Online View Planning for Inspecting Unexplored Underwater Structures

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

***Abstract*—In this paper, we propose a method to automate the exploration of unknown underwater structures for autonomous underwater vehicles (AUVs). The proposed algorithm iteratively incorporates exteroceptive sensor data and replans the next-best-view (NBV) in order to fully map an underwater structure. This approach does not require prior environment information. However, a safe exploration depth and the exploration area (defined by a bounding box, parameterized by its size, location and resolution) must be provided by the user. The algorithm operates online by iteratively conducting the following three tasks: 1) Profiling sonar data is firstly incorporated into a 2-dimensional (2D) grid map, where voxels are labeled according to their state (a voxel can be labeled as empty, unseen, occluded, occplane, occupied or viewed). 2) Useful viewpoints to continue exploration are generated according to the map. 3) A safe path is generated to guide the robot towards the next viewpoint location. Two sensors are used in this approach: a scanning profiling sonar, which is used to build an occupancy map of the surroundings, and an optical camera, which acquires optical data of the scene. Finally, in order to demonstrate the feasibility of our approach we provide real-world results using the Sparus II AUV.**

***Index Terms*—Motion and Path Planning, Autonomous Vehicle Navigation, Marine Robotics.**

I. INTRODUCTION

AUTONOMOUS underwater vehicles (AUVs) are capable of conducting tasks where human intervention is limited to monitoring and selection of the operations to perform. Among all possible applications of these vehicles, inspection of underwater structures is of special interest. Underwater environments provide challenging scenarios, mainly due to technological difficulties related to exploration (i.e., lack of reliable localization and communication, autonomy limitations), but also due to water currents and complex underwater reliefs. These are problems not easy to deal with, and that is the reason why new technology to explore the ocean floor is under continuous development nowadays.

Usually, AUVs are used for underwater mapping in places where a simple 2-dimensional (2D) coverage path is sufficient to gather required data (i.e., low profile shipwreck exploration, [1] and [2]). In these cases, path planning techniques are not required. Instead, the mission is preplanned by the user, typically by defining a sequence of waypoints which are followed at a constant safe altitude.

However, there are other situations (i.e., ship hull inspections [3], explorations in confined underwater environments such as caves [4], and seamount mapping [5]) in which following a preplanned trajectory is unfeasible due to the requirement of perfect localization and complete knowledge about the environment. Most underwater mapping solutions for complex scenarios still require some form of prior knowledge about the area to be explored, such as an approximate map, in order to plan inspection paths. Most of the times, however, such prior knowledge is simply not at scientists' disposal.

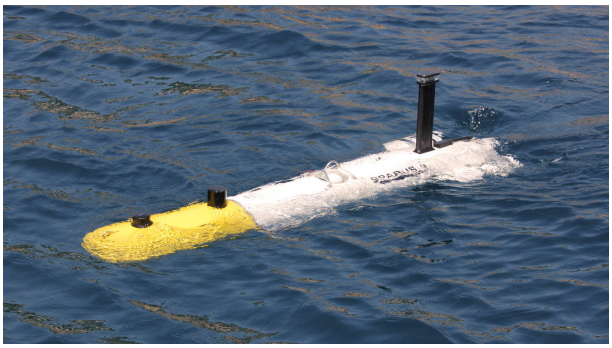


Fig. 1. Sparus II AUV, a torpedo-shaped underwater robot with partial hovering capabilities used to validate our approach.

Aiming to reduce the amount of prior information required for underwater mapping, this paper proposes a methodology for underwater exploration of unknown structures, which has been built upon view planning (VP) strategies. VP can be defined as the process of determining a suitable set of viewpoints and associated imaging parameters for a specified object reconstruction or inspection task. In our contribution, the robot is guided towards locations that provide new and useful information in order to continue the exploration. This is done by repeatedly selecting the next-best-view (NBV). The ability to make such decisions online enable AUVs to be used in complex scenarios with minimal human intervention. Our view planning proposal is specially conceived for underwater vehicles equipped with two different sensor technologies: acoustic profilers for detecting the objects and optical cameras for mapping them. Our methodology generates viewpoints that guarantee the coverage of the area with both sensors. Additionally to most VP algorithms, the proposed method does not restrict viewpoint locations to be around a known object position in a controlled environment, but generates viewpoints in a full unknown 2D space in which objects are detected.

Manuscript received: September, 10, 2016; Revised December, 7, 2016; Accepted February, 2, 2017.

This paper was recommended for publication by Editor Jonathan Roberts upon evaluation of the Associate Editor and Reviewers' comments. Work on this paper has been supported by the EXCELLABUST and ARCHROV Projects under the Grant agreements H2020-TWINN-2015, CSA, ID: 691980 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. eduard.vidalgarcia@udg.edu, juandhv@eia.udg.edu, klemen.istenic@udg.edu and marc.carreras@udg.edu.

