To appear in the Proc. of the 2016 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems.

Planning Feasible and Safe Paths Online for Autonomous Underwater Vehicles in Unknown Environments

Juan David Hernández, Mark Moll, Eduard Vidal, Marc Carreras, and Lydia E. Kavraki

***Abstract*— We present a framework for planning collision-free and safe paths online for autonomous underwater vehicles (AUVs) in unknown environments. We build up on our previous work and propose an improved approach. While preserving its main modules (mapping, planning and mission handler), the framework now considers motion constraints to plan feasible paths, *i.e.,* those that meet vehicle's motion capabilities. The new framework also incorporates a risk function to avoid navigating close to nearby obstacles, and reuses the last best known solution to eliminate time-consuming pruning routines. To evaluate this approach, we use the Sparus II AUV, a torpedo-shaped vehicle performing autonomous missions in a 2-dimensional workspace. We validate the framework's new features by solving tasks in both simulation and real-world in-water trials and comparing results with our previous approach.**

I. INTRODUCTION

Most common and potential applications of autonomous underwater vehicles (AUVs) include imaging and inspecting different kinds of structures such as confined natural environments [1], in-water ship hulls [2], natural structures on the sea floor [3], as well as collecting oceanographic information as biological, chemical, and even archaeological data. Albeit these applications share some common requirements with others in the domain of mobile robots (*e.g.,* localization, mapping, vision, etc.), navigating autonomously while conducting such type of tasks in an underwater milieu differs in certain factors, such as the presence of external disturbances (currents), low-range visibility and limited navigation accuracy.

One option to deal with such constraints is to use a planner capable to plan collision-free paths online, thus allowing to adapt and replan in order to overcome global position inaccuracy, especially when navigating in close proximity to nearby obstacles. In this respect, Petillot *et al.* [4] proposed a first approach for underwater vehicles to plan paths while avoiding obstacles online, which used real-world multibeam sonar datasets of acoustic images obtained by a remotely operated vehicle (ROV). However, they validated their approach by guiding a simulated model. Moreover capability for mapping and planning online and simultaneously was not proven. Along this line, Maki *et al.* [5] presented a method to plan paths online, which used landmarks to guide an AUV. However, their approach did not permit replanning maneuvers and results were obtained in a water tank, *i.e.,* in a highly controlled environment.

Fig. 1: Sparus II, a torpedo-shaped AUV.

To cope with some of the aforementioned limitations, we presented a framework for solving start-to-goal queries in unknown environments for AUVs [6]. In doing so, the framework establishes an online mapping and path planning architecture that leads the AUV to navigate while building, simultaneously, a 2-dimensional (2D) representation of vehicle surroundings. The framework has a *mapping* module that uses Octomaps [7] to represent the environment, a *planning* module that calculates online collision-free paths, and a *mission handler* that coordinates the planner and the AUV controllers.

Although the framework succeeded in navigating to a specified goal position in an unknown environment, simulation and real-world results also outlined different aspects that could be enhanced such as decreasing computation time and reducing the number of replanning maneuvers. With these aspects in mind, in this paper we extend, improve and validate our framework for planning collision-free and feasible (doable) paths for a torpedo-shaped AUV. The new framework can be used not only for AUVs, but also for ground or aerial vehicles that deal with online computation constraints in unknown environments.

While preserving characteristics from our previous work [6], the main contributions of this paper are 1) the incorporation of motion constraints as a mechanism to avoid generating unfeasible paths, thus reducing the number of potentially expensive replanning maneuvers, 2) a path optimization function that combines collision risk and path length, for which we have defined different formulations to trade off its computational speed and accuracy, 3) the reuse of the last best known solution as a starting point for an

Work on this paper by J.D. Hernández, E. Vidal, and M. Carreras 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 the Colombian Government through its Predoctoral Grant Program offered by Colciencias. Work on this paper by M. Moll and L.E. Kavraki has been supported in part by NSF IIS 1317849.

J.D. Hernández, E. Vidal, and M. Carreras are with the Underwater Vision and Robotics Research Center (CIRS), University of Girona, Spain. juandhv@eia.udg.edu, rc4559@gmail.com, marc.carreras@udg.edu. M. Moll and L.E. Kavraki are with the Department of Computer Science at Rice University, Houston, TX, USA. {mmoll, kavraki}@rice.edu

anytime tree-based path planning approach, which permits to eliminate time-consuming pruning procedures previously used, and 4) the experimental evaluation of the new planning framework using the Sparus II AUV (see Fig. 1) in a real-world setting. Results demonstrate the suitability of our approach for the aforementioned applications.

II. FRAMEWORK FOR PLANNING FEASIBLE AND SAFE PATHS ONLINE FOR AUVs

Figure 2 shows the path planning framework pipeline whose structure remains as originally proposed in [6]. While the *mission handler* uses the same communication protocol to coordinate the other two functional modules, and the *mapping* module continues representing the environment with Octomaps [7], the *planning* module, on the other hand, contains major novelties. This section details on such changes, which seek to obtain a better quality solution as far as path length and vehicle safety are concerned.

