To appear in *Journal of Field Robotics*, 2019.
http://dx.doi.org/10.1002/rob.21827

Online Motion Planning for Unexplored Underwater Environments using AUVs

Juan David Hernández*
Underwater Vision and Robotics Research Center (CIRS)
University of Girona
Pic de Peguera, 13 (La Creueta)
Girona, Spain 17003
juandhv@eia.udg.edu

Eduard Vidal
Underwater Vision and Robotics Research Center (CIRS)
University of Girona
Pic de Peguera, 13 (La Creueta)
Girona, Spain 17003
eduard.vidalgarcia@udg.edu

Mark Moll
Department of Computer Science
Rice University
6100 Main St.
Houston, Texas, USA, 77005
mmoll@rice.edu

Narcís Palomeras
Underwater Vision and Robotics Research Center (CIRS)
University of Girona
Pic de Peguera, 13 (La Creueta)
Girona, Spain 17003
npalomer@silver.udg.edu

Marc Carreras
Underwater Vision and Robotics Research Center (CIRS)
University of Girona
Pic de Peguera, 13 (La Creueta)
Girona, Spain 17003
marc.carreras@udg.edu

Lydia E. Kavraki
Department of Computer Science
Rice University
6100 Main St.
Houston, Texas, USA, 77005
kavraki@rice.edu

Abstract

We present an approach to endow an autonomous underwater vehicle (AUV) with the capabilities to move through unexplored environments. To do so, we propose a computational framework for planning feasible and safe paths. The framework allows the vehicle to incrementally build a map of the surroundings, while simultaneously (re)planning a feasible path to a specified goal. To accomplish this, the framework considers motion constraints to plan feasible 3D paths, *i.e.,* those that meet the vehicle's motion capabilities. It also incorporates a risk function to avoid navigating close to nearby obstacles. Furthermore, the framework makes use of two strategies to ensure meeting online computation limitations. The first one is to reuse the last best known solution to eliminate time-consuming pruning routines. The second one is to opportunistically check the states' risk of collision.

To evaluate the proposed approach, we use the Sparus II performing autonomous missions in different real-world scenarios. These experiments consist of simulated and in-water trials for different tasks. The conducted tasks include the exploration of challenging scenarios such as artificial marine structures, natural marine structures, and confined natural environments. All these applications allow us to extensively prove the efficacy of the presented approach, not only for constant-depth missions (2D), but, more importantly, for situations in which the vehicle must vary its depth (3D).

*J.D. Hernández is currently a Postdoctoral Research Associate in the Department of Computer Science at Rice University, Houston, TX, USA. juandhv@rice.edu

1 Introduction

Since its beginning in the late 1950s, the capabilities and applications of autonomous underwater vehicles (AUVs) have continuously evolved. Their most common applications include imaging and inspecting different kinds of structures such as in-water ship hulls (Hover et al., 2012; Hurtós et al., 2015), natural structures on the sea floor (Galceran et al., 2014), as well as collecting oceanographic information such as biological (Grasmueck et al., 2006), chemical (Wei Li et al., 2006), and even archaeological data (Bingham et al., 2010).

Most of the aforementioned applications require *a priori* information of the area or structure to be inspected, either to navigate at a safe and conservative altitude (Grasmueck et al., 2006; Bingham et al., 2010) or to pre-calculate a survey path that may be corrected or reshaped online (Hover et al., 2012; Galceran et al., 2014). There exist other applications that require a vehicle to navigate in close-proximity to obtain more detailed information or to explore confined natural environments (*e.g.,* underwater caves) (Mallios et al., 2015). In these scenarios, *a priori* information of the surroundings might not be available, AUVs must operate in unexplored and cluttered environments, and therefore they are more exposed to collisions. In order to safely navigate in such environments, the AUV must be able not only to plan 3-dimensional (3D) paths that are feasible according to its motion capabilities, but also to calculate or modify such paths as new information of the surroundings becomes available.

Although these AUV applications share some common requirements with others in the domain of aerial and terrestrial robots (*e.g.,* localization, mapping, vision, etc.), navigating autonomously while conducting such type of tasks in an underwater environment differs in certain aspects, such as the presence of external disturbances (currents), low-range visibility, and limited navigation accuracy. Dealing with such constraints requires a path planner with online capabilities to help overcome the lack of environment information and the global position inaccuracy, especially when navigating in close proximity to nearby obstacles.

In this context, we have previously presented a framework for navigating safely from a start pose to a goal pose in unknown underwater environments (Hernández et al., 2015; Hernández et al., 2016c). The framework establishes an online mapping and motion planning architecture that leads an AUV to navigate while simultaneously building a representation of the vehicle's surroundings. The framework is composed of three main modules: 1) a *mapping* module that uses Octomaps (Hornung et al., 2013) to represent the environment, 2) a *planning* module that calculates online collision-free, feasible and safe paths, and 3) a *mission handler* that coordinates the planner and the AUV controllers. The framework considers the vehicle's motion constraints and avoids generating unfeasible paths; it also uses a path optimization objective that combines collision risk and path length, thus ensuring the vehicle's safety; and finally, it reuses the last best known solution as a starting point for an anytime tree-based path planning method.

Despite these positive results, the above-mentioned planning framework had some limitations. First, the framework had been only demonstrated to be effective for 2-dimensional (2D) missions. Furthermore, trials had been limited to scenarios with artificial and symmetric underwater structures, which did not prove the framework's efficacy when navigating through more challenging environments, such as natural formations. Finally, the controller used to reach the goal position by following a series of discretized waypoints, rather than the original resulting path. With the previous limitations in mind, this paper presents an extended and improved version of our framework that attempts to overcome these limitations.

While preserving the main characteristics from our previous work (Hernández et al., 2015; Hernández et al., 2016c), the main contributions of this paper are: 1) the state space extension to support 3D missions by combining the 2D motion (x, y, ψ) and the vertical motion (z), both constrained according to the vehicle's motion capabilities. While preliminary results were already presented in (Hernández et al., 2017), this paper includes additional tests and a more detailed description of the extended formulation and its implementation. 2) A new *mission handler* capable of interchanging information between the planning framework and a path tracking controller. This improved functional module eases the integration of the proposed approach with

different vehicles. 3) Finally, a comprehensive evaluation of the extended planning framework using the Sparus II (Carreras et al., 2018). This evaluation includes simulated and in-water trials in different real-world scenarios.

The remainder of this paper is organized as follows. Section 2 introduces previous research related to path planning for AUVs, especially for those vehicles that operate under under both motion and online computation constraints. Section 3 describes the problem of planning 2D paths under motion constraints, it then presents two alternatives to solve this problem, and finally it introduces an extended formulation for calculating 3D feasible and safe AUV paths. Section 4 explains in detail the framework introduced in (Hernández et al., 2015; Hernández et al., 2016c), while making the necessary extensions for 3D motions. In Section 5, different real-world scenarios are used in simulation and in-water trials to validate the proposed approach. Finally, concluding remarks and directions for further research are given in Section 6.

2 Related Work

AUVs are commonly divided into two main subcategories: buoyancy-driven and propeller-driven. The former category corresponds to the widely-known underwater gliders, or AUVs that are capable of modifying its buoyancy to generate sinusoidal-like vertical motion that, together with its wings, also permits horizontal (forward and lateral) motion. Gliders' propulsion technique results in low energy consumption, but they are also less maneuverable. The propeller-driven AUVs, on the other hand, correspond to those that use propellers or thrusters to generate 3D motions. The work presented in this paper is focused on this latter kind of vehicle.

