Online Path Planning for Autonomous Underwater Vehicles in Unknown Environments

Juan David Hernández, Eduard Vidal, Guillem Vallicrosa, Enric Galceran, and Marc Carreras

***Abstract*— We present a framework for planning collision-free paths online for autonomous underwater vehicles (AUVs) in unknown environments. It is composed of three main modules (mapping, planning and mission handler) that incrementally explore the environment while solving start-to-goal queries. We use an octree-based representation of the environment and we extend the optimal rapidly-exploring random tree (RRT*) using concepts of anytime algorithms and lazy collision evaluation, thus including the capability to replan paths according to nearby obstacles perceived during the execution of the mission. To validate our approach, we plan paths for the SPARUS-II AUV, a torpedo-shaped vehicle performing autonomous missions in a 2-dimensional workspace. We demonstrate its feasibility with the SPARUS-II AUV in both simulation and real-world in-water trials.**

I. INTRODUCTION

Technological developments and trends in sensors, processors and actuators for autonomous underwater vehicles (AUVs) have fostered new potential applications, especially those devoted to imaging and inspecting different kinds of structures such as in-water ship hulls [1], complex structures on the sea floor [2] or confined natural structures (*e.g.,* underwater caves) [3]. In these scenarios, AUVs must operate in unknown, and potentially cluttered and dynamic environments, and therefore are more exposed to collisions. Furthermore, drift effects on the position estimated by navigation systems affect AUVs conducting autonomous missions in an underwater milieu. Dealing with such constraints requires a path planner with online capabilities that contributes to overcome global position inaccuracy, especially when navigating in close proximity to nearby obstacles.

Little research has addressed motion planning for AUVs, especially for the case of online motion planning in unknown environments. Petillot *et al.* [4] presented an initial approach for online obstacle avoidance and path planning for underwater vehicles, by using a real-world dataset of acoustic images obtained by a remotely operated vehicle (ROV) equipped with multi-beam forward looking sonar. They demonstrated the validity of their framework by guiding a simulated model of a ROV based on dataset information. However, online mapping and planning capacity was not proven. Maki *et al.* [5] proposed an online path planning method that uses landmarks to guide the vehicle, though it does not permit replanning and results were obtained in a controlled environment (*i.e.,* in a water tank).

This work was supported by the MORPH EU FP7-Project under the Grant agreement FP7-ICT-2011-7-288704 and partially supported by the Colombian Government through its Predoctoral Grant Program offered by Colciencias.

J.D. Hernández, E. Vidal, G. Vallicrosa, and M. Carreras are with the Underwater Robotics Research Center (CIRS), University of Girona, Girona, Spain. juandhv@eia.udg.edu, rc4559@gmail.com, gvallicrosa@eia.udg.edu, marc.carreras@udg.edu. E. Galceran is with the Department of Naval Architecture and Marine Engineering, University of Michigan, Ann Arbor, MI 48109, USA. egalcera@umich.edu

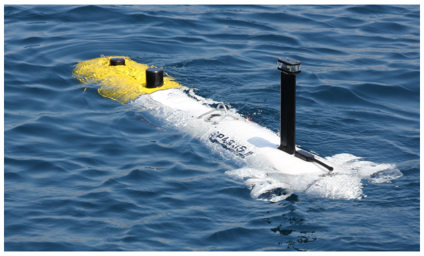


Fig. 1: SPARUS-II, a torpedo-shaped and non-holonomic AUV used to validate our approach for planning collision-free paths online in an unown environment.

Aiming to cope with the requirements of the aforementioned scenarios, this paper presents a framework for solving start-to-goal queries in unknown environments online for AUVs. The framework is composed of three main modules: 1) a *mapping* module that incrementally builds an occupancy map of the environment using on-board perception sensors; 2) a *planning* module that generates collision-free paths online; and 3) a *mission handler* module that works as a high-level coordinator of the planner and the AUV controllers.

The main contributions of this paper are 1) the combination of ideas from lazy collision evaluation [6] and anytime path planning algorithms [7], [8], [9] to extend the RRT* method [10] for planning collision-free paths online over an octree-based and incrementally built representation of the environment, and 2) the experimental evaluation of the resulting planning framework using the SPARUS-II AUV in a real-world setting. Results demonstrate the suitability of our approach for the aforementioned applications.