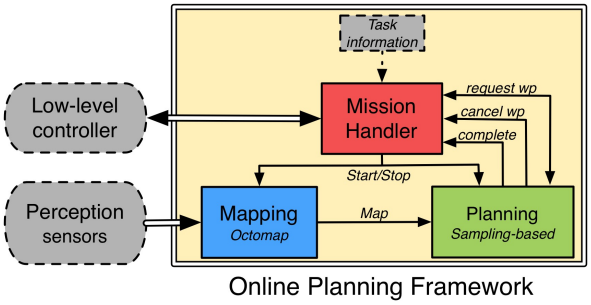


Fig. 2: Main modules of the online planning framework.

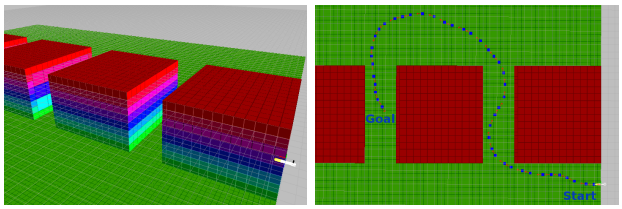
A. Planning Under Motion Constraints

In this work, we consider an AUV that navigates at a constant depth and has two back thrusters for motion on the horizontal plane, but no thruster for lateral motion. This means that the vehicle is subject to motion constraints or differential constraints. Planning under such restrictions is commonly known as *kinodynamic motion planning*, a term introduced by Donald *et al.* [8], which refers to planning collision-free motions by considering the limits of feasible system maneuvers expressed by differential equations. In the particular case of an AUV, this corresponds to a second-order differential equation of an underactuated system. However, for online path planning purposes, we represent the non-holonomic torpedo-shaped AUV as a simple car-like vehicle with dynamics defined by:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v.cos\,(\psi) \\ v.sin\,(\psi) \\ r \end{bmatrix}, \tag{1}$$

where $q = [x, y, \psi]^T$ is the state of the vehicle that includes its 2D position and orientation with respect to an inertial reference frame, and $\dot{q} = \left[\dot{x}, \dot{y}, \dot{\psi}\right]^T$ is the first time derivative that depends on the state itself and the control inputs, linear/surge speed (v) and rate of turn (r). From Eq. (1), it can be concluded that the configuration space (C-Space) of a torpedo-shaped AUV performing tasks in a plane is 3-dimensional, *i.e.,* each $q \in SE(2) = R^2 \times \mathcal{S}$.

Sampling-based approaches, such as the rapidly-exploring random tree (RRT) [9], have been demonstrated efficient for solving path planning problems for this type of systems [10]. The RRT is mainly composed of two procedures, *sample* and *extend*. The first of them randomly samples configurations to explore the C-Space, while the second expands the tree towards those configurations. In the case of expanding an RRT under differential constraints, new states (tree nodes) are obtained by integrating differential equations such as Eq. (1). Different variants of this approach have been proposed for aerial and terrestrial vehicles in order to generate smooth and feasible paths. An example of them is one in which a standard RRT is used to find a series of collision-free waypoints, which are then interpolated by using a cubic Bézier spiral to generate a smooth path to be followed by an unmanned aerial vehicle (UAV) [11]. Another alternative uses an RRT that is expanded by considering not only the vehicle dynamic model, but also the controller behavior [12]. Nonetheless, their major drawback is the lack of optimality in any possible metric.



(a) (b)

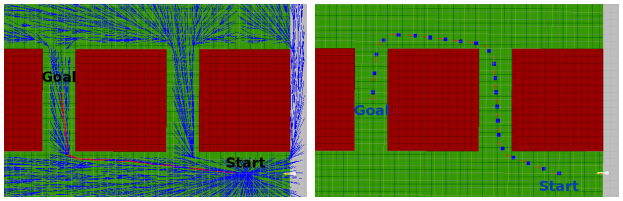
Fig. 3: (a) an Octomap represents a 3D environment that is composed of a series of equidistant blocks. (b) a top view shows a collision-free path that connects a start and goal configurations. The path was obtained by expanding an RRT using Eq. (1).

The solution path presented in Fig. 3b is an example of solving a start-to-goal query for an AUV, in which Eq. (1) has been used to expand an RRT, thus calculating a collision-free path that connects an initial and final configuration (position and orientation). Furthermore, given that Eq. (1) describes the vehicle's motion capabilities while navigating at constant depths, the solution paths are more likely to be feasible for a non-holonomic AUV, such as the Sparus II. However, as occurs with similar approaches mentioned before, their major drawback is the lack of optimality, a characteristic of most sampling-based approaches, at least in their original form. This aspect could become into a critical requirement for real-world missions that require optimization criteria, such as visibility for gathering information, vehicle autonomy, or even vehicle safety when navigating in close-proximity to nearby obstacles.

Karaman and Frazzoli presented the RRT* [13], a variant that includes the asymptotic optimality property. Its

main difference is a routine that checks if reconnecting new state's nearest nodes improves their associated cost, thus implying that the probability of obtaining an optimal path increases over time (see Fig. 4a). For doing this, the RRT* requires a steering function which permits such states (nodes) reconnection, which in the case of systems under differential constraints calculates the required input to dynamically evolve the system from a given state to a desired one. However, defining such function may become an intractable nonlinear control problem, especially when considering online computation constraints.