An important aspect to consider when comparing the motion planning approaches for propeller-driven AUVs is their application. Based on this, the different contributions can be classified into two main categories. The first group gathers those applications that require coverage path planning (CPP) techniques, which are commonly applied to guide AUVs in survey tasks. The most common examples within this group include coverage missions used for creating detailed bathymetric maps of the seabed (Fang and Anstee, 2010; Galceran and Carreras, 2012; Galceran and Carreras, 2013; Galceran et al., 2013), detecting potential targets (such as underwater mines (Stack and Smith, 2003; Williams, 2010)) and inspecting artificial structures (such as in-water ship hulls (Hover et al., 2012; Hollinger et al., 2013; Englot and Hover, 2013)), as well as natural marine formations (Galceran et al., 2014).

The second group of propeller-driven AUV applications, on the other hand, includes those that are focused on safely and efficiently guiding the vehicle from one initial position to a specified goal. For doing so, different strategies have been applied to underwater vehicles. In fact, these start-to-goal methods are commonly used as low-level motion planners required for the aforementioned coverage applications. The work presented throughout this paper seeks to make contributions to this group of start-to-goal applications. It is therefore important to identify the main requirements among the different approaches used.

2.1 Planning Feasible AUV Paths

Motion planning alternatives for propeller-driven AUVs make use of different approaches such as potential fields (Warren, 1990; Sequeira and Ribeiro, 1994), genetic algorithms (Sugihara and Yuh, 1997; Alvarez et al., 2004; Hong-jian et al., 2004), as well as sensor-based (Ying et al., 2000; Houts et al., 2012), grid-based (Sequeira and Ribeiro, 1994; Carroll et al., 1992; Yan et al., 2012; Kim et al., 2012), and sampling-based methods (Tan et al., 2004; Rao and Williams, 2009; Caldwell et al., 2010; Poppinga et al., 2011; Heo and Chung, 2013; McMahon and Plaku, 2016). In some cases, the vehicle operates in open sea areas, where it does not usually have to deal with obstacles, narrow passages, or high-relief environments. However, this is not the case for the new potential applications presented in the Introduction.

A second requirement, especially critical in such new applications, is the capability of generating paths that meet the vehicle's motion constraints. The terms path planning and motion planning are often used interchangeably (Sucan, 2011), (LaValle, 2006). Throughout this paper, we will use the term motion planning, to emphasize the motion constraints involved. There are different methods that deal with the motion constraints imposed by AUVs. A first group includes those approaches that use sensor-based methods either to navigate through unknown underwater environments (Ying et al., 2000), or to follow the terrain shape of a given bathymetric map (Houts et al., 2012). In both cases, the vehicle is assumed to be equipped with a forward-looking sonar to reactively avoid collisions. A local planner computes maneuvers that satisfy the AUV kinematics or dynamics. An important characteristic of this kind of approach is the lack of global knowledge of the environment (*i.e.,* a map is not incrementally built), which can cause the vehicle to get trapped in complex scenarios.

A similar reactive approach establishes a set of inequality constraints that describes the convex obstacles contained in the configuration space (C-Space) (Petillot et al., 2001). The initial configuration is treated as the starting point of a nonlinear search, where the goal configuration is assumed to be a unique global minimum of the objective function. The start-to-goal query is then solved as an optimization problem, in which a local planner takes into account the vehicle's constraints. This strategy was one of the first online obstacle avoidance approaches for underwater vehicles; it used a real-world dataset of acoustic images obtained by a remotely operated vehicle (ROV) equipped with a multibeam forward-looking sonar. Its validity was demonstrated by guiding a simulated ROV. However, this work still lacked the capability to simultaneously map (detection) and plan online.

A formulation that represents obstacles as convex regions has likewise been used either to represent obstacles detected online, which trigger collision avoidance maneuvers (Qu and Yuan, 2009), or to approximate the terrain shape that must be followed by the vehicle (Murthy and Rock, 2010). In both cases, the low-level controller attempts to generate feasible (doable) trajectories by using the AUV kinematic equations and spline-based interpolation techniques, respectively. However, the main drawback of these approaches is the difficulty in creating a convex representation of complex obstacles.

Another common strategy used in some of the aforementioned methods, is trying to get as close as possible to a path that can be followed by an AUV. Petres et al., for instance, proposed a fast marching (FM)-based approach to find collision-free paths, which are smoothed by a cost function that contains kinematic and curvature constraints (Petres et al., 2007). Likewise, another example based on genetic algorithms (GAs) finds a valid route to the goal by using basis spline (B-spline) curves, thus seeking to generate more feasible trajectories for AUVs (Cheng et al., 2010).

Unlike the previous sensor-based approaches, another group of path planning methods for motion-constrained AUVs uses grid-based approaches. Sequeira and Ribeiro, for example, presented a two-layer framework that is composed of a high-level planner (HLP) and a low-level planner (LLP) (Sequeira and Ribeiro, 1994). The HLP creates a visibility graph using the information of the known obstacles, sea currents, and specified waypoints of the mission. Furthermore, the energy required to move between the graph nodes corresponds to the edge weights. The global and optimal motion plan to the goal is then found by Dijkstra's algorithm. Finally, in order to calculate the vehicle's maneuvers between the different solution segments, the LLP uses an artificial potential field (AFP). This way the total artificial force includes a 3D double integrator that takes into account the AUV motion constraints. There are other similar two-layer approaches, where the global path planning problem is tackled with a grid-based method, and the local motion planning deals with the AUV constraints (Arinaga et al., 1996). The main disadvantage of this line of works is that the grid-based layer generally requires *a priori* information of the environment, *e.g.,* a navigation map.

A more recent group of contributions includes those that use sampling-based planning algorithms. The most common approach builds a tree of collision-free configurations, which are obtained by integrating the differential equation that describes the AUV dynamic behavior (Tan et al., 2004; Caldwell et al., 2010; Heo and Chung, 2013). This strategy and its use with a torpedo-shaped AUV will be explained in more detail in Section 3.1.1.

Finally, there is another concept for generating feasible paths. It consists in utilizing Dubins curves (Dubins, 1957), which establishes a set of six maneuvers (RSR, RSL, LSR, LSL, RLR, LRL, where R states for Right, L for Left, and S for straight) to connect two configurations $q_i, q_j \in \mathcal{C} = SE(2)$. When no obstacles are involved, these curves correspond to the shortest trajectory between q_i and q_j for a car-like vehicle that moves under certain motion constraints. They have previously been shown useful of trajectory generation (Wehbe et al., 2014; Cao et al., 2016) and will be used in this work as well.

2.2 Online Mapping and Motion Planning

As mentioned before, another requirement for coping with the restrictions imposed by the new and potential applications for AUVs is to endow the vehicles with the capability of navigating through unknown or unexplored environments. For doing so, it is necessary that the vehicle can (re)plan its own motions while exploring the surroundings. However, little research has addressed this problem of mapping and planning safe routes simultaneously and online for AUVs. Maki *et al.* proposed an online motion planning method that uses landmarks to guide an AUV. Nonetheless, their approach does not permit replanning and, furthermore, results were obtained in a controlled environment (*i.e.,* in a water tank) (Maki et al., 2007). The approach presented in this paper has also been evaluated in challenging real-world scenarios.

Finally, Table 1 presents a summary of the main applications and contributions of motion planning methods in AUVs. For each contribution (row), the table list (columns from left to right), its bibliographical reference, wether or not: the planning method can be conducted online, requires an a priori map, calculates 3D paths, considers the vehicle's motion constraints, and, finally, some additional remarks.

Table 1: Summary of main applications of motion planning methods in AUVs. The sixth column, Constraints *, refers to motion constraints.