Digital Object Identifier (DOI): see top of this page.

Finally, the article shows extensive experimental evaluation of the proposed algorithms using the Sparus II AUV in a real underwater environment.

The remainder of this paper is organized as follows. Section II introduces relevant related work. Section III describes the proposed online VP method for autonomous inspection of unexplored underwater structures. First, a description of the world representation is provided. Then, the process of incorporating sensor data to build the map is explained in detail. Finally, generation of the viewpoints and the path to achieve those viewpoints is addressed, providing insight into the controller used to follow the generated path. In Section IV results are brought out, showing the performance of the algorithm under real conditions using the Sparus II AUV (see Fig. 1). Finally, Section V presents conclusion remarks, sums up our approach contributions and discusses different alternatives for further work.

II. RELATED WORK

Some authors have proposed different techniques to map underwater structures, based on prior knowledge about the environment to various degrees. Some are related to VP while others just use coverage path planning (CPP).

Vasquez-Gomez et al. presented an algorithm that selects the NBV for a range camera to model 3-dimensional (3D) arbitrary objects [6]. Their algorithm is strongly inspired by the classic VP publication [7]. The algorithm uses a 3D grid map with labels to represent the environment. Only viewpoints located in a sphere established around the object are considered, and the quality of the viewpoints is assessed by using a quality function. The location and approximate size of the object must be known in advance.

VP strategies have been applied to ship hull inspection in [8] and [9]. For the sides of the hull, a preplanned trajectory in hull-relative coordinates is followed, and perception-driven navigation (PDN) is used to determine when a revisiting action must be taken to reduce pose uncertainty using simultaneous localization and mapping (SLAM). For the propellers and rudders, a rough map obtained from an initial preplanned mission is used to plan a second survey, from which a higher quality mesh is obtained. The path is planned using variants of the art gallery problem (AGP) and the traveling salesman problem (TSP).

Williams et al. [10] presented an algorithm for AUVs equipped with synthetic aperture sonar (SAS) sensors capable of adapting the survey heading (preplanned coverage type survey, without obstacle avoidance) based on the orientation of water currents. They also propose a VP algorithm for target reinspection which acquires multiple views of suspicious objects until a pre-specified (maximum) number of total object reinspections has been completed.

Galceran et al. [5] presented a 2.5-dimensional (2.5D) CPP approach for inspecting complex structures on the ocean floor. The algorithm plans a nominal coverage path using a map of the environment, which needs to be known in advance, and then the author proposes a method to take into consideration the uncertainty of the robot pose and sensor acquisition by adapting the path in real time to cope with the actual target structure perceived in situ.

In [11] a different approach is presented for 3D site modeling for an unmanned ground vehicle (UGV), which is composed of two stages. First, using a 2D map, which has to be known in advance, the algorithm finds a reduced number of viewpoints from which an initial map is generated. Then, a NBV approach is used to improve the final reconstruction.

Finally, Bircher et al. presented a VP approach for structural inspection using an unmanned aerial vehicle (UAV) [12]. The algorithm is similar to [9], but instead of trying to minimize the number of viewpoints, it re-samples new viewpoints that still satisfy the covering restrictions while improving the cost of the overall path. The 3D structure to be inspected needs to be known a priori, and is represented by a triangular mesh.

In comparison, our algorithm does not require a pre-existent map of the scene, the environment is modeled by a labeled 2D grid map and it relies upon VP techniques to determine the NBV.

III. VIEW PLANNING FOR UNDERWATER INSPECTION OF UNEXPLORED ENVIRONMENTS

The proposed algorithm is designed to enhance the capabilities of AUVs, allowing autonomous inspection of unexplored underwater scenes. The algorithm has been designed so that a prior map of the scene is not required. In order to inspect the area of interest, the vehicle is equipped with two sensors (see Fig. 2):

- *Scanning profiling sonar*, which acquires range measurements by mechanically scanning the surroundings in a 2D plane. It usually takes several seconds to complete a full scan.
- *Optical camera*, which is used to obtain the images of the observed scene. For planning purposes, only the estimation of the field of view (FOV) of the camera is used. Images are then later used as an input of an optical 3D reconstruction pipeline.

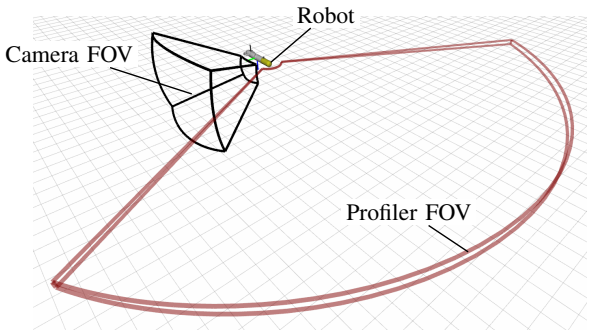


Fig. 2. 3D representation of the camera FOV (black frame), which in this case is pointing to the right of the vehicle, and the profiling sonar FOV (red frame).

