongly believe our work can be interesting to many other people working in robotic exploration and view planning.

Further work will include going from a 2-dimensional (2D) map to a 3-dimensional (3D) map of the environment. The data acquisition sensor will probably have to be upgraded, but authors believe that the main ideas of the algorithm will remain valid. It also remains to be studied how to improve the algorithm performance when higher amounts of navigation drift are present.

REFERENCES

[1] E. Vidal, J. D. Hernández, K. Istenic, and M. Carreras, "Online View Planning for Inspecting Unexplored Underwater Structures," *IEEE Robotics and Automation Letters (RA-L)*, vol. 99, 2017.

[2] W. Scott, G. Roth, and J. F. Rivest, "View Planning for Automated 3D Object Reconstruction Inspection," *ACM Computing Surveys*, vol. 35, no. 1, pp. 64–96, 2003.

[3] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings - IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pp. 146–151, IEEE Comput. Soc. Press, 1997.

[4] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[5] A. Kim and R. M. Eustice, "Next-best-view visual {SLAM} for bounded-error area coverage," *IROS Workshop on Active Semantic Perception*, no. Mi, 2012.

[6] F. S. Hover, R. M. Eustice, A. Kim, B. J. Englot, H. Johannsson, M. Kaess, and J. J. Leonard, "Advanced Perception, Navigation and Planning for Autonomous In-Water Ship Hull Inspection," *The International Journal of Robotics Research*, vol. 31, no. 12, pp. 1445–1464, 2012.

[7] A. Bircher, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel, and R. Siegwart, "Structural Inspection Path Planning via Iterative Viewpoint Resampling with Application to Aerial Robotics," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6423–6430, 2015a.

[8] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding Horizon "Next Best View" Planner for 3D Exploration," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1462–1468, 2016.

[9] C. Papachristos, S. Khattak, and K. Alexis, "Uncertaintyaware Receding Horizon Exploration and Mapping using Aerial Robots," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4568–4575, 2017.

[10] D. P. Williams, F. Baralli, M. Micheli, and S. Vasoli, "Adaptive underwater sonar surveys in the presence of strong currents," *Proceedings - IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2016-June, pp. 2604–2611, 2016.

[11] E. Galceran, R. Campos, N. Palomeras, M. Carreras, and P. Ridao, "Coverage path planning with realtime replanning for inspection of 3D underwater structures," *Proceedings - IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6586–6591, 2014.

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

[13] C. Connolly, "The Determination of next best views," *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, pp. 432–435, 1985.

[14] H. González-Baños and E. Mao, "Planning robot motion strategies for efficient model construction," *Robotics Research*, vol. 19, pp. 345–352, 2000.

[15] J. I. Vasquez-Gomez, E. Lopez-Damian, and L. E. Sucar, "View planning for 3D object reconstruction," *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 4015–4020, 2009.

[16] J. I. Vasquez-Gomez, L. E. Sucar, and R. Murrieta-Cid, "View/state planning for three-dimensional object reconstruction under uncertainty," *Autonomous Robots*, vol. 41, no. 1, pp. 89–109, 2017.

[17] S. Isler, R. Sabzevari, J. Delmerico, and D. Scaramuzza, "An information gain formulation for active volumetric 3D reconstruction," *Proceedings - IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2016-June, pp. 3477–3484, 2016.

[18] 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.

[19] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[20] I. A. Sucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, pp. 72–82, dec 2012.

[21] J. D. Hernández, M. Moll, E. Vidal, M. Carreras, and L. E. Kavraki, "Planning Feasible and Safe Paths Online for Autonomous Underwater Vehicles in Unknown Environments," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[22] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, Ltd, 2011.

[23] 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, pp. 189–206, 2013.

[24] R. Sedgewick, *Algorithms, 2nd ed.* Addison-Wesley, 1988.

[25] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[26] J. D. Hernández, K. Istenič, N. Gracias, N. Palomeras, R. Campos, E. Vidal, R. García, and M. Carreras, "Autonomous Underwater Navigation and Optical Mapping in Unknown Natural Environments," *Sensors*, vol. 16, no. 8, p. 1174, 2016.