Category/Approach	**References**	**Online**	**A priori map**	**3D**	**Constraints***	**Remarks**
Sensor-based	(Ying et al., 2000)	Yes	No	Yes	No	Bug-like method. Vehicle may get trapped in complex scenarios.
	(Houts et al., 2012)	No	Yes	Yes	Yes	Trajectory planning approach. Online reactive system is used to handle unmapped obstacles.
	(Maki et al., 2007)	Yes	No	No	No	Behavior-based planner that uses landmarks to detect, react, and avoid collisions.
Potential fields	(Warren, 1990)	No	Yes	Yes	No	Possible local minima issues.
	(Sequeira and Ribeiro, 1994)	No	Yes	Yes	Yes	Potential fields for local planning.
Non-lin. programming	(Petillot et al., 2001)	Yes	No	No	Yes	Detected obstacles are represented as convex polygons. Nonlinear search.
Genetic algorithms	(Sugihara and Yuh, 1997) (Alvarez et al., 2004) (Hong-jian et al., 2004)	No	Yes	Yes	No	Commonly used in long-distance missions. Online reactive behavior to deal with unmapped obstacles.
Grid-based	(Sequeira and Ribeiro, 1994)	No	Yes	Yes	Yes	Dijkstra's algorithm solves global planning problem. Constraints are dealt by the local planner.
	(Carroll et al., 1992)	No	Yes	No	Yes	A* algorithm.
	(Yan et al., 2012; Kim et al., 2012)	No	Yes	Yes	Yes	A* algorithm variants that use circle searching approach.
Sampling-based	(Tan et al., 2004)	No	Yes	No	Yes	RRT algorithm that uses the dynamic model to expand the tree.
	(Caldwell et al., 2010)	No	Yes	No	Yes	Variant of RRT algorithm that uses model predictive control (MPC) to consider the system's constraints.
	(Poppinga et al., 2011)	No	Yes	No	No	Direct application of RRT and PRM algorithms for a 6D AUV.
	(Heo and Chung, 2013)	No	Yes	Yes	Yes	RRT algorithm that uses a 4D dynamic model to expand the tree.
	(McMahon and Plaku, 2016)	No	Yes	No	Yes	This work assumes virtual obstacles to represent test scenarios.
	[this paper]	**Yes**	**Yes**	**Yes**	**Yes**	This work proposes a sampling-based planning approach that seeks to endow an AUV with the capabilities to navigate unmapped environments. To do so, the proposed method requires (re)planning paths online. Such paths have to be 3D, and have to meet motion constraints.

3 Planning Feasible Paths Under AUV Motion Constraints

This section firstly explains the equations of motion used to approximate the 2D (at a constant depth) and 3D vehicle's behavior for motion-planning purposes. Secondly, it explains two different approaches to integrate such equations into the motion planner. Lastly, it presents and discusses the results obtained with both approaches in a simulated environment.

3.1 First-Order Motion Model

3.1.1 Constant-depth motions

In some AUV applications, it is reasonable to assume that the vehicle navigates at a constant depth. Furthermore, if the vehicle is a torpedo-shaped AUV it will usually only be able to change its direction of motion by moving forward/backward. In the case of the Sparus II AUV, the vehicle is equipped with two back thrusters that allow it to spin around. However, it is still not capable of conducting lateral motion. Therefore, the vehicle is subject to a motion constraint along its y-axis.

Differential motion constraints can be expressed as a set of differential equations in the general form $\dot{q} = f(q, u)$, where q and $\dot{q}$ are the system state and its first derivative, respectively, and u is the control input. Having this in mind, and considering the aforementioned constraints when navigating at a constant depth, we represent a non-holonomic and torpedo-shaped AUV as a simple car-like vehicle, with a first-order motion model defined as follows:

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

where $q = [x, y, \psi]^T$ corresponds to the system state 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, *i.e.,* linear/surge speed (v) and turning rate (w).

3.1.2 Variable-depth motions

Although there is an important number of AUV applications in which the vehicle navigates at a constant altitude or depth, there are other scenarios in which the vehicle is required to conduct 3D motions (see Secs. 1 and 2). For 3D motions, six different variables specify the vehicle's position and orientation, which means that the C-Space is $\mathcal{C} = SE(3)$ (see Fig. 1). Such AUV motions can be formulated with kinematic and dynamic models. The former ones describe the geometry of motion by relating the system positions and velocities. Dynamic models, additionally take into account the forces and torques that generate the motions.

Dynamic models, which normally consider all the 6 variables, have been used in some AUV motion-planning applications (Ying et al., 2000; Wehbe et al., 2014). However, their high computational cost makes them an inappropriate approach for the scenarios proposed in this paper, which require online (re)planning. The associated overhead is especially significant when using a global planning method, where all configurations[1] are generated by evolving the system, *i.e.,* performing numerical integration of the equations of motion. A kinematic model provides a less accurate approximation for the vehicle's motion constraints, but enables more computation time to be dedicated to finding better collision-free paths.

There exist different kinematic models that could be used for describing 3D motions. An AUV such as the Sparus II has some characteristics that allow simplifying such a model. First of all, let us assume that the

[1] All configurations refer not only to those that are part of the final solution path, but also those that create a tree of alternative paths evaluated during each planning cycle. In the tests presented in this paper, the planning cycle period is between 0.5s and 0.8s, in which a tree of hundreds of configurations is generated.

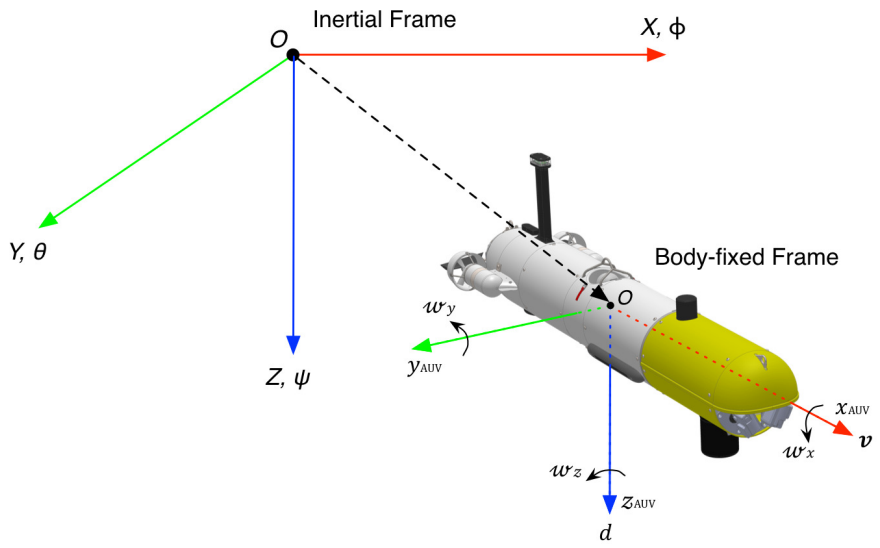


Figure 1: Perspective view of the Sparus II AUV, including the inertial and body-fixed frames, and the 6D vehicle's state and control variables

vehicle is capable of maintaining a near-zero roll angle ($\phi \approx 0$). Second, since the AUV has an independent propulsion force for its vertical motion, it is also possible to assume that the vehicle can follow a near-zero pitch vehicle motion ($\theta \approx 0$). This latter assumption allows us to decouple the motion in the horizontal plane from the heave (vertical) motion. Therefore, the 3D kinematic model can be described as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v\ \cos(\psi) \\ v\ \sin(\psi) \\ d \\ w \end{bmatrix}, \tag{2}$$

which can be written in the form $\dot{q} = f(q, u)$, where $q = [x, y, z, \psi]^T$, $\dot{q} = \left[\dot{x}, \dot{y}, \dot{z}, \dot{\psi}\right]^T$, and the control input $u = [v, d, w]^T$ that corresponds to the surge speed (v), the heave speed (d), and the rate of turn (w). This motion separation is not new for underwater vehicles, since it has also been done for dynamic models formulations (Mišković, 2010). Section 5 will present the evaluation of this approach with the Sparus II AUV. Finally, although both approximations, *i.e.,* near-zero roll and near-zero pitch motion, are valid for online motion planning purposes. It is important to note that the vehicle's controller uses a model that fully considers the vehicle's dynamics (Wadoo and Kachroo, 2010; Fossen, 2011).