The algorithm can be classified as 2D and grid based. Furthermore, it does not use sampling for the generation of the viewpoints, and at every iteration the algorithm computes the NBV to continue the exploration. Finally, the map generation algorithm is able to deal with outliers and erroneous measurements.

A. World Representation

We have chosen to use a 2D grid map to incrementally build a model of the environment. Figure 3 presents all possible labels and corresponding colors that can be assigned to each voxel in the map. This way of representing the environment is based on [6]. However, in this work a new label has been added to characterize those regions that have been observed by the optical camera.

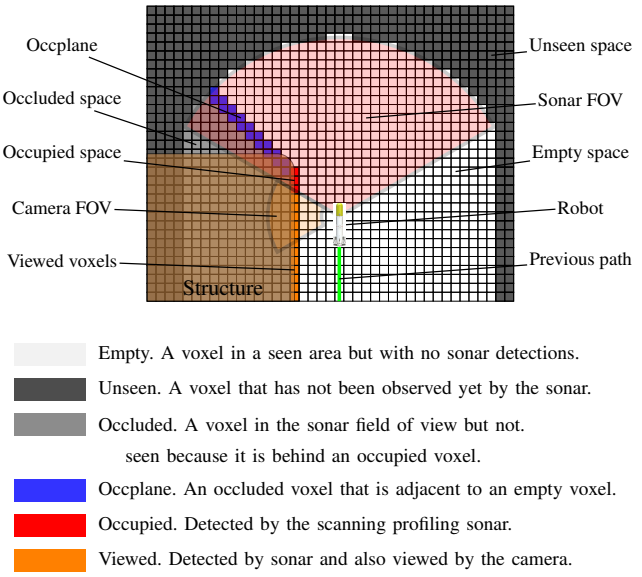


Fig. 3. Synthetic example of world representation and map generation: mapping a corner under the water. All labels are present. Camera and sonar FOVs are also represented. Robot is moving from the bottom to the top of the image.

Figure 3 shows a synthetic example of a map generated while the robot inspected an underwater structure. In this case, the structure resembles a concrete block corner (brown, translucent). The green path indicates the robot's previous trajectory. In this example, the robot is moving from the bottom to the top of the scene. White voxels represent space that has been perceived as empty (without collision) by the sonar. The sonar FOV is represented in translucent red color. Black voxels correspond to the space that has not yet been covered by the sonar FOV. In this example, the camera points to the left, and it is possible to see how all voxels that have been within the camera FOV (orange, translucent) have been labeled as viewed (orange). Occupied voxels in red denote areas perceived as occupied (with collision) by the sonar, but at the same time these areas have not been viewed by the camera yet. Finally, those voxels that have been inside the sonar FOV but, at the same time, have been also occluded by occupied and viewed voxels are marked as occluded voxels (gray). Of these, the ones neighboring at least one empty voxel are classified as occplane (blue).

The voxel size must be small enough so that the required map accuracy is achieved, but large enough so that sonar measurements are sufficient to completely determine their state without leaving gaps in the map. For our purposes, a voxel size of 0.5 meters has proved to work fine.

B. Map Generation

In order to generate a map like the one described in the previous section, basic filtering is first applied to the sonar beam measurements to avoid noise and outliers:

- Measurements close to minimum and maximum range are discarded, since they proved to be unreliable.
- Measurements close to the water surface are discarded to avoid reflections.
- Roll and pitch are tracked and data is only considered reliable when the robot is stable.

Then, given that our planning algorithm operates in 2D, 3D sonar data is projected to the specified mission depth to build the map. For each sonar measurement, it is required to update all the voxels that fall within the beam. A well known ray tracing algorithm has been used to traverse required voxels (Amanatides et al. [13]).

In order to determine the final label of each voxel in the map, the *voxel logic* shown in Fig. 4 is followed. It differs from [14] and [6] in that it incorporates new labels to represent additional information about what has been seen by the camera.

For each voxel in the map, two counters store the number of times that each voxel has been detected as empty or occupied by the sonar, and one flag indicates whether the voxel has been within the camera FOV or not.

First, empty counters are incremented for all voxels prior to the detection (if the measurement reports no detection, this happens for all voxels in the beam). After that, the occupied counter for the detected voxel is increased. Finally, the remaining voxels (voxels behind the detection) are marked in the map as occluded if both empty and occupied counters remain at zero. Periodically, camera readings are also taken into account by computing the voxels that lay within its FOV. The appropriate flag is updated for all voxels in this situation. In Fig. 5 a black frame on the left side of the robot represents an estimation of what the camera is observing.