As an alternative to define a steering function, in this paper we adopt the Dubins vehicle model [14], whose dynamics has the general form presented in Eq. (1). Using three possible maneuvers as input, left, straight or right, Dubins curves define six possible paths that characterize the optimal trajectory between two states for a Dubins vehicle. This approach permits us to include motion constraints and use the RRT* (see Fig. 4b).



(a) (b)

Fig. 4: (a) RRT* expansion without using differential constraints, solution and intermediate states, for a start-to-goal query. (b) resulting path for the same query (including orientation) using an RRT* and Dubins maneuvers as steering function.

B. Planning Safe Paths Using Risk Functions

Conducting underwater missions with AUVs is a challenging task, especially when the vehicle is kinematically constrained as occurs with a non-holonomic AUV. However, including differential constraints to the path/motion planner may not suffice to minimize the risk of colliding with nearby obstacles, especially if the vehicle is exposed to external perturbations that do not permit to accurately follow the calculated path. Figure 5a, for instance, shows a feasible path, *i.e.,* one that considers the vehicle motion constraints, but leads the vehicle close to nearby obstacles, especially in turning maneuvers. Bearing this in mind, we now extend our problem to not only plan feasible paths, but to attempt to minimize the risk of colliding with the surroundings. To accomplish it, we propose an optimization objective that combines the length and the safety of the path. In this section, we present different approaches to establish the safety of the path, which are evaluated and compared in Sec. III.

1) Path Length + Clearance: a straightforward option is to maintain a minimum safe distance, or clearance, to the obstacles. For doing so, it is necessary to define a weighted metric that combines path length and clearance in order to minimize detrimental effects in the path quality, thus defining the associated cost of each configuration when planning with an RRT*. However, this approach has two main drawbacks, its high computational cost and the need to correctly specify weights to obtain a balanced metric, which is a non-trivial problem [15].

2) Risk Zones: including clearance calculation is especially expensive for sampling-based path planning methods, since it has to be performed for each sampled configuration and its intermediate steps when connecting to the others (*e.g.,* RRT* expansion). As an alternative, we propose to heuristically establish risk zones around the vehicle, as shown in Fig. 6a. The red and blue zones represent the collision and the non-risk zones, respectively, while the others (orange, yellow and green) have associated decreasing risk values as collisions move away from the vehicle position, as presented in Eq. (2) where n is the number of zones, $zone_{i-1}$ is closer than $zone_i$ to the collision zone, $risk_{i-1} > risk_i > \ldots > risk_n > 1$.

$$Risk(q) = \begin{cases} risk_1, & if\ Collision(zone_1) \\ risk_2, & if\ Collision(zone_2) \\ \ldots, & \ldots \\ risk_n, & if\ Collision(zone_n) \\ 1, & if\ not\ Collision(any\ zone) \end{cases} \qquad q = [x, y, \psi]^T \quad (2)$$

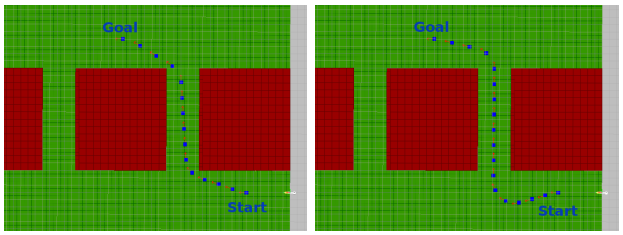
In order to combine this function with the path length, the total accumulated cost associated with each configuration is calculated as the integral of risk with respect to distance, as presented in Eq. (3). Such a cost function not only combines the risk and the length associated to a path, but also establishes the optimization criterion required in order to plan feasible and safe paths. A visual comparison between paths calculated using only the path length and the risk zones can be observed in Fig. 5.

$$Cost(q) = \int_0^q Risk(q)\,\mathrm{d}q \quad (3)$$

Finally, it is important to highlight that checking for collision from the inner to the outer zone, for a limited number of zones, is computationally more efficient than calculating position clearance. This is especially true when the environment is not described with a simplified representation (*e.g.,* convex obstacles), where distance to obstacles can be rapidly calculated, but instead a more detailed representation is used (*e.g.,* Octomaps).

3) Direction Vectors Risk: the previous approach attempts to penalize those configurations close to nearby obstacles by specifying a risk value depending on the affected zone. However, another alternative is to limit such evaluation to the directions defined by the possible maneuvers of the vehicle at the considered time. We call this approach *direction vectors risk*. We now define three (3) vectors in the straight and lateral motion directions and, instead of checking for complete zones, we check points for collision along the

vectors and assign risk values with the same principle as done with risk zones, *i.e.,* moving away from the vehicle decreases the risk (see Fig. 6b). Given that we are using Octomaps to represent the environment, checking collision for single points is more efficient than doing the check for zones (multiple points).