Once the constraints have been established, the next step is to define a strategy to generate feasible motions that meet such constraints. The following sections present two different approaches, which use sampling-based algorithms to calculate such kind of motions.

3.2 Motion Planning Using Differential Equations

Sampling-based algorithms, especially tree-based algorithms like expansive-spaces tree (EST) (Hsu et al., 1999) and rapidly-exploring random tree (RRT) (LaValle and Kuffner, 2001) and their variants, have been proven to efficiently explore the C-Space while taking the motion constraints into account. Furthermore, there are other characteristics such as the incremental search behavior, which allows these methods to continuously (re)plan the solution path as the vehicle moves through unexplored environments. Altogether, this makes an RRT-based algorithm an appropriate approach for the intended AUV applications.

Algorithm 1: sampleRRT (LaValle and Kuffner, 2001)

Input:
q_{start} : Start configuration.
q_{goal} : Goal configuration.
$\mathcal{C}$: C-Space.
Output:
Rapidly-exploring Random Tree (RRT): T = (V, E).

```
1 begin
2     V = {q_start}
3     E = { }
4     while not stopCondition(T, goal) do
5         q_rand = C.generateRandomConf()
6         extendRRT(T, q_rand)
```

Algorithm 2: extendRRT (when dealing with motion constraints) (LaValle and Kuffner, 2001)

Input:
T: tree of collision-free configurations.
q_{rand}: configuration towards which the tree will be extended.
$\mathcal{C}$: C-Space.
Output:
Result after attempting to extend.

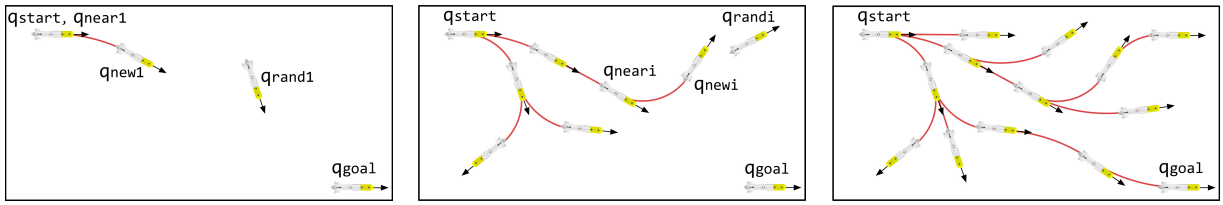
```
1 begin
2     q_near ← T.findNearestNeighbor(q_rand)
3     u_near_to_rand ←findInput(T, q_near, q_rand)
4     q_new, collision ←calcNewState(q_near, u_near_to_rand, Δt)
5     if not collision then
6         V.addNewNode(q_new)
7         E.addNewEdge(q_near, q_new)
```

A standard RRT algorithm builds a single tree that is rooted at an initial state (configuration)[2], which is incrementally expanded towards uniformly random *sampled* states (see Algorithm 1 (LaValle and Kuffner, 2001)). In order to *extend* the tree towards q_{rand} while meeting the vehicle's motion constraints, new collision-free motions (states) are obtained by integrating Eq. (1) or (2), as shown in Algorithm 2. This procedure firstly requires finding the state q_{near}, that is the nearest to q_{rand} (line 2) (according to some distance metric over configurations).

Once the nearest motion has been found in the tree, the next step is to calculate the control input $u_{near\_to\_rand} = [v, w]$ that has to be applied in order to change the vehicle's state from q_{near} towards q_{rand} (line 3). With an RRT algorithm under geometric constraints the tree is iteratively expanded a geometric distance ϵ, in the case of an RRT algorithm under motion constraints the tree is expanded by applying an input u for a period of time Δt (line 4).

When motion constraints are present, there are different approaches that permit calculating $u_{near\_to\_rand}$. Two alternatives, presented by LaValle and Kuffner (LaValle and Kuffner, 2001), are either to randomly sample the control space, or to test a number of possible inputs and then select the one that generates the state q_{new} that is closest to q_{rand}. Figure 2 depicts an example of an RRT algorithm that solves a start-to-goal query for an AUV. The vehicle is assumed to be navigating at constant depth under motion constraints

[2]There exists another RRT variant called RRT-Connect that grows two trees, one from the start state and another one from the goal state. However, this is not possible when dealing with motion constraints, since merging both trees at a common configuration may become computationally intractable (Kuffner and LaValle, 2000).



(a) (b) (c)

Figure 2: Tree expansion of an RRT algorithm that considers the differential constraints described by Equation (1). (a) Given the start and goal states, a first random sample is used to generate the first expansion of the tree (q_{new}). (b) The expansion i^{th} of the tree. (c) A feasible path has been found from the start to the goal state.

described by Eq. (1), and the control input $u_{near\_to\_rand}$ has been randomly sampled.

3.3 Motion Planning Using Dubins Curves

There are several examples where a purely geometric path is first computed, and then it is transformed into a trajectory that is appropriate for the considered vehicle. For example, Yang *et al.* used an RRT algorithm to find a route of collision-free waypoints, which is interpolated with a cubic Bézier spiral (Yang and Sukkarieh, 2008). This seeks to convert the initial route into a smooth and feasible path for an unmanned aerial vehicle (UAV). As another example, Kuwata *et al.* proposed to expand an RRT by considering not only the vehicle's dynamics, but also its controller behavior (Kuwata et al., 2009).

Nonetheless, all those approaches, including the one presented in the previous section, have a major drawback: they do not guarantee any kind of optimality. This issue could be critical in underwater applications, which may require optimizing different criteria such as visibility (for gathering information), vehicle's autonomy, or even the safety associated with a path when navigating in close-proximity to nearby obstacles.

One option is to use the asymptotically optimal RRT (RRT*), which is a variant that guarantees convergence to a global optimum with respect to a given cost function (Karaman and Frazzoli, 2011). Its main difference with respect to other RRT-based methods is a routine that checks if reconnecting a new state's nearest nodes improves their associated cost (see Algorithm 3). For doing this, the RRT* algorithm requires a steering function that permits calculating such states (nodes) reconnection (line 9). In the case of systems under motion constraints, having such a function implies calculating the required input to dynamically evolve the system from a given state to a desired one. However, defining this function requires solving a two-point boundary value problem, which is in general a very difficult problem.

For the purposes of this paper, as an alternative to defining a steering function, we adopt the Dubins vehicle model (Dubins, 1957). Dubins geometrically demonstrated that, for a system that is only capable of traveling forward and with a constraint on the curvature of the path, the shortest path to connect any two configurations (states) $q_i, q_j \in SE(2)$, when no obstacles are present, can be obtained analytically by the combination of circular arcs and straight lines. Using three possible maneuvers as input, left (L), straight (S) or right (R), Dubins curves define six possible combinations: RSR, RSL, LSR, LSL, RLR, LRL. For a Dubins vehicle, at least one of these characterizes the optimal (shortest) trajectory between two states (see Fig. 3).

3.3.1 Dubins Curves for Constant-Depth Motions

For a torpedo-shaped AUV, Equation (1) describes an AUV that navigates at constant depth and with a constant surge speed v, where the maximum turning rate w_{max} establishes the minimum turning radius

Algorithm 3: extendRRT* (Karaman and Frazzoli, 2011)

Input:

T: tree of collision-free configurations.