II. FRAMEWORK FOR PATH PLANNING ONLINE FOR AUVs

This section details our approach to solve start-to-goal queries for an AUV in an unknown environment. Figure 2 presents the proposed framework with its three main modules, and indicates its inputs and outputs.

A. Mission Handler

The *mission handler* controls the general flow of the proposed path planning framework. To ensure that the vehicle is prepared for solving and conducting a task, this module

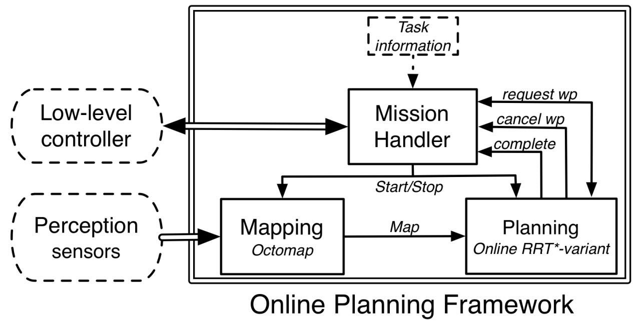


Fig. 2: Framework for planning online and its main modules.

communicates with other vehicle's functional modules and verifies that navigation data is being received, perception sensors are providing valid data and vehicle's controllers are not conducting any low-level safety maneuver. When the mission has started, a bidirectional communication with the *planning* module is established. The *mission handler* requests waypoints, when required, and receives and adapts them for the vehicle's low-level controller. Additionally, the *planning* module can notify the *mission handler* to cancel ongoing waypoint requests.

B. Mapping

The *mapping* module incrementally builds a representation of the environment using information received from different perception sensors, such as multibeam or mechanically scanned profiling sonars, echosounders, etc. Such sensors provide range information about nearby obstacles and, combined with the vehicle navigation (position and orientation), defines the free and occupied space with respect an inertial coordinate frame. To process this information, we use an octree-based representation, named Octomap [11], which is a framework for modeling volumetric information with three main characteristics. The first of which is the probabilistic state representation that not only updates map information but also protects it from noisy measurements, *i.e.,* a position state considers previous information and calculates its new value according to probabilistic functions. The second is the capacity of representing unexplored areas, which can be relevant for guiding exploration in unknown environments. Finally, Octomap offers an efficient way to model volumetric information, including its ability to enlarge or extend the map as demanded. For all this, Octomap results in an efficient method for collision checking purposes in a sampling-based algorithm, such as our RRT* variant.

C. Planning

The *planning* module receives a query to be solved, which is specified as a start and goal vehicle configuration, as well as additional planning parameters, such as the available computing time, minimum distance to the goal, and boundaries of the workspace. This module contains a modified version of the RRT* [10], which, as any other RRT-variant, has as a main characteristic the rapid and efficient exploration of the configuration space (C-Space), but also includes the asymptotic optimality property. Finally, our modified version of the RRT* incorporates concepts of *anytime* algorithms and *lazy collision evaluation* that, together with its incremental nature, make it suitable for online (re)planning applications. as described in the following section.

III. ANYTIME AND LAZY COLLISION EVALUATION FOR ONLINE PATH PLANNING

Path planning consists in finding a collision-free path from a start configuration to a goal configuration in the C-Space, which is the space of possible configurations of the robot. Sampling-based algorithms [12] have proved effective in problems involving high-dimensional configuration spaces, motion constraints and online computation requirements. This section presents the proposed RRT* variant and two concepts that have been used to extend it for solving online path planning tasks.

A. Anytime Approach for Replanning Online

The RRT* is itself an anytime algorithm. Karaman *et al.* [9] improved and formalized such extension though. Just as any other rapidly-exploring random tree (RRT) variant, our version is mainly composed by two procedures, `build` and `extend`. An additional procedure creates the tree rooted at the start configuration, and controls the execution according to other parameters such as the period of time (T_{build}) for each callback of the `build` procedure and the computation time (t_{comp}) that this procedure has for extending the tree.