(a) (b)

Fig. 5: (a) RRT* expansion for a start-to-goal query using Dubins curves with path length as cost criterion. (b) RRT* expansion with risk zones cost criterion. It is worth noting that the resulting path is far from corners in turning maneuvers and attempts to stay in the middle of the corridor when navigating between two obstacles.

(a) (b)

Fig. 6: Sparus II AUV navigating between two obstacles, (a) an example of risk zones around the vehicle and (b) direction vectors risks, where it is observed how the risk decreases as moves away from the vehicle (red means high risk, while blue low risk).

C. Planning Feasible and Safe Paths Online

In the previous two sections, we presented our approaches to plan collision-free paths that are intended to be feasible (doable) and safe for a torpedo-shaped AUV. However, and as initially stated, our main goal is to provide the AUV the capability to navigate autonomously in an unknown environment, which means to map and plan paths simultaneously and online. In this section, we present two new characteristics with respect to our previous work [6] that contribute to satisfy online computation constraints while planning feasible and safe paths. Furthermore, we summarize the planning framework pipeline.

1) Opportunistic Risk Checking: without initial information of surroundings, the AUV is required to incrementally map and continuously (re)plan collision-free paths according to its partial knowledge of the environment. Under this situation, it is unnecessary to attempt to calculate the risk associated to configurations that lie in undiscovered areas. With this in mind, we propose to use what we call *Opportunistic Risk Checking*. With this strategy, we assume that any configuration (sampled or resulting from a tree expansion) that belongs to unexplored areas is safe (*i.e.,* maximum clearance or minimum risk), thus avoiding unnecessary risk checking routines. This is possible within our framework since it uses Octomaps to represent the environment, which permit to establish in advance if a configuration lies in an explored area or not. Section III shows different tests that demonstrate the advantages of using this strategy.

2) Reusing the Last Best Known Solution: in our previous work [6], we presented a framework that used a tree-based path planning method to solve start-to-goal queries, in which a tree of configurations was periodically traversed, checked and pruned as the vehicle moved and explored the environment. The main objective was to preserve the information about collision-free areas and known paths, while discarding those that become invalid. We now present a similar iterative scheme, but instead of conserving the whole tree, we propose to use the *last best known solution* as the remainder of the path calculated in the previous planning cycle, which starts at the point that the vehicle will reach at the next execution cycle. Furthermore, using last solution to start a new planning cycle implies not only a new valid solution according to an updated map, but also one that is at least as optimal as the previous one. Finally, this alternative approach avoids checking a subtree in which many of its configurations have probably become invalid because of the *Opportunistic Risk Checking* previously explained.

3) Framework for Online Planning Feasible and Safe Paths: throughout the previous sections, we have presented different characteristics that, together, endow a non-holonomic AUV with the capability to simultaneously map and plan feasible and safe paths online through an unknown environment. These characteristics extend our previous planning framework [6]. Algorithm 1 presents the execution pipeline to incrementally solve a start-to-goal query. It also summarizes the main contributions to the planning module presented in this paper (see [6] for further details concerning to mapping and mission handler modules).

In Algorithm 1, the main input parameters to solve a query are the start and goal configurations (position and orientation). To initialize the incremental solving routine, we select the RRT* as the planner that computes paths for a Dubins vehicle, set an empty list as the *Last Best Known Solution*, and define q_{new_start} as the starting configuration that will change as the vehicle conduct the mission (lines 2-4). To incrementally find a path to the goal (line 5), a main loop requests an updated version of the map (Octomap, lines 6-7), informs the planner to start from *Last Best Known Solution* (line 8), uses the planner to find a path (line 9), and gets the solution (line 10). At this point, the planner has provided a valid path that must be as optimal as the previous one, except that it has produced a longer but safer path. Before concluding a planning cycle, the incremental solving routine checks if a replanning maneuver has been requested. This would imply that the *mission handler* has detected that

the path from the current configuration to the last waypoint (WP) sent to the AUV controllers is not feasible and might lead the vehicle to a collision, thus implying that a new path should be found from current configuration (see lines 11-13). Finally, if a new WP is required, the last q_{new_start} will be sent to the *mission handler* (lines 14-15).

It is important to note that even if the previous (*Last Best Known*) solution is not in collision with an updated version of the map, the *planning* module will use it as a starting point (line 8) in order to attempt to improve such existing solution during a new planning cycle (lines 9 and 10).

Algorithm 1: incSolveStart2Goal(q_{start}, q_{goal})

```
   Input:
   q_start: Start configuration.
   q_goal: Goal configuration.
1  begin
2      planner ← RRT* (DubinsStateSpace)
3      last_best_known_solution ← {}
4      q_new_start ← q_start
5      while not stop_condition do
6          map ←reqUpdatedMap()
7          planner.updateMap(map)
8          planner.startFrom(last_best_known_solution)
9          planner.solve(q_new_start, q_goal)
10         last_best_known_solution ←
           planner.getSolution()
11         if replanning_requested then
12             q_new_start ← getCurrentConf()
13             last_best_known_solution ← {}
14         else if new_waypoint_requested then
15             sendWP(last_best_known_solution.pop())
16 end
```

III. RESULTS

In order to assess the new planning framework capabilities, we used the Sparus II (see Fig. 1), an AUV rated for depths up to $200m$ with hovering capabilities, which has two back thrusters for motion on the horizontal plane and one vertical thruster, *i.e.,* it can be actuated in surge, heave and yaw degrees of freedom (DOF). The AUV is equipped with a navigation sensor suite that includes a pressure sensor, a doppler velocity log (DVL), an inertial measurement unit (IMU) and a GPS to receive position fixes while at surface. To perceive and detect the surroundings, a mechanically-scanning profiler is located to cover a scan sector in the horizontal plane in the vehicle's direction of motion.