Then, for each voxel in the ray its final map state is computed. If the voxel has been detected as empty or occupied at least once, it will be empty or occupied in the map. The proportion τ between occupied detections and total detections is computed as described in Eq. 1. If the proportion exceeds a user defined threshold, the voxel is labeled as occupied (Eq. 2). Otherwise, it is labeled as empty. A good experimental value for the threshold is 0.1, which means that 10% of occupied detections is enough to label a voxel as occupied. By taking into account every measurement for each voxel, the algorithm is able to cope with outliers and erroneous measurements.

$$\tau = \frac{\#occupied\ detections}{\#occupied\ detections + \#empty\ detections} \tag{1}$$

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

If the proportion τ determines that a voxel is occupied, then the camera information is used to determine if the voxel should be further labeled as viewed.

Finally, when a voxel changes from any state to empty or vice versa, neighbors must be reconsidered (there could be occplane voxels that might change their state).

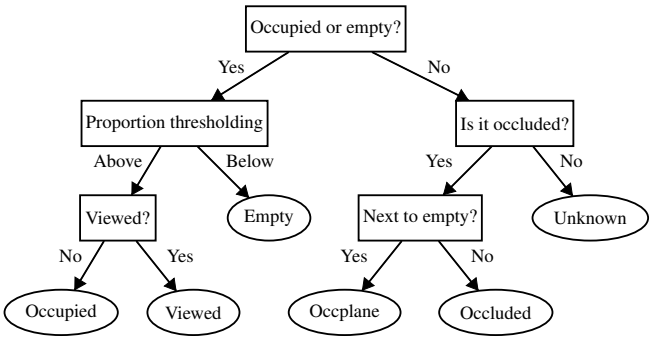


Fig. 4. Map generation algorithm. After following the algorithm, a voxel is classified and a label is obtained (leafs).

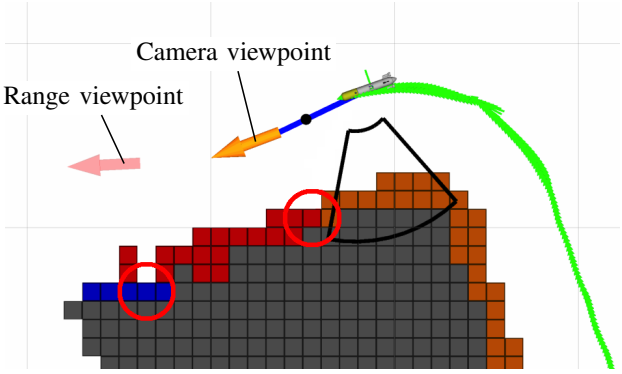


Fig. 5. Example of viewpoint generation. In this situation, the camera viewpoint was selected (solid orange arrow), while range viewpoint was discarded (light red arrow). The direction of the arrows corresponds to the desired vehicle orientation. Voxels that generated final viewpoints have been highlighted inside red circles. Previous vehicle positions and orientations are represented using small green arrows (jagged green path).

C. Viewpoint Generation

Once the sonar data has been incorporated into the map, viewpoints can be computed in order to guide the robot towards locations that can potentially provide new information. Two different kind of viewpoints are generated:

- *Range viewpoints*, which are designed to move the robot toward locations where sonar data could reveal new information about the scene.
- *Camera viewpoints*, which are generated so that new optical information about the scene can be obtained by the camera.

Range viewpoints are generated by firstly detecting occplane voxels adjacent to occupied voxels. Those voxels are important because they represent the boundary between 3 regions: what was perceived by the sonar, what could not yet be seen by the sonar because it was occluded, and what can be explored because it is next to empty space. Viewpoints are generated at a configurable distance from occplane voxels, along the perpendicular of the surface (see Section III-D for details). This type of viewpoints are represented with a red arrow in the visualizations (we use the RViz visualizer from the robot operating system (ROS)).

Camera viewpoints, on the other hand, are generated by firstly detecting occupied voxels adjacent to viewed voxels. Those voxels are important because they represent the boundary between the region that has been seen and the region that has not yet been seen by the camera. When the vehicle tries to map a new structure, which has been detected by the sonar but it has not been inside the camera FOV, camera viewpoints are generated for every occupied voxel, since we assume that it does not matter from where optical exploration begins. Camera viewpoints are also generated at a configurable distance along the perpendicular of the surface. This type of viewpoints are represented with an orange arrow in the visualizer.

Once the range and camera viewpoints have been calculated, it is necessary to select the next best viewpoint. Initially, the possibility of solving a RRT* planner query to determine the distance between the robot and each viewpoint was evaluated, but it was soon discarded because of its computational cost. Instead, the distance between the robot and each viewpoint (for both, range and camera viewpoints) is computed according to Eq.3:

$$\text{dist}(p,\theta,v,\gamma) = \sqrt{(p-v)\cdot(p-v)^T} \; + \tag{3}$$

$$D\cdot\left(\left|\frac{\text{atan2}((p-v)[1],(p-v)[0])}{\pi}\right| + \left|\frac{\text{wrap}(\theta-\gamma)}{\pi}\right|\right)$$

Where p corresponds to the robot position, θ is the robot orientation, v is the viewpoint position, γ is the viewpoint orientation and D is a user-defined penalizing distance. The function wrap() converts an angle to $(-\pi, +\pi]$.