Algorithm 1 presents the *build* procedure, which samples configurations uniformly distributed (line 4) to explore the C-Space by expanding the tree towards them (line 5, see Algorithm 2). There are two main differences with respect to other RRT variants. The first of them is presented in line 2, where `updateTree` is called. Before starting sampling, this procedure traverses the RRT with a Depth-first Search (DFS) algorithm to verify if any part of the tree (nodes and edges) is under collision. If a collision is detected in any node, its corresponding subtree (*i.e.,* the tree rooted at the node under collision) will be discarded. However, if the root of the RRT is under collision or this procedure identifies that moving from current vehicle's configuration to the root is not possible, it informs the *mission handler* to cancel the current point and starts over the planning from the current configuration. This occurs because at any moment of the mission execution, the tree root denotes the configuration that the vehicle is moving to, as explained below.

The second difference in Algorithm 1 is its behavior in an *anytime* fashion. If an expansion of the RRT results in a new configuration that meets the pre-established minimum distance to the goal (line 7), it is added to a list of possible solutions (line 8). After expanding the tree, if the *mission handler* has made a request and there exists a solution (line 10), the planner selects the best stored solution (line 11), *i.e.,* the one with the minimum associated cost, sends the child configuration (node) of the root to the *mission handler* (line 13), and prunes the RRT by establishing that node as

Algorithm 1: buildRRT(T)

```
Input:
T: configurations tree (RRT).
1  begin
2      updateTree()
3      while not $stop\_condition$ do
4          $q_{rand}$ ←sampleConf()
5          $result, q_{new}$ ←extendRRT($T, q_{rand}$)
6          if $result \neq TRAPPED$ then
7              if dist($q_{new}, q_{goal}$) $< \epsilon_{goal}$ then
8                  addSolution($q_{new}$)
9                  $solution\_found \leftarrow true$
10     if $solution\_found$ and $wp\_req$ then
11         $result\_path$ ←getBestSolution()
12         $new\_root \leftarrow result\_path[1]$
13         sendWaypoint($new\_root$)
14         pruneTree($new\_root$)
15 end
```

Algorithm 2: extendRRT(T, q_{rand})

```
Input:
T: configurations tree (RRT).
$q_{rand}$: state which RRT will be attempted to extend to.
Output:
Result after attempting extension.
$q_{new}$: New configuration if succeeded.
1  begin
2      $q_{near}$ ←findNearestNeighbor($T, q_{rand}$)
3      $u_{near\_rand}$ ←findInput($T, q_{near}, q_{rand}$)
4      $q_{new}, collision$ ←calcNewConf($q_{near}, u_{near\_rand}$)
5      if not $collision$ then
6          addNewNode($T, q_{new}$)
7          $Q_{near}$ ←findNearestNeighbors($T, q_{new}$)
8          $q_{min\_cost}$ ←findMinCost($T, Q_{near}, q_{new}$)
9          addNewEdge($T, q_{min\_cost}, q_{new}$)
10         reconnectNearNeighbors($T, Q_{near}, q_{new}$)
11         return ADVANCED
12     else
13         return TRAPPED
14 end
```

the new root (line 14). During the *pruning* process, nodes and edges connected to the initial root (excepting the new root) are discarded.

Algorithm 2 presents the standard procedure of expansion of an RRT. It receives a random configuration to which the RRT will attempt to expand to. `findInput` has been included explicitly in line 3 to generalize the procedure when differential constraints are included, if they are not, as in a geometrical case, the connection between configurations is a straight line segment. Finally, we include the connection and reconnection of nodes by comparing nodes costs, as proposed with the RRT* [10] (lines 7-10). This procedure returns a confirmation to indicate if expansion was succeeded (line 11) or not and (line 13), in either case.

B. Lazy Collision Evaluation for Replanning Online

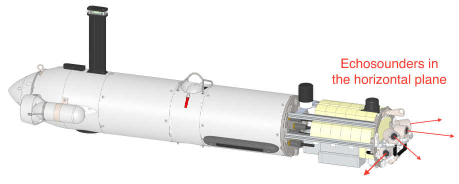
In general, the range of perception sensors is limited so that the workspace information improves as the vehicle moves through it. In our framework, with a sampling-based algorithm such as the RRT variant we present, most of the configurations used to expand and explore the C-Space are located in undiscovered regions of the workspace. With Octomap, we can verify in advance if a configuration is located in an explored area or not, thus avoiding unnecessary collision routine callbacks. When a configuration, sampled or resulting from an expansion, is out of known or explored area, the planner assumes it as valid. As the vehicle moves through the workspace and explores the area, such parts of the tree are verified and discarded if found under collision. This approach is based on the *Lazy Collision Evaluation* concept introduced by Bohlin *et al.* [6] for sampling-based planning algorithms.