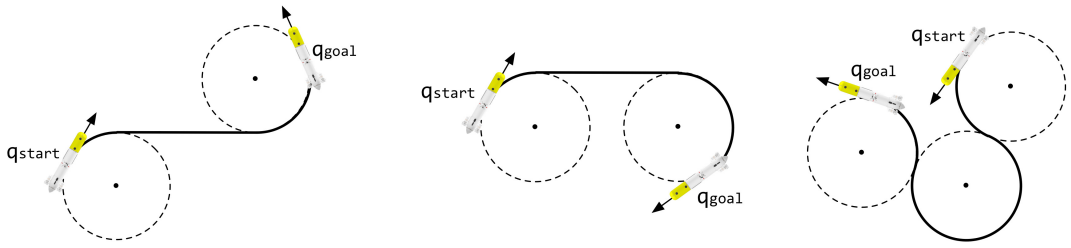
q_{rand}: state towards which the tree will be extended.

$\mathcal{C}$: C-Space.

Output:

Result after attempting to extend.

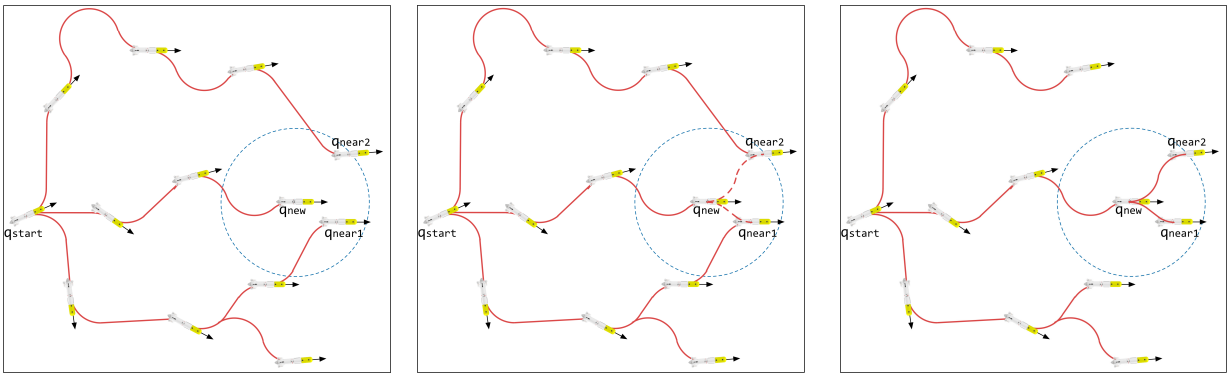
```
1  begin
2      q_near ← T.findNearestNeighbor(q_rand)
3      q_new, collision ←calcNewConf(q_near, q_rand, δ)
4      if collision = FALSE then
5          addNewNode(T, q_new)
6          Q_near ←findNearestNeighbors(T, q_new)
7          q_min_cost ←findMinCost(T, Q_near, q_new)
8          addNewEdge(T, q_min_cost, q_new)
9          reconnectNearNeighbors(T, Q_near, q_new)
10         return ADVANCED
11     else
12         return TRAPPED
```



(a) RSL (b) RSR (c) LRL

Figure 3: Examples of Dubins curves

r_{min} for a Dubins vehicle. This alternative formulation for the AUV motion constraints can be used with incremental search methods such as the RRT variants. In such a case, the control input $u_{near\_to\_rand}$, required to evolve the system from q_{near} to q_{rand} (Algorithm 2, line 3), can be replaced with Dubins curves. Furthermore, it is important to note that Dubins curves also work as a steering function, thus permitting to generate near-optimal paths with an RRT* algorithm. Figure 4 depicts an example of how this approach is used for reconnecting configurations near to q_{new}, and improving their associated cost. In this example the cost is assumed to be the path length. Further details about Dubins curves are found in references (Dubins, 1957; Shkel and Lumelsky, 2001).



(a) (b) (c)

Figure 4: RRT* algorithm reconnection using Dubins curves as steering function. (a) Once a new configuration q_{new} has been obtained, (b) the RRT* algorithm checks if the nearest configurations to q_{new} can be reconnected. (c) If such a reconnection decreases the cost associated with the nearest configurations, new paths are created while discarding previous ones. This allows to progressively improve the solution path.

3.3.2 Dubins Curves for Variable-Depth Motions

Different authors have used Eq. (2) to generate 3D motions for torpedo-shaped and propeller-driven AUVs. For instance, Heo *et al.* presented an approach to build a tree of collision-free and feasible configurations, over which a final path is found with the A* algorithm (Heo and Chung, 2013). The tree expansion is done by directly integrating Eq. (2), similarly as it was explained in Section 3.2. Although A* is a valid method to solve this problem, its main limitation is the need to discretize the 4-dimensional C-Space (x, y, z, ψ). This discretization resolution is especially critical for the exploration of unknown environments, in which a particular C-Space discretization may work for some scenarios, but might fail in others, *e.g.,* when navigating through variable-width environments.

An alternative to find better 3D solution paths is to use the RRT* algorithm. The previous section explained the use of Dubins curves as a steering function for 2D paths. The concept is now extended to 3D motions. As was mentioned before, a valid approximation is to decouple the motion in the horizontal plane from the heave (vertical) motion. This allows to directly deal with the former component as stated in Section 3.1.1, while treating the latter component as an additional constraint over an extended C-Space, $\mathcal{C} = SE(2) \times \mathbb{R}$. In this formulation, the path between two states for the 3D decoupled kinematics is the one that connects their components in the horizontal plane using Dubins curves, and their vertical components with a linear interpolation. The vertical motion must have a gradient no greater that the maximum ascending/descending AUV speed, thus meeting the vehicle's motion capabilities. The distance metric we use combines the lengths of the horizontal and the vertical paths as:

$$\rho\left(q_i, q_j\right) = \rho_h\left(q_i, q_j\right) + \rho_v\left(q_i, q_j\right), \tag{3}$$

where the first term corresponds to the length of the Dubins path between q_i and q_j in the horizontal plane,

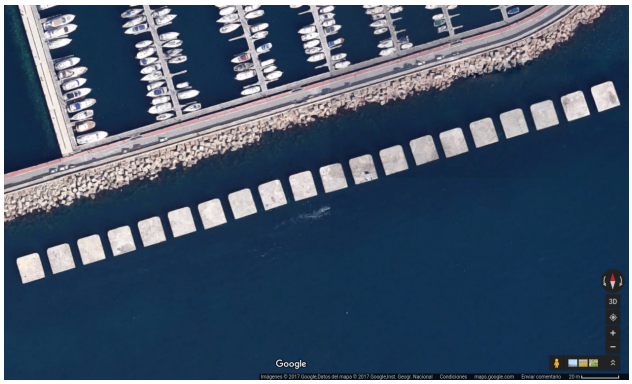


Figure 5: Breakwater structure composed of a series of concrete blocks (14.5m long and 12m width, separated by a four-meter gap) in Sant Feliu de Guíxols in (Spain). Image credit: Map data ©2017 Google.

and the second term corresponds to the distance in the heave (vertical) motion:

$$\rho_v\left(q_i, q_j\right) = \left|q_{iz} - q_{jz}\right|. \tag{4}$$

All this together, *i.e.,* the mathematical formulation, the use of the Dubins curves over a 4D space, and the corresponding metric over configurations, establish a computationally efficient approach to calculate 3D motions for torpedo-shaped AUVs under motion constraints. Note that in the 3D case, we no longer have an analytical solution for the shortest path between two poses. However, under the assumption that the AUV moves with a constant surge speed, the RRT* algorithm will, with the distance metric defined above, converge on the shortest path that satisfies the descent speed constraint.

Finally, in order to fully understand this approach, the next section presents solutions for different start-to-goal queries that have been obtained using both geometric and motion constraints. Moreover, it also demonstrates the advantages of using Dubins curves instead of using a standard RRT algorithm by integrating Equations (1) or (2).