The Euclidean distance is used, but is also combined with an additional heuristic to favor certain viewpoints according to the vehicle orientation (e.g., viewpoints in front of the robot and pointing forward are preferred). The best viewpoint is considered to be the closest one according to this distance.

In the visualizer, the selected viewpoint is displayed as a solid arrow, while the discarded viewpoint is displayed with a certain amount of transparency (translucent arrow). Figure 5 depicts an example of viewpoints generated during the exploration.

Finally, once the robot has started mapping some structure, the vehicle will not stop until it completes the exploration of that structure. This is accomplished by keeping track of structures in an object map (the same object label is assigned to contiguous occupied and viewed voxels). When the robot has completely mapped a structure, it continues by the closest structure that still has voxels to be explored. To increase robustness and avoid noise-related issues, elements with an area below a configurable threshold are ignored. The robot stops when no more viewpoints can be generated.

D. Computation of Surface Normal

In order to compute viewpoint locations it is necessary to determine the surface normal direction at the selected target

voxels. The approach used to compute the surface normal (see Fig. 6) is an adaptation of a basic technique described in [15]:

- Given a target occupied voxel at the surface of the structure, compute the center of gravity of the empty voxels in their surroundings. This has been implemented using a region growing strategy with limited depth of exploration.
- The surface normal direction is computed as the direction of the segment that joins the target voxel and the computed center of gravity.

This approach has proven to work well in our experiments even for large voxel sizes (voxel sizes of up to 0.5 meters).

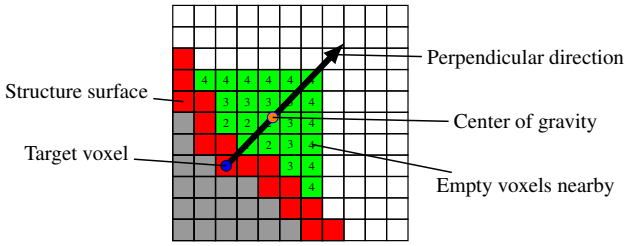


Fig. 6. Surface normal computation. Direction is given by the segment that joins the center of the target voxel (blue dot) with the center of gravity of its surrounding empty voxels (orange dot). Depth of exploration set to 4.

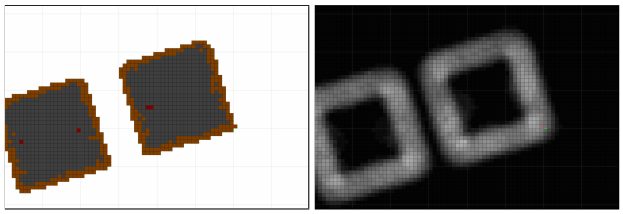
E. Path Generation

Once the next best viewpoint has been selected, the robot has to safely navigate to reach that viewpoint. In order to generate safe paths to guide the vehicle from its initial position to a particular viewpoint, we use the open motion planning library (OMPL), which is an adaptable path-planning framework [16].

Our implementation uses the asymptotic optimal rapidly-exploring random tree (RRT*) path planner, which is a sampling-based planner with asymptotic optimality. Paths are planned in a 2D space ($state \in \mathbb{R}^2$).

In OMPL, the desired behavior is achieved by selecting and adapting core components related to the chosen path planning algorithm. In order to tailor the behavior to our needs, the state validity checker, the sampler and the cost function have been implemented as follows:

- *State validity checker*. When the path planner wants to check whether a state is valid or not (i.e., equivalent to checking for collision) it uses the state validity checker. In our case, we have implemented the state validity checker so that a state is valid when all voxels inside the smallest possible circle containing the robot are not occupied (the state is valid in any given orientation).
- *Sampler*. Since RRT* is a sampling-based planner, we have to define how states are sampled in our configuration space (C-Space). The default sampler in OMPL randomly samples the C-Space, which is a valid approach in most cases. However, we have modified the sampler according to a strategy which consists on reusing the last best known solution (Hernández et al. [17]). This permits the sampler to take the previous valid path as an argument. Sampling the states of the previous best solution allows ensuring a path that is at least as good as the previous one.
- *Cost function*. An integral objective cost function has been used in order to compare how good a path is with respect to another path. This means that the cost of a path corresponds to the integral of a risk function along the path. In our case, risk associated with a path has been chosen to reflect how close to an obstacle a sample is (similar to the approach using path length with clearance in [17]). Therefore, the risk is high next to occupied voxels and vanishes as distance increases. This implies that we assume that water current disturbances are handled by the vehicle controllers, and that there are no small objects in the survey area that may be hard to detect, such as an anchor line or a fishing line. Instead of computing the risk of a sample each time the planner needs it, a risk map is computed when the map is updated (which happens fewer times per second, leading to a faster implementation). Figure 7 shows a visual representation of a risk map.