IV. RESULTS

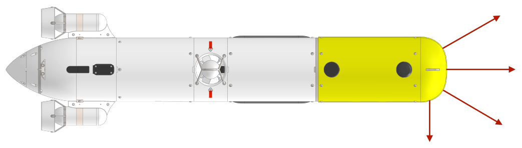
To evaluate our path planning framework, we used the SPARUS-II AUV (see Fig. 1), a torpedo-shaped vehicle with hovering capabilities, rated for depths up to $200m$. The robot has three thrusters (two horizontal and one vertical) and can be actuated in surge, heave and yaw degrees of freedom (DOF). The vehicle is equipped with a navigation sensor suite including a pressure sensor, a doppler velocity log (DVL), an inertial measurement unit (IMU) and a GPS to receive fixes while at surface. To perceive the environment, a set of five echosounders are located within the vehicle payload (front) area (see Fig. 3a). Four of them are in the horizontal plane, three are separated by 45^o, with the central one looking forward and parallel to the vehicle's direction of motion, while the fourth one is perpendicular to the central one (see Fig. 3b).

In order to evaluate the effectiveness of our approach, we used the harbour of Sant Feliu de Guíxols in Catalonia (Spain) as test scenario. Experiments were conducted in the external and open area of the harbour, in a breakwater structure (marked with a red ellipse as per Fig. 4a) that is composed of a series of concrete blocks of $14.5m$ long and $12m$ width, separated by a four-meter gap with an average depth of $7m$.

In this scenario, the SPARUS-II AUV had to move amidst the concrete blocks without any previous knowledge of their location. All queries have been defined to conduct missions with a constant depth, since most of perception sensors (echosounders) are located to cover the horizontal plane, thus the motion is restricted to a 2-dimensional (2D) task. We tested our framework in simulation and in the real world scenario. Results of these experiments are reported next.

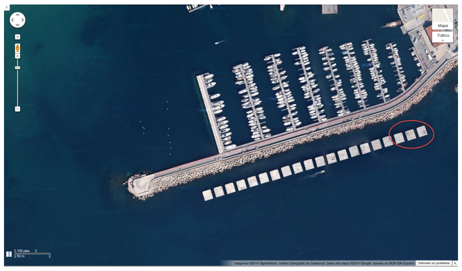


(a)



(b)

Fig. 3: Perception sensors configuration. (a) visible payload area in front of the vehicle. (b) top view, echosounders beams direction in the horizontal plane.



(a)

(b)

Fig. 4: Experiments scenario. (a) harbor of Sant Feliu de Guíxols in Catalonia, Spain, where a breakwater structure composed of concrete blocks is demarcated. (b) SPARUS-II AUV submerged and conducting autonomously a start-to-goal query moving amidst a series of concrete blocks.

A. Simulation Results

SPARUS-II, as well as the other AUVs developed at the underwater robotics research center (CIRS)[1], are controlled through the component oriented layer-based architecture for autonomy (COLA2) [13], a control architecture that is completely integrated with the robot operating system (ROS). Besides operating aboard real robots, COLA2 can interact with the underwater simulator (UWSim) [14], which can import 3D environment models and simulate the vehicle's sensors and dynamics with high fidelity. We used UWSim with a model of our target environment to do tests of our planning approach before conducting real-world in-water trials. We make use also of the open motion planning library (OMPL) that offers a convenient framework that can be adapted to specific planning problems [15].

We defined different start-to-goal queries and solved them using our framework. In all cases, start and goal were located on each side of the breakwater structure, in such a way that the only feasible solution requires to navigate amidst the obstacles (concrete blocks in the selected scenario). We defined $T_{build} = 2s$ as the period for the main iterative process. In simulation, we have observed that $t_{comp} = 1.0s$ is enough time for the *planning* module to find at least one solution, including the worst scenario, when a collision in the root of the tree is detected, and the framework has to replan and start over, discarding the existent tree. This means that in such case, the vehicle will not need more than one complete period ($2s$) to find its next waypoint. Figure 5 presents a simulation of SPARUS-II executing one of such paths.