Sparus II AUV uses the component oriented layer-based architecture for autonomy (COLA2) [16], a control architecture integrated with the robot operating system (ROS). COLA2 not only operates aboard the real vehicle, but also interacts with the underwater simulator (UWSim) [17], which permits importing 3D environment models and simulating the vehicle's sensors and dynamics. We also make use of the open motion planning library (OMPL) that offers a convenient framework that can be adapted to specific planning problems [18].

To validate our approach, we have selected different scenarios for simulation and in-water real-world experiments. Before presenting such results, we first assess and establish the best alternative to estimate the risk of the path for both known and unknown environments. Then, we compare the framework performance solving different start-to-goal queries with respect the initial version [6] and the modified one resulting from the changes introduced in this work.

A. Comparison of Risk Functions

As a first test scenario, we selected the external and open area of the harbour of Sant Feliu de Guíxols (Spain), specifically a breakwater structure 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 $7m$ (see Fig. 7). In this scenario, start and goal configurations were located in opposite sides of the breakwater structure, thus the Sparus II AUV has to move amidst the concrete blocks. All queries have been defined to conduct missions at a constant depth, thus the motion is restricted to a 2D task.

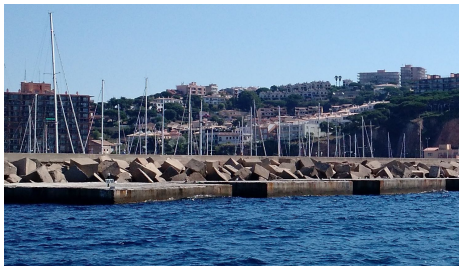
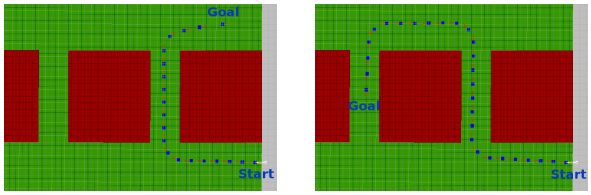


Fig. 7: Breakwater structure composed of a series of concrete blocks in Sant Feliu de Guíxols in Catalonia (Spain).

1) Planning Safe Paths in Known Environments: assuming the environment as explored, which in our case implies that we previously built an Octomap, we solved different start-to-goal queries to obtain feasible and safe paths. Using aforementioned alternatives (Section II-B) to estimate the risk of the path, we generated paths as those presented in Fig. 8, in which can be observed how the path maintains a safe distance from nearby obstacles (zones in red), in contrast to what occurs when only path length criterion is considered (see Fig. 5a). Even though all approaches generate similar results, computation time differs considerably. Figure 9 presents the average computation time required to solve $Task1$ and $Task2$ by each approach (see Fig. 8), where using only *Path Length* is clearly the least expensive method, while *Path Length + Clearance* and *Direction Vectors Risk* are the most and least expensive, respectively, when including the risk of the path.

2) Planning Safe Paths in Unknown Environments: from the last section, it can be concluded that the best computational alternative to include the risk associated to a path when the environment is completely mapped is the *Direction Vectors Risk* approach. However, when dealing with unknown



(a) (b)

Fig. 8: Start-to-goal queries solutions include risk functions to obtain safe paths for (a) $Task1$ and (b) $Task2$.

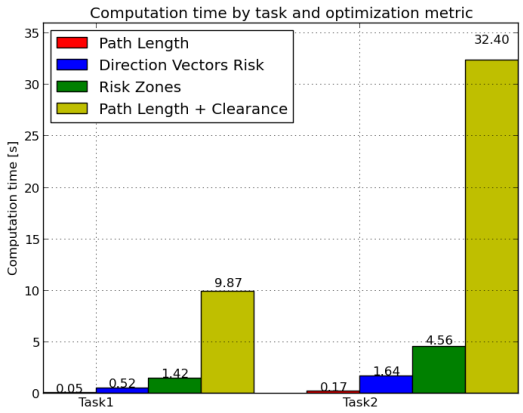
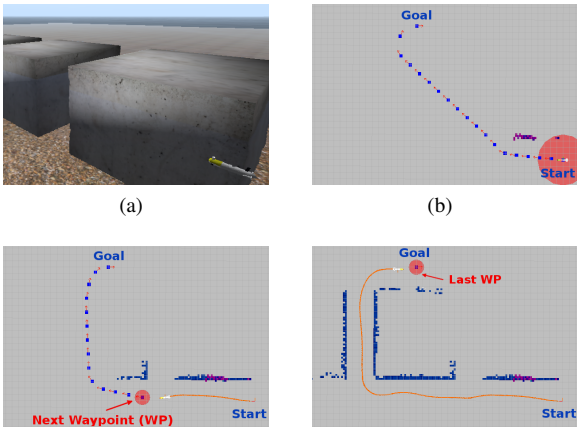