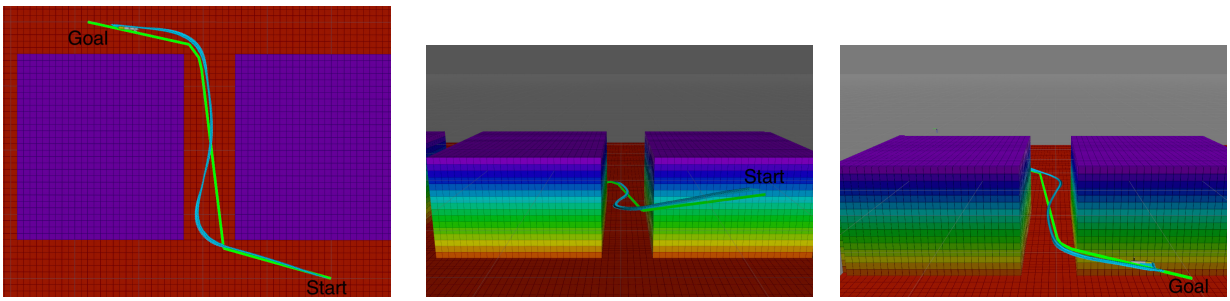
3.4 Simulation Results of Planning under Geometric and Motion Constraints

This section presents simulation results of the Sparus II conducting 3D start-to-goal missions in different scenarios. The solution paths have been calculated under both geometric and differential (motion) constraints. In all cases, the vehicle navigated in a pre-explored environment, *i.e.,* the obstacles and their locations were known. We present cases that emphasize the importance of taking the motion constraints into consideration when using non-holonomic AUVs.

3.4.1 Conducting 3D Missions under Geometric Constraints

Let us define a start-to-goal query where $q_{start} = [x_0, y_0, z_0, \psi_0]$ and $q_{goal} = [x_0 + distance, y_0, z_0 + depth, \psi_0]$, which means that the vehicle must conduct forward and vertical motions simultaneously. Furthermore, let us assume that the vehicle is required to navigate at a constant surge speed (v), as occurs in several AUV applications. Under such constraints, the time required to travel the horizontal distance can be calculated as $t_h = \frac{distance}{v}$, while the descending speed (d) can be adjusted up to a maximum value (d_{max}), which can be calculated as $d = \frac{depth}{t_h}$. This kind of task could also be tackled with geometric constraints, however there may be situations in which the resulting path cannot be followed by the AUV. For example, if t_h allows enough time to descend the desired vertical distance, *i.e.,* $d \leq d_{max}$, the vehicle will successfully complete the task, otherwise it may not reach the desired depth.

To validate the proposed motion planning framework, we present simulations of the Sparus II AUV conduct-



(a) Query top view (b) Query front view (c) Query back view

Figure 6: Virtual breakwater structure composed of two obstacles that create a narrow passage, where a start-to-goal query is defined as $q_{start} = [x_0, y_0, z_0]$ and $q_{goal} = [x_1, y_1, z_1]$. The solution path (in green) is calculated only taking into account geometric constraints. A simulated Sparus II AUV attempted to follow the path, however the resulting trajectory (in light blue) differs from the one calculated, thus leading to risky situations when conducting turning maneuvers.

ing autonomous missions in a virtual breakwater structure scenario. This simulated environment corresponds to a real-world structure that is located in the external and open area of a harbor (see Fig. 5). Figure 6 depicts the virtual breakwater structure over which a start-to-goal query solution and its execution are presented, where $q_{start} = [x_0, y_0, z_0]$ and $q_{goal} = [x_1, y_1, z_1]$, which means that the orientation was not taken into account. The 3D solution path was calculated using an RRT* algorithm under geometric constraints, and it was followed by the simulated Sparus II. It can be observed that, although the change of depth did not require a descending speed greater than the vehicle's capabilities (*i.e.,* $d < d_{max}$), the vehicle followed a trajectory that deviated from the planned path due to the fact that the planner did not consider the vehicle's motion constraints.

3.4.2 Conducting 2D Missions under Motion Constraints

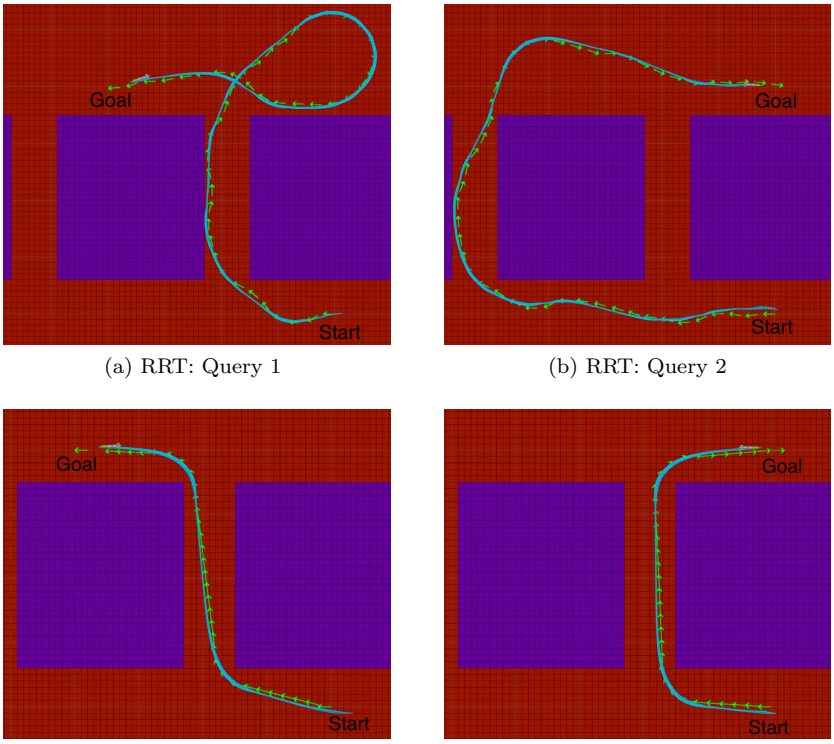
We evaluated the two alternatives described above to incorporate Eq. (1) into the motion planner: numerical integration and the use of Dubins curves. In order to compare these approaches, both were used to solve two different start-to-goal queries with a constant depth. For the first alternative, Figures 7a and 7b depict a simulated Sparus II AUV following solution paths that were calculated by an RRT algorithm that mathematically integrates Eq. (1). The RRT algorithm does not provide any guarantee for optimality.

The second alternative uses the RRT* algorithm and the Dubins curves, which work as a steering function equivalent to Eq. (1). Figures 7c and 7d depict how, with this approach, the simulated Sparus II AUV follows asymptotically optimal paths of minimal length. Although this latter approach generates better and more feasible solution paths, there are also some risky states; these were generated because the planner tried to provide short paths, which were close to nearby obstacles. This specific issue about the safety of the resulting path is addressed in detail in Section 4.

3.4.3 Conducting 3D Missions under Motion Constraints

Let us assume that the vehicle has to solve a task for which a start-to-goal query is defined as $q_{start} = [x_0, y_0, z_0, \psi_0]$ and $q_{goal} = [x_0, y_0, z_0 + depth, \psi_0]$; furthermore, let us consider that the vehicle has to navigate with a constant surge speed v and a descending speed up to a maximum value d_{max}.

Under such constraints, the motion planner must find a solution that combines turning and descending maneuvers to reach the final position and orientation. This problem can be solved with the extended Dubins curves approach that has been explained in previous sections. One possible solution, where the vehicle circumnavigates while descending to reach the desired depth and orientation, can be observed in Fig. 8.