(a) (b)

Fig. 7. Risk map representation. Comparison between real map (a) and its corresponding risk map representation (b). Risk is higher next to obstacles. In the risk map representation, the highest risk is represented using white color and the lowest risk using black color.

F. Trajectory Tracking Controller

Once the path has been generated, the robot must follow it, minimizing the error between the path and the actual trajectory. For this purpose, a line of sight (LOS) controller has been implemented. The LOS strategy consists on guiding the robot towards an intermediate goal that is located some meters ahead in the path [18]. At each iteration:

- The robot position is projected into the path.
- An intermediate goal is computed some meters ahead of the projection point.
- Control commands are computed so that the robot moves towards this intermediate goal.

Finally, when the robot is close enough to the end of the path, it stops and orients according to the viewpoint direction.

G. Automatic Sonar Beam Orientation

The sonar beam scans 120 degrees in a sweeping movement along the vertical axis. However, instead of pointing always to the front, the beam is dynamically reoriented towards the region of the map that has to be scanned (a restriction is imposed so that the sweeping movement always covers the front of the robot to aid in obstacle avoidance, so the maximum allowed deviation is half of the sonar FOV). Automatic

orientation of the sonar beam allows the robot to perform only the control actions that are necessary to follow the trajectory, avoiding turning maneuvers for exploration purposes only. This is particularly useful when the scene contains tight corners because there is no need for the robot to turn around to inspect what is behind them.

IV. RESULTS

In order to validate the proposed algorithm, in this section we present real-world results using the Sparus II AUV.

A. Experimental Setup

Sparus II (Figure 1) is an autonomous underwater vehicle (AUV) developed recently at the Underwater Robotics Research Center (CIRS) [19]. It is a versatile robot, compact, but with enough space to be reconfigured with different sensors. Sparus II AUV has been used as the experimental platform throughout this paper.

B. Survey Areas

Two different survey areas have been used to validate our approach. Both areas are located in the harbor of St. Feliu de Guíxols, Girona. Figure 8 shows a satellite view:

- The *harbor area* has been used to do initial tests and a simple survey using the real robot.
- The *blocks area* is composed by a sequence of concrete blocks, which are used as breakwaters, and they are designed to dissipate the force of incoming waves by allowing water to flow around rather than against them. For this reason, it is evident that they provide a challenging scenario for underwater robotics because of the waves and currents. Furthermore, this scenario has also been used in prior relevant studies (such as [17] and [20]) to test different path planning solutions.

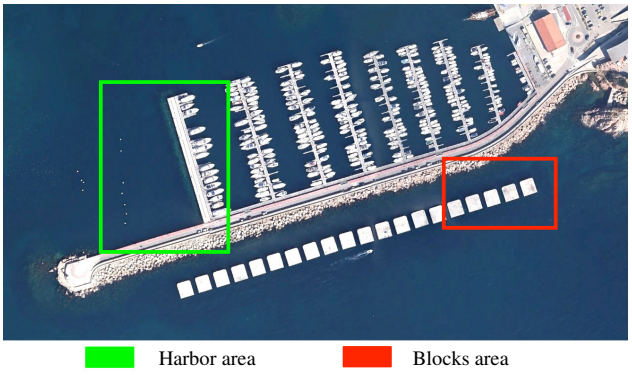


Fig. 8. St. Feliu harbor. Survey areas. Initial tests with the real robot have been performed inside the harbor. Outside the harbor, breakwater blocks provide a challenging scenario.

C. Real-world Experiments

All experiments have been performed at a constant depth of exploration of 1.5 meters and at a maximum speed of 0.4 m/s.

1) Harbor Environment: In this survey the robot successfully mapped two walls of the harbor (Figure 9). Due to the fact that it replans online the NBV taking into account what the sensors are perceiving, the algorithm was able to overcome the navigation drift. The estimation regarding the quality of the generated views predicts that 86% of the views would have been taken within 5 degrees of the surface normal. This experiment was executed in 314 seconds, and the robot traveled 90 meters. Because of the viewpoint generation strategy, and also because of the risk cost map, the robot does not navigate neither too close nor too far from the wall.