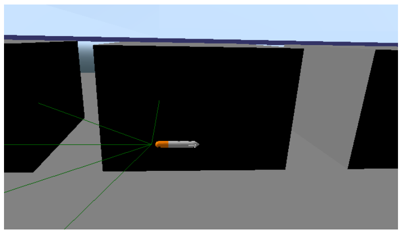
B. Real-world Results

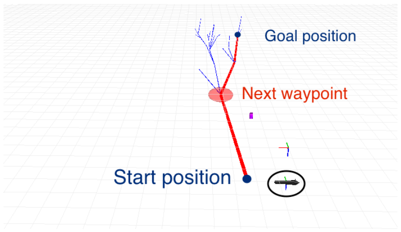
After the positive results obtained in simulation, we moved to the real-world environment. Path planning queries were defined as in simulated scenarios, *i.e.,* start and goal configurations were located in opposite sides of the breakwater structure. For security reasons, the vehicle is connected to surface with a wireless access point buoy that allows us to monitor while conducting the mission and abort it in case of unexpected behavior is detected. The SPARUS-II performed different autonomous missions with a constant surge speed $u = 0.5m/s$ and a maximum turning rate $r_{max} = 0.3rad/s$.

In real-world in-water tests there are situations that differ from the behavior in the simulated environment. Fig. 6 presents the reconstruction of the environment perceived by the vehicle, where the resulting Octomap has been overlaid on a real image of the concrete blocks. Albeit Octomaps use a probabilistic state representation, we observed noisy acoustic data that was included in the map and remains over the complete execution of the mission. This is mainly caused when an echosounder introduces a single noisy data over an unexplored position on the map, which is marked as an occupied state. However, when correct information about such state is repetitively included, it not only corrects it, but also makes it more immune to future noisy measurements.

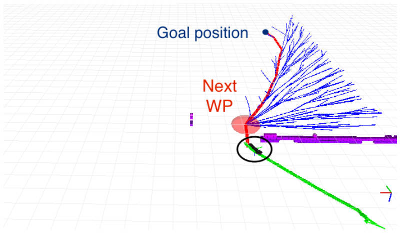
For in-water trials, it was necessary to increase t_{comp} up to $1.5s$ to guarantee, experimentally, that in case of replanning, a solution can be found. However, the quality of that first solution can be inefficient in terms of distance. The t_{comp} increase and the quality decrease of the first result after replanning was expected due to the greater computational load (sensors processing) and less computational power of the vehicle. It was observed that due to a non-zero pitch motion, together with the echosounder that works as a single beam sensor, oftentimes it takes longer to detect obstacles in the vehicle's direction of motion, which delays replanning maneuvers (see Fig. 7a, 7b). Despite these delays,

[1]http://cirs.udg.edu/

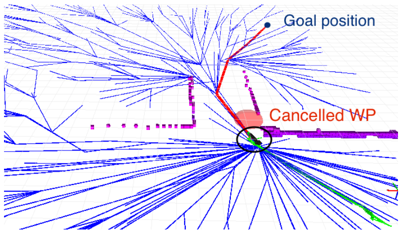

(a)



(b)



(c)



(d)

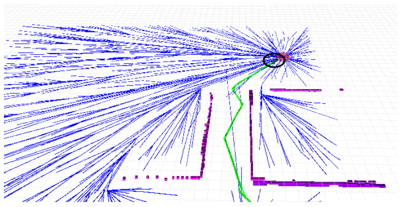

(e)

Fig. 5: (a) SPARUS-II AUV in UWSim [14]. In blue, the branches of the RRT*. In green, the path followed by the robot. In red, path with the best known cost (length) to the goal. (b) the first waypoint sent to vehicle, where the solution path from start to goal configurations resembles a straight line when environment is completely unknown. (c) new waypoint is sent to the controller and the tree is pruned. (d) a waypoint is invalidated for being under collision. (e) the vehicle has moved through the gap between the two blocks and approaches to the last waypoint.