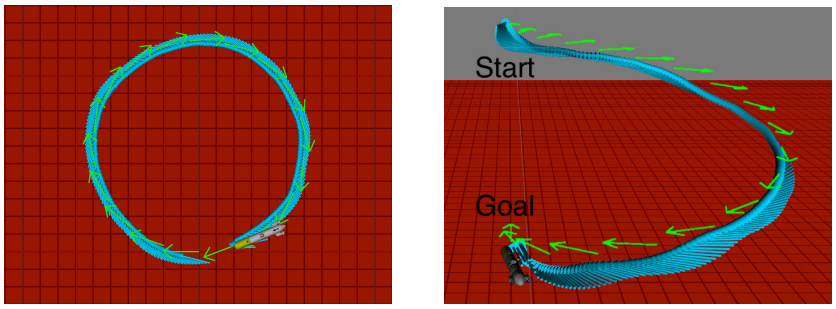
(a) RRT: Query 1

(b) RRT: Query 2

(c) RRT* with Dubins: Query 1

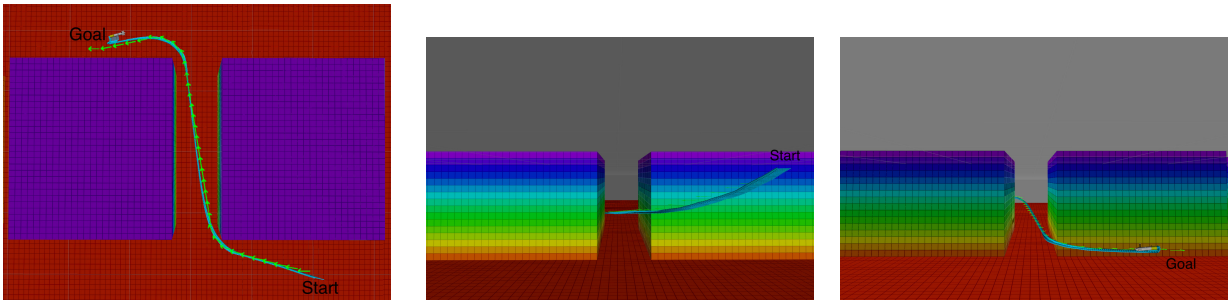
(d) RRT* with Dubins: Query 2

Figure 7: Virtual breakwater structure composed of obstacles that create narrow passages. Sparus II AUV follows the calculated paths. (a), (b) Two different start-to-goal query solutions that were calculated by an RRT algorithm under motion constraints are shown in green. (c), (d) The same start-to-goal queries were solved by an RRT* algorithm with Dubins curves as steering function. Solutions are shown in green.



(a) Top view (b) Perspective view

Figure 8: Start-to-goal query that is defined as $q_{start} = [x_0, y_0, z_0, \psi_0]$ and $q_{goal} = [x_0, y_0, z_0 + depth, \psi_0]$. The AUV is assumed to navigate at a constant surge speed v, and a descending speed up to d_{max}. The solution path (in green) is calculated with an RRT* that uses the extended Dubins curves, which incorporate the vehicle's 3D motion capabilities. The simulated Sparus II successfully describes a helical trajectory (in light blue) that follows the calculated path.



(a) Query top view. (b) Query front view. (c) Query back view.

Figure 9: Virtual breakwater structure composed of two obstacles that create a narrow passage. The solution path (in green) is calculated by an RRT* that uses the extended Dubins curves approach. A simulated Sparus II AUV describes a trajectory (in light blue), which does not differ significantly from the one calculated, even when conducting turning maneuvers.

Furthermore, this solution is near-optimal under the assumption of a constant surge speed.

Figure 9 presents another test over the virtual breakwater structure. Similarly as it occurred in the simulation presented in Fig. 6, the start-to-goal query requires the vehicle to completely change its 3D position, but this time it also specifies the desired AUV heading. This time the vehicle's trajectory during execution coincides with the planned path, even when conducting turning maneuvers.

There are, however, some aspects that have to be addressed in order to make this approach really useful for the proposed applications. The first of them is that the intended usage scenarios involve generally unexplored spaces, which means that a map is not available. A second aspect is the safety of the vehicle, which can be compromised when navigating in close proximity to the obstacles. These and other aspects, as well as the different strategies proposed to cope with them, will be discussed in detail in the next section.

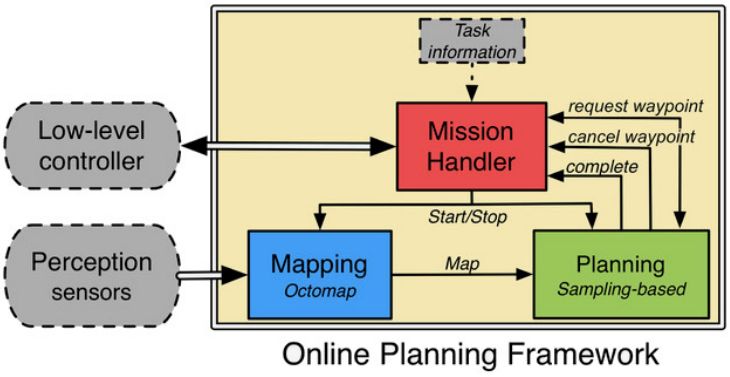


Figure 10: Framework for online AUV motion planning.

4 Framework for Online Motion Planning in Unexplored Environments

New and potential AUV applications that were presented in Section 1, establish most of the requirements and objectives for the research developed in this paper. One of those requirements is the capability of planning feasible paths that take into account the vehicle's motion constraints. Another requirement, and probably the most relevant, is the necessity of incrementally building a map of the surroundings, while simultaneously (re)planning the collision-free path to the goal. This characteristic would allow the vehicle to navigate through unexplored environments, as well as to overcome part of the navigation inaccuracy.

By using the foundations provided in Section 3 for planning AUV feasible paths, this section now presents a comprehensive planning framework to solve start-to-goal queries online for an AUV that operates in unexplored environments. The framework is composed of three functional modules. 1) A *mapping* module that builds an occupancy map of the environment using on-board perception sensors. 2) A *planning* module that generates safe (collision-free) and feasible paths online. 3) A *mission handler* that works as a high-level coordinator that exchanges information with the other two modules and the AUV's controllers. Figure 10 depicts the proposed framework. Since the framework was originally introduced in (Hernández et al., 2015; Hernández et al., 2016c), this paper will only discuss in detail the *mission handler* and the *planning* module, which have been considerably modified.

4.1 Mission Handler

The *mission handler* is in charge of controlling and coordinating the other modules (*mapping* and *planning*). It also verifies whether the AUV is prepared to start solving and conducting a task. To do so, this module communicates with other functional modules in the vehicle to verify that navigation data is correctly being generated and that the vehicle's low-level controllers are not conducting any safety maneuvers. After completing this initial checking stage, the *mission handler* sends a starting signal to the *mapping* module, and generates the star-to-goal query for the *planning* module.

In our previous works (Hernández et al., 2015; Hernández et al., 2016c), the *mission handler* received a discretized version of the solution path in a format of a set of waypoints, which were sent, one by one, to the low-level controller. However, this format required establishing a discretization value, which in case of being to too high could generate gaps in between that could be difficult to handle in case a replanning maneuver was triggered, or in case of being too low could affect the smooth and continuous execution of the path. This latter situation was mainly due to communication delays when requesting and receiving a new waypoint.

The *new mission handler* uses a different approach. This approach requires the *planning* module to provide the original path, *i.e.,* without discretizing it, which is then sent to a path-following controller. The iterative mapping-planning process now consists in projecting the vehicle's current configuration to the previous path. Such a projection establishes the initial configuration for every planning cycle. This avoids having initial states that are not close enough to any waypoint of a discretized path. Furthermore, it makes the planning framework itself more independent of the controller, eliminating the need to tune a discretization value. Finally, in case of a replanning maneuver, the *new mission handler* only has to provide a new valid path which must override the previous one, and in case of not finding a solution, it can make the vehicle stop.

4.2 Module for (Re)Planning Paths Online

The *planning* module is in charge of calculating a collision-free and feasible path for the AUV. For doing so, this module receives a query that is specified with a start configuration (q_{start}) and a goal configuration (q_{goal}), and other parameters, such as the available computing time and the minimum valid distance to the goal. Furthermore, given that the vehicle navigates in an unexplored (or unknown) environment, this module must continuously verify and repair (if necessary) the path from the vehicle's current configuration to q_{goal}.