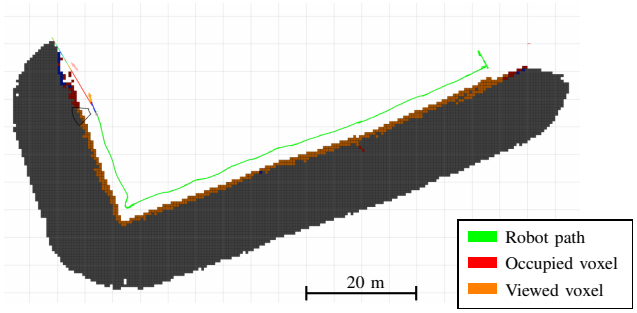


Fig. 9. Real survey in the harbor environment (see Fig. 8. Here the figure has been rotated 90 degrees). Successful exploration of two walls of the harbor. Navigation drift can be perceived but caused no problems. The robot trajectory started on the right of the image.

2) Blocks Environment: Two experiments were run: in one survey the camera was considered to be pointing to the left, but no cameras were actually mounted on the vehicle, and, in the other survey, the camera setup was pointing to the right. Figure 10 shows a survey on the blocks environment with the camera pointing to the left. The robot successfully mapped four concrete blocks. The planning algorithm estimates that about 80% of the camera images would have been taken within less than 5 degrees from the perpendicular direction. This experiment took 1364 seconds, and the total robot displacement was 326 meters.

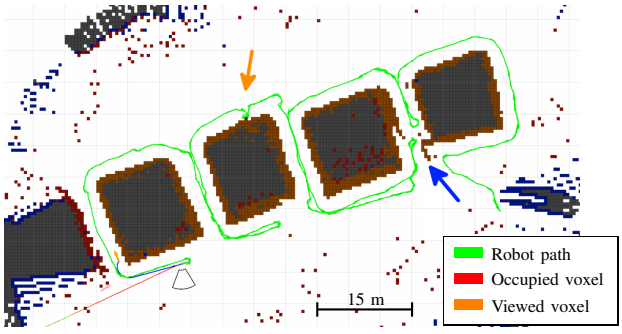


Fig. 10. Real survey in the blocks environment. Camera is pointing to the left. Navigation drift becomes noticeable. First block to be inspected appears on the right of the figure. Large arrows indicate interesting behaviors.

In the first experiment (see Fig. 10. This experiment is included because it shows what happens when there is a lot of navigation drift and noise in the sensors), the robot started mapping the first concrete block (on the right of the figure)

almost from the corridor, between first and second block. Because of that, the robot finished mapping the first block while still moving between the blocks and, in order to continue mapping the second block, a complete 180 degrees turn was performed inside the corridor (blue arrow in Fig. 10).

Since the robot started mapping the second block inside the corridor, to fully map the second block the robot had to go inside the same corridor coming from the other side. Furthermore, because of the navigation drift, the map did not perfectly match (also blue arrow in Fig. 10).

While mapping the third block, the robot went significantly close to the wall. A gap in the map caused the planner to plan a path that went against the wall. This was caused by noisy detections in the scanning profiling sonar, and improved filtering may have helped to avoid this situation. At the end, sensor measurements closed the gap and the robot continued (orange arrow in Fig. 10).

Figure 11 shows a survey on the blocks environment with the camera setup pointing to the right. In this experiment the robot mapped 8 blocks turning to the right, in a spiral motion. Figure 12 shows the robot while performing the autonomous mission. This experiment took 2930 seconds, and the total robot displacement was 669 meters. About 84% of the camera images were taken within less than 5 degrees from the perpendicular direction. The fact that the robot successfully mapped 8 consecutive blocks without colliding or without deviating from the expected behavior demonstrates the reliability of the proposed algorithm.

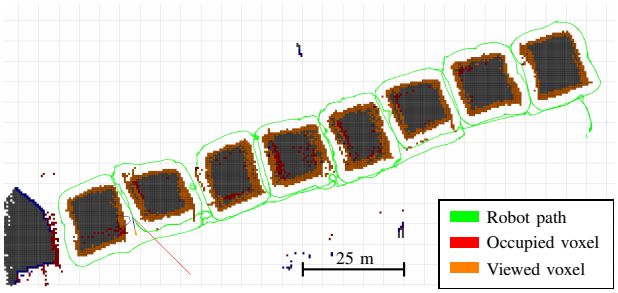


Fig. 11. Real survey in the blocks environment. Camera is pointing to the right. Navigation drift is also noticeable. First block to be inspected appears on the right of the figure.

Fig. 12. Sparus II AUV performing an autonomous mission in the blocks environment.