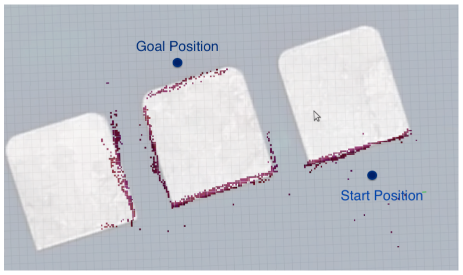


Fig. 6: Resulting map after conducting a mission. Start and goal positions are indicated.

vehicle safety remained throughout missions execution. Another clearly observed situation was the repetitive replanning because of resulting paths that did not consider motion constraints of the vehicle, especially in turning maneuvers. We present preliminary results to cope with this situation in next section. Figure 7 shows different situations presented throughout of the execution of an autonomous mission in the test scenario.

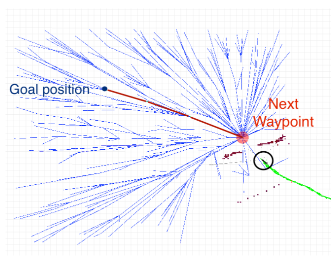
C. Simulation Results with Differential Constraints

Planning with differential constraints is related to consider the limits of feasible system maneuvers, which are described by a set of differential equations. Although some of the classical grid-based methods have addressed this problem, their approaches discretize the C-Space by defining a finite set of possible maneuvers, consequently, limiting the solution paths [16], [17]. On the other hand, because of their random nature, sampling-based algorithms have proved to efficiently explore the C-Space without limiting or discretizing it. This, together with factors mentioned in Section I, including applications requirements in underwater settings such as adaptability for planning on-line, re-planning and precise maneuvers accounting for vehicle constraints, motivated us to base our framework on a sampling-based algorithm, as presented before.

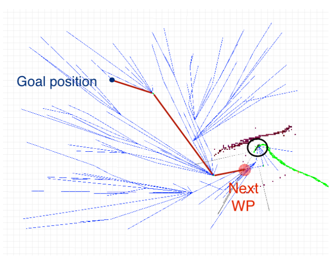
We have simulated start-to-goal queries and solved them with our framework using an RRT considering the vehicle differential constraints (see Figs. 8). Figure 8b presents a simulation of SPARUS-II executing one of such paths in UWSim. These simulations generate paths that prove to be collision-free and feasible (*i.e.,* expected maneuvers meet vehicle motion constraints), which reduces the replanning procedure callbacks and, consequently, unnecessary maneuvers. A next step will be to conduct real-world experiments with this additional consideration.

V. CONCLUSIONS AND FURTHER WORK

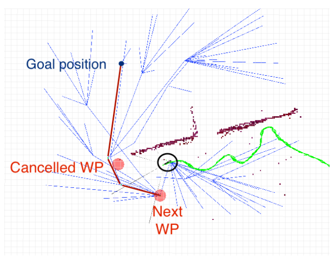
In this paper, we proposed a framework for planning collision-free paths online for an AUV, using our variant of the RRT* sampling-based algorithm, which permits replanning while moving through an unknown scenario. To do so, the framework also has an online mapping module and a mission handler that coordinates the execution of the task. To validate our approach, we presented the execution of missions in both simulated and real-world scenarios. In the



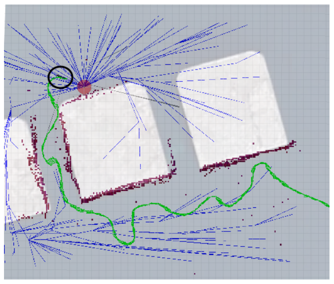
(a)



(b)

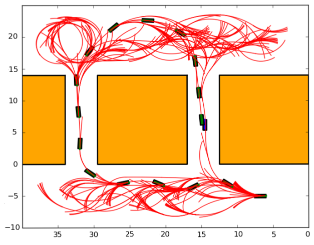


(c)

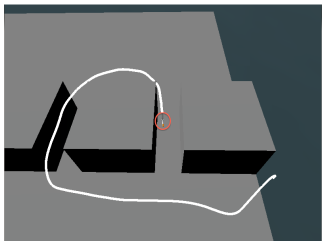


(d)

Fig. 7: Results from the real-world experiments. (a) the vehicle approaches to a waypoint that is located inside a concrete block that has not been mapped completely yet. (b) concrete block has been detected and replanning process has generated waypoints that maneuver the vehicle to avoid collision. (c) a waypoint has been generated close to a series of occupied positions (wall of the second block). Such point is later invalidated and a new waypoint is generated by replanning process. (d) the vehicle has moved through the four-meter gap between the second and third concrete block and approaches to the last waypoint.