Fig. 9: Average computation time, over 20 runs, required to solve $Task1$ and $Task2$ (Fig. 8) using different approaches to include risk of the path as the optimization criterion.

environments, *i.e.,* exploring while mapping incrementally, this approach may cope with situations in which partial information of the environment does not permit to estimate correctly the risk. In turning maneuvers, for instance, if an obstacle is located in the lateral motion direction, and it is not completely represented in the map so that the direction vector risk does not coincide with the available partial information, the *Direction Vectors Risk* approach will indicate the configuration as safe, while the *Risk Zones* will estimate correctly the risk. For this reason, we use the latter approach when dealing with unknown environments.

Although planning feasible and safe paths using *Risk Zones* is considerably faster than calculating clearance explicitly (see Fig. 9), having calculation times in the order of seconds may become into a limitation when coping with online computation constraints. In Section II-C we proposed *Reusing the Last Best Known Solution* and *Opportunistic Risk Checking* as mechanisms to overcome this situation. To validate their efficiency, we solved and simulated the execution of a start-to-goal query in the virtual scenario of concrete blocks, but without assuming any previous information of the environment (see Fig. 10).

In assessing the utility of using *Opportunistic Risk Checking*, we have solved the query presented in Fig. 10 with and without using this approach. Albeit in both cases the framework succeeded in conducting the task, Fig. 11a demonstrates that without it almost 80% of the total computation time is dedicated to risk checking routines over the whole



(a) (b) (c) (d)

Fig. 10: (a) Sparus II initial position for different start-to-goal queries in an equivalent virtual scenario of the breakwater structure in UWSim [17]. (b) Sparus II starts a mission by submerging to a specified depth, while it maps and solves a start-to-goal query simultaneously. (c) Equipped with a scanning profiler, it incrementally builds a 3D representation of the environment and corrects the path, (d) to finally approach to the specified goal configuration.

mission. In the opposite scenario, *i.e.,* using *Opportunistic Risk Checking*, its associated computation time increases as the environment is progressively explored, but even so, it does not consume such a percentage of computation time, not even at the end of the mission. This is especially noticeable when a mission does not require exploring and mapping completely the environment. Therefore, the use of the proposed mechanism affects not only the time required to find a solution, since it permits a better tree expansion (more states, see Fig. 11b), but also improves the workspace exploration and the path quality given by the asymptotic optimal algorithm.

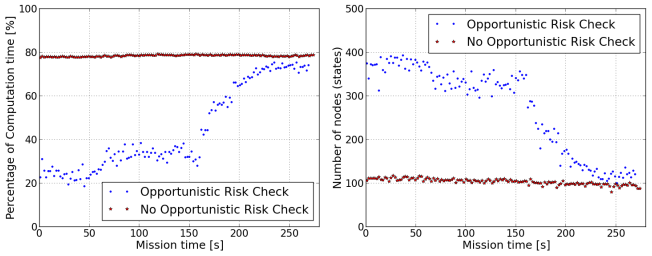
B. Solving Start-to-goal Queries in Unknown Environments

Finally, to sum up the overall improvement of the proposed framework, we have run several tests, including simulations and real-world in-water trials. Results of these tests are reported next.

1) Simulation Results: using simulated environments permits to easily evaluate and estimate the expected system behavior before conducting a task in a real-world scenario. We defined two different virtual scenarios and attempted to solve different start-to-goal queries to compare our previous work [6] with the one presented in this paper. The first of them is the breakwater structure mentioned throughout previous sections (see Figs. 7, 8). In such scenario, we attempted to solve two different queries (see Fig. 12). After executing 10 times the same mission, we observed that the number of successful attempts increased and the mean of replanning maneuvers (over the total successful missions) decreased considerably by using the new framework (see Table I).

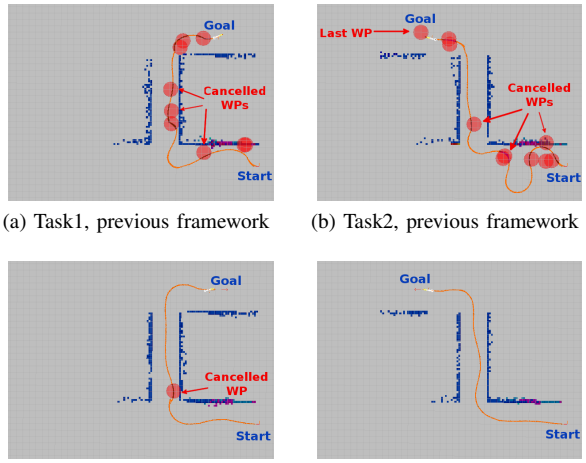
	Virtual Scenario1: Breakwater Structure (Fig. 12)				Virtual Scenario2: Sea Rocks (Fig. 13)	
	Task1 (10 attempts)		Task 2 (10 attempts)		Task 3 (10 attempts)	
	# Succ. attempts	Mean of replan. man.	# Succ. attempts	Mean of replan. man.	# Succ. attempts	Mean of replan. man.
Prev. Fram.	7	8.85	7	6.42	6	6.6
New Fram.	10	0.3	9	0.1	7	1.14

TABLE I: Comparison of solving the tasks shown in Figs. 12 and 13 using our previous work [6] and the new planning framework presented in this paper. Effects of proposed improvements are reflected in: 1) the increase in the number of successful attempts, 2) the decrease of the mean of replanning maneuvers over the total successful missions. This latter implies that the vehicle has to deal with less risky situations, since a replanning maneuver supposes that the vehicle was being led to a possible collision.