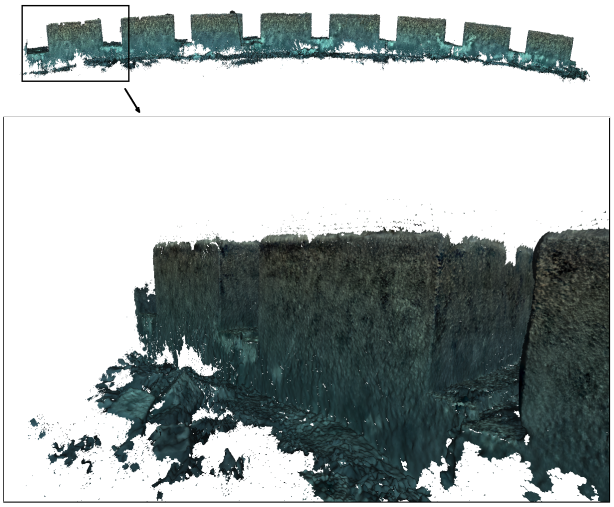


Fig. 13. Scene reconstruction using optical data. The image on the top corresponds to the side view of the 8 blocks. The bending in the vertical axis (depth) is caused by the reconstruction procedure.

In this experiment the robot was additionally equipped with a set of three unsynchronized GoPro Hero 4 Black edition cameras (GoPro, San Mateo, CA, United States). The cameras were positioned at the front of the vehicle and oriented right, right-down and forward-right-down to ensure the highest possible coverage, while still maintaining the ability to perform feature matching between images taken from different perspectives.

In order to demonstrate that the planning algorithm ensured the acquisition of optical data of the complete scene, acquired images were subsequently used as an input of an optical 3D reconstruction procedure, as described in [21] (the optical reconstruction procedure is out of the scope of this work), to obtain a final model of the observed environment (Figure 13).

V. CONCLUSIONS AND FURTHER WORK

In this paper, a new online view planning (VP) method to inspect unexplored underwater structures has been developed. The proposed algorithm is designed to work in an autonomous underwater vehicle (AUV), providing autonomous capabilities for underwater inspection of unknown structures. The algorithm does not require a prior map of the scene. By generating the next-best-view (NBV) at each iteration, the robot is guided towards locations that provide useful information for continuing the exploration. Several simulations and real-world experiments show the algorithm performing as expected.

The algorithm has proven to work even when a limited amount of navigation drift is present. However, for larger scenarios, where the navigation drift might be larger, the generated map could present some inaccuracies, thus leading to a longer than expected inspection survey. In those cases, using simultaneous localization and mapping (SLAM) techniques is suggested to reduce navigation drift to lower levels, enabling again the use of the presented algorithm.

Further work includes switching from 2-dimensional (2D) to 3-dimensional (3D) planning. The possibility to use an octree data structure will be evaluated, and efficiency of map and viewpoint generation will be improved.

REFERENCES

[1] N. Gracias, P. Ridao, R. Garcia, J. Escartin, M. L'Hour, F. Cibecchini, R. Campos, M. Carreras, D. Ribas, N. Palomeras, L. Magi, A. Palomer, T. Nicosevici, R. Prados, R. Hegedus, L. Neumann, F. de Filippo, and A. Mallios, "Mapping the Moon: Using a lightweight AUV to survey the site of the 17th century ship 'La Lune'," *OCEANS 2013 MTS/IEEE Bergen: The Challenges of the Northern Dimension*, 2013.

[2] B. Bingham, B. Foley, H. Singh, and R. Camilli, "Robotic Tools for Deep Water Archaeology: Surveying an Ancient Shipwreck with an Autonomous Underwater Vehicle," *J. Field Robotics*, vol. 27, no. 6, pp. 702–717, 2010.

[3] G. A. Hollinger, B. Englot, F. S. Hover, U. Mitra, and G. S. Sukhatme, "Active planning for underwater inspection and the benefit of adaptivity," *The International Journal of Robotics Research*, vol. 32, no. 1, pp. 3–18, 2012a.

[4] A. Mallios, P. Ridao, D. Ribas, M. Carreras, and R. Camilli, "Toward autonomous exploration in confined underwater environments," *Journal of Field Robotics*, vol. 7, nov 2015.

[5] 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*, pp. 6586–6591, 2014.

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

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

[8] 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.

[9] 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," *Intl. J. of Robotics Research*, vol. 31, no. 12, pp. 1445–1464, 2012.

[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*, vol. 2016-June, pp. 2604–2611, 2016.

[11] 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*, pp. 417–422, 2007a.

[12] 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," *International Conference on Robotics and Automation*, pp. 6423–6430, 2015a.

[13] J. Amanatides and A. Woo, "A Fast Voxel Traversal Algorithm for Ray Tracing," *Eurographics*, vol. 87, no. 3, pp. 3–10, 1987.

[14] 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.

[15] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. Pearson, 2007.

[16] 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.

[17] 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/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

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

[19] M. Carreras, C. Candela, D. Ribas, A. Mallios, L. Magí, E. Vidal, N. Palomeras, and P. Ridao, "SPARUS II, design of a lightweight hovering AUV," in *International Workshop on Marine Technology*, 2013.

[20] J. D. Hern, E. Vidal, G. Vallicrosa, E. Galceran, and M. Carreras, "Online Path Planning for Autonomous Underwater Vehicles in Unknown Environments," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1152–1157, 2015.

[21] 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.