(a)



(b)

Fig. 8: (a) resulting paths and all the intermediate states of the RRT expansion with differential constraints for test query. (b) simulation over UWSim of SPARUS-II executing the resulting path (a).

latter, the incidence of noisy measurements was overcome thanks to the probabilistic mapping framework and the replanning capacity of our approach.

Results in AUV mapping have shown the capability to incrementally build maps while planning paths at the same time. We also presented preliminary results on path planning while considering differential constraints. Working along this line, we plan to extend this approach to consider the vehicle motion constraints. Finally, we will extend our approach to 3D motion by modifying specific parts of the RRT*, avoiding the need of altering the main structure of the framework.

References

[1] F. S. Hover, R. M. Eustice, A. Kim, B. 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, pp. 1445–1464, Nov. 2012.

[2] E. Galceran, R. Campos, N. Palomeras, D. Ribas, M. Carreras, and P. Ridao, "Coverage Path Planning with Real-time Replanning and Surface Reconstruction for Inspection of Three-dimensional Underwater Structures using Autonomous Underwater Vehicles," *Journal of Field Robotics*, 2014.

[3] A. Mallios, P. Ridao, M. Carreras, and E. Hernàndez, "Navigating and mapping with the SPARUS AUV in a natural and unstructured underwater environment," in *MTS/IEEE OCEANS*, (Waikoloa), 2011.

[4] Y. Petillot, I. T. Ruiz, and D. M. Lane, "Underwater vehicle obstacle avoidance and path planning using a multi-beam forward looking sonar," *IEEE Journal of Oceanic Engineering*, vol. 26, no. 2, pp. 240–251, 2001.

[5] T. Maki, H. Mizushima, H. Kondo, T. Ura, T. Sakamaki, and M. Yanagisawa, "Real time path-planning of an AUV based on characteristics of passive acoustic landmarks for visual mapping of shallow vent fields," in *MTS/IEEE OCEANS*, (Vancouver), 2007.

[6] R. Bohlin and L. E. Kavraki, "Path planning using lazy PRM," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, pp. 521–528, 2000.

[7] K. Belghith, F. Kabanza, L. Hartman, and R. Nkambou, "Anytime dynamic path-planning with flexible probabilistic roadmaps," in *IEEE International Conference on Robotics and Automation (ICRA)*, no. May, pp. 2372–2377, 2006.

[8] D. Ferguson and A. Stentz, "Anytime RRTs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, no. line 3, pp. 5369–5375, Oct. 2006.

[9] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime Motion Planning using the RRT*," in *IEEE International Conference on Robotics and Automation (ICRA)*, no. May, pp. 1478–1483, May 2011.

[10] S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," *The International Journal of Robotics Research*, vol. 30, pp. 846–894, June 2011.

[11] 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, Feb. 2013.

[12] S. M. LaValle, *Planning Algorithms*. Cambridge: Cambridge University Press, 2006.

[13] N. Palomeras, A. El-Fakdi, M. Carreras, and P. Ridao, "COLA2: A Control Architecture for AUVs," *IEEE Journal of Oceanic Engineering*, vol. 37, pp. 695–716, Oct. 2012.

[14] M. Prats, J. Perez, J. J. Fernandez, and P. J. Sanz, "An open source tool for simulation and supervision of underwater intervention missions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2577–2582, Oct. 2012.

[15] 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.

[16] A. Alvarez, A. Caiti, and R. Onken, "Evolutionary Path Planning for Autonomous Underwater Vehicles in a Variable Ocean," *IEEE Journal of Oceanic Engineering*, vol. 29, pp. 418–429, Apr. 2004.

[17] B. Garau, A. Alvarez, and G. Oliver, "Path Planning of Autonomous Underwater Vehicles in Current Fields with Complex Spatial Variability: an A* Approach," in *IEEE International Conference on Robotics and Automation (ICRA)*, no. April, pp. 194–198, 2005.