(a) (b)

Fig. 11: Incidence of *Opportunistic Risk Checking* approach when solving query presented in Fig. 10. (a) if configurations located in undiscovered areas are not assumed as safe, risk checking routines require almost 80% of computation time during the whole mission. Otherwise, it will increase progressively as the environment is explored. (b) consequently, if a high percentage of computing power is dedicated to risk checking, the number of tree nodes (states) will remain low during the whole mission, thus limiting the tree expansion and path quality.



(a) Task1, previous framework (b) Task2, previous framework

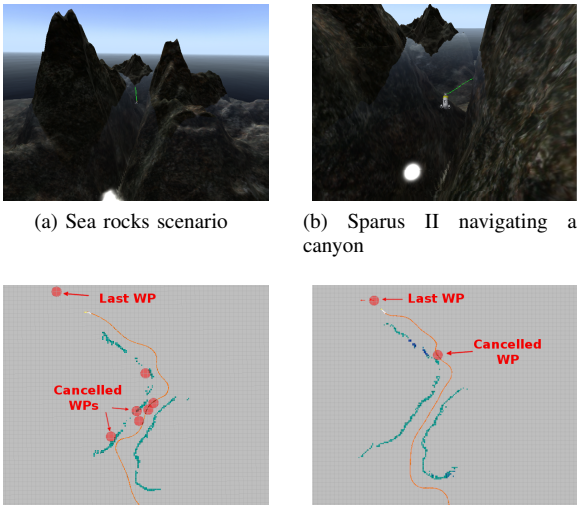
(c) Task1, new framework (d) Task2, new framework

Fig. 12: Solving start-to-goal queries in a virtual scenario of concrete blocks. Our previous work [6] and the new planning framework have been used to solve the tasks. The number of replanning maneuvers are equivalent to the number of cancelled WPs.

The second virtual scenario resembles a natural environment composed of a series of sea rocks with a canyon between them (see Figs. 13a, 13b). Over it, we attempted to solve a start-to-goal query (see Figs. 13c, 13d). Again, we observed that the number of successful attempts increased and the mean of replanning maneuvers decreased considerably (see Table I).

2) Real-world Results: after validating our new framework in simulation, we conducted in-water trials in the real-world breakwater structure scenario (see Fig. 7). For safety reasons, the vehicle was connected to surface with a wireless access point buoy that allowed us to monitor the mission and abort it in case of detecting an unexpected behavior. The Sparus II performed different autonomous missions with a constant surge speed $u = 0.5m/s$ and a maximum turning rate $r_{max} = 0.3rad/s$.

Figure 14 presents the Sparus II AUV conducting one of such missions, which consisted of solving a start-to-goal query that required the vehicle to navigate between two concrete blocks of the breakwater structure. Both the previous and the new proposed frameworks were used to solve the task. Figs. 14a and 14b not only present a similar reconstruction of the environment (since this work does not

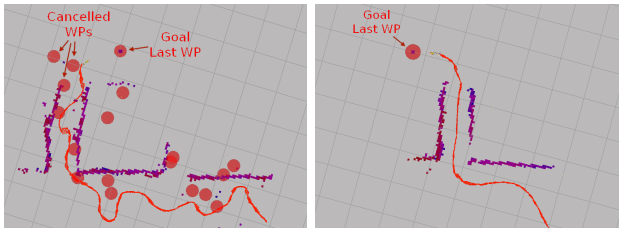


(a) Sea rocks scenario (b) Sparus II navigating a canyon

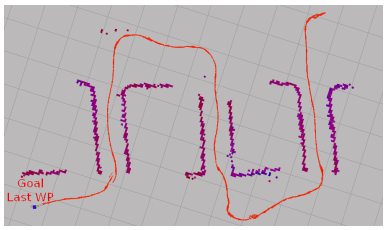
(c) Previous framework (d) New framework

Fig. 13: Solving start-to-goal query in a virtual scenario of sea rocks. Our previous work [6] and the new framework have been used to solve the task. The number of replanning maneuvers are equivalent to the number of cancelled WPs.

introduce any relevant change in the *mapping* module), but also proves how replanning maneuvers decreased, just as expected from simulation results. Apart from visual results, it is also worth to mention that number of successful attempts were considerably higher with our new approach, 7 over 13, while experiments with the old framework only succeeded 2 times of 5. However, these results are not presented in a Table, since experiments were conducted in different days, *i.e.,* with different weather conditions. Trials with our new approach dealt with worse conditions indeed.



(a) Previous framework (b) New framework



(c) New framework solves successive start-to-goal queries

Fig. 14: Results from the real-world experiments. (a), (b) Sparus II navigates amongst two concrete blocks to move from one side to the other of the breakwater structure. (c) Sparus II traverses the breakwater structure multiple times by solving successive start-to-goal queries.

IV. CONCLUSIONS AND FURTHER WORK

In this paper, we extended our previous work and proposed an improved framework for planning feasible and safe paths online for an AUV. To guarantee these characteristics, we considered motion constraints to ensure planning paths that are feasible (doable) according to the vehicle's motion capabilities. Furthermore, we presented and assessed multiple alternatives to evaluate the risk associated to a solution path. As a result, we established a function that combines the length and safety of the path into a single optimization objective. Finally, we proposed *Reusing the Last Best Known Solution* to avoid the need of pruning the tree of configurations, as it was done previously, and also introduced the *Opportunistic Risk Checking* strategy, both of them as mechanisms to meet online computation constraints. This latter serves as an alternative to fully considering the sensing and motion uncertainty that could be prohibitively expensive, especially in the proposed application that requires planning paths while exploring the unknown environment, thus implying limited computational resources.

To validate our new approach and its characteristics, we presented the execution of missions in both simulated and real-world scenarios and compared the results with those obtained with our previous approach. Results showed the increase in the number of successful missions and the decrease of replanning maneuvers, which permitted conducting longer tasks composed of successive start-to-goal queries. We also presented preliminary simulation results on a virtual scenario that resembles a natural environment. However to be able to successfully deal with such type of scenarios, we plan to extend our framework to consider 3D workspaces. Furthermore, while in future work we plan to model ocean currents, especially for long missions where the induced error can be significant, Fig. 14c shows that incremental (re)planning can be effective at navigating around obstacles subject to currents over long time horizons. Finally, it is also important to point out that the heuristic used to define the risk zones and their respective values works well, however further work is needed to generalize its use for different scenarios.

REFERENCES

[1] A. Mallios, P. Ridao, D. Ribas, *et al.*, "Toward autonomous exploration in confined underwater environments," *J Field Robot*, vol. 7, nov 2015.

[2] F. S. Hover, R. M. Eustice, A. Kim, *et al.*, "Advanced perception, navigation and planning for autonomous in-water ship hull inspection," *Int J Robot Res*, vol. 31, pp. 1445–1464, nov 2012.

[3] E. Galceran, R. Campos, N. Palomeras, *et al.*, "Coverage Path Planning with Real-time Replanning and Surface Reconstruction for Inspection of Three-dimensional Underwater Structures using Autonomous Underwater Vehicles," *J Field Robot*, 2014.

[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 J Ocean Eng*, vol. 26, no. 2, pp. 240–251, 2001.

[5] T. Maki, H. Mizushima, H. Kondo, *et al.*, "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*, 2007.

[6] J. D. Hernández, E. Vidal, G. Vallicrosa, *et al.*, "Online path planning for autonomous underwater vehicles in unknown environments," in *IEEE Int Conf Robot Autom*, (Seattle), pp. 1152–1157, may 2015.

[7] A. Hornung, K. M. Wurm, M. Bennewitz, *et al.*, "OctoMap: an efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, pp. 189–206, feb 2013.

[8] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic Motion Planning," *Journal of the ACM*, vol. 40, pp. 1048–1066, nov 1993.

[9] S. M. LaValle and J. J. Kuffner, "Randomized Kinodynamic Planning," *Int J Robot Res*, vol. 20, pp. 378–400, may 2001.

[10] S. M. LaValle, *Planning Algorithms*. Cambridge Univers. Press, 2006.

[11] K. Yang and S. Sukkarieh, "3D smooth path planning for a UAV in cluttered natural environments," in *IEEE/RSJ Int Conf Intel Rob Syst*, pp. 794–800, 2008.

[12] Y. Kuwata, S. Karaman, J. Teo, *et al.*, "Real-Time Motion Planning With Applications to Autonomous Urban Driving," *IEEE Trans Control Syst Technol*, vol. 17, pp. 1105–1118, sep 2009.

[13] S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," *Int J Robot Res*, vol. 30, pp. 846–894, jun 2011.

[14] L. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *Amer J Math*, vol. 79, no. 3, pp. 497–516, 1957.

[15] K. I. Tsianos, I. A. Sucan, and L. E. Kavraki, "Sampling-based robot motion planning: Towards realistic applications," *Computer Science Review*, vol. 1, pp. 2–11, aug 2007.

[16] N. Palomeras, A. El-Fakdi, M. Carreras, *et al.*, "COLA2: A Control Architecture for AUVs," *IEEE J Ocean Eng*, vol. 37, pp. 695–716, oct 2012.

[17] M. Prats, J. Perez, J. J. Fernandez, *et al.*, "An open source tool for simulation and supervision of underwater intervention missions," in *IEEE/RSJ Int Conf Intel Rob Syst*, pp. 2577–2582, oct 2012.

[18] I. A. Sucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robot Autom Mag*, vol. 19, pp. 72–82, dec 2012.