To enable replanning and ensure safety during incremental exploration, this (planning) module contains a further modified version of the RRT* algorithm over what was presented in Section 3. Our extensions include an optimization function that combines collision risk and path length; a strategy to avoid unnecessary and expensive state checking routines; and the reuse of the last best known solution as a starting point for an anytime planning approach. Although these strategies were previously introduced by the authors in (Hernández et al., 2015; Hernández et al., 2016c), they are repeated in this paper for completeness and to provide further details.

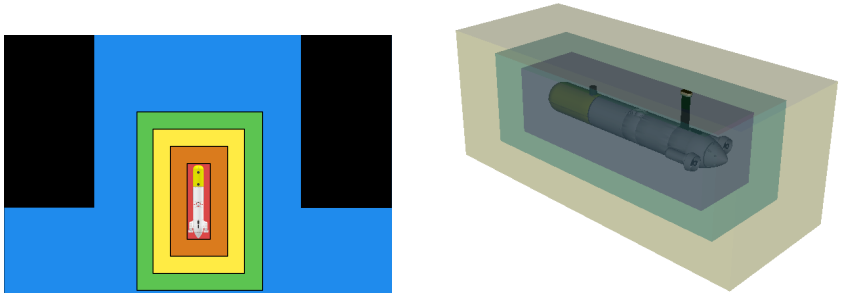
4.2.1 Planning Safe Paths using Risk Functions

Including motion constraints in the path planner may not be sufficient to avoid collisions with nearby obstacles. This is especially true when the vehicle is exposed to external perturbations, which do not permit the AUV to accurately follow the calculated path. Figures 7c and 7d, for instance, show feasible paths, but it leads the vehicle close to nearby obstacles, as the solution is optimized with respect to its length.

One alternative to cope with this kind of risky situation consists in estimating the probability of collision with nearby obstacles while, simultaneously, planning feasible paths. To do so, there are planning methods that allow considering different sources of uncertainty, however most of them are too computationally expensive, especially for applications that must meet online computation constraints, such as the ones presented in this work. One relevant contribution that seeks to overcome this computation limitation proposes to express the probabilistic obstacle avoidance problem as a Disjunctive Linear Program using linear chance constraints (CC) (Blackmore and Williams, 2006). This concept has also been extended to sampling-based methods such as the CC-RRT (Luders et al., 2010) and the CC-RRT* (Luders et al., 2013). Nonetheless, it is important to notice that the original concept and the subsequent extensions include some strong assumptions that cannot be made in underwater environments, such as considering convex obstacles. The authors of this paper have discussed and extended the aforementioned works by removing the convex obstacles assumptions, among other considerations for meeting online computation constraints (Pairet et al., 2018). However, such an approach is still limited to planning 2D motions.

Therefore, in order to guarantee that the vehicle safely navigates unexplored environments, our objective is to extend the planner to plan short, feasible paths that minimize the risk of colliding with the surroundings. This can be framed as defining an optimization objective for RRT* that combines the length and the safety of the path. The following sections present different approaches to achieve this.

a) *Path Length + Clearance:* a first and straightforward option is to maintain a minimum safe distance, or



(a) 2D risk zones. (b) 3D risk zones.

Figure 11: (a) Sparus II AUV navigating between two obstacles (black), where an example of the risk zones around the vehicle can be observed. The risk associated with each zone decreases as the zone is further from the vehicle's position. In this example, red means the zone with the highest risk of collision, while blue corresponds to the one with the lowest risk. (a) Extension of risk zones for 3D motions.

clearance, to the obstacles. For doing so, it is necessary to establish a weighted metric that combines path length and clearance in order to minimize detrimental effects in the path quality. This allows us to define the associated cost of each configuration when planning with an RRT* algorithm. However, the 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 (Tsianos et al., 2007).

b) *Risk Zones:* including clearance calculation is especially expensive for sampling-based motion planning methods, since it has to be performed for each sampled configuration and its intermediate steps when connecting to the others (*e.g.,* RRT* algorithm expansion). One alternative is to heuristically establish risk zones around the vehicle, as shown in Fig. 11. Red and blue represent the zones with the highest and the lowest risk of collision, respectively, while the others (orange, yellow and green) have associated risk values that decrease as the zone is further from the vehicle's position. Therefore, the risk associated with a given configuration q is determined by the innermost zone under collision. This risk association can be represented by Eq. (5), where n is the number of zones, and $zone_{i-1}$ is closer than $zone_i$ to the vehicle's position, and thus $risk_{i-1} > risk_i > \ldots > risk_n > 1$.

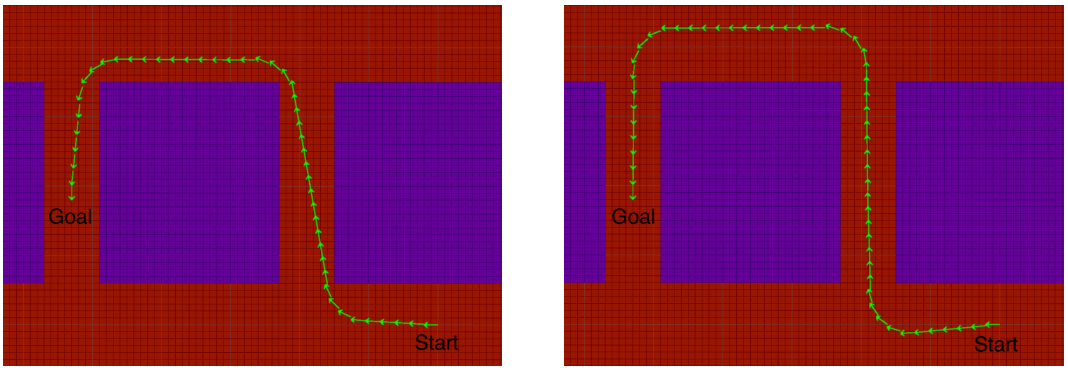
$$Risk(q) = \begin{cases} risk_1, & if\ Collision(zone_1) \\ risk_2, & if\ Collision(zone_2) \\ \ldots, & \ldots \\ risk_n, & if\ Collision(zone_n) \\ 1, & if\ not\ Collision(any\ zone) \end{cases} \tag{5}$$

In order to combine this function with the path length, the total accumulated cost associated with each configuration q_i in the tree produced by RRT* is calculated as the integral of risk with respect to distance along the path between the root of the tree, q_0, and q_i:

$$Cost(q_i) = \int_{q_0}^{q_i} Risk(q)\, \mathrm{d}q \tag{6}$$

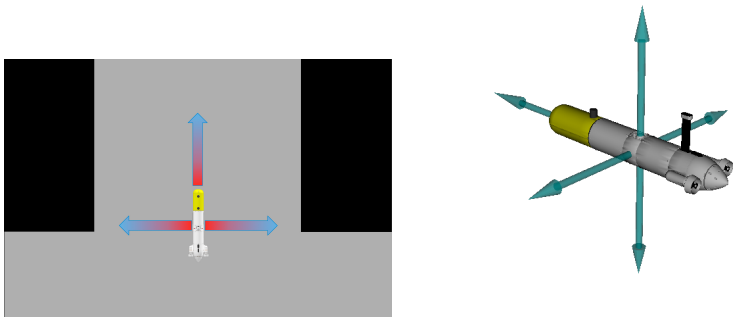
This cost function is then the optimization criterion for planning feasible and safe paths. A visual comparison between paths calculated using only the path length and the those with risk zones can be observed in Fig. 12.

c) *Risk Vectors:* the previous approach attempts to penalize those configurations close to nearby obstacles by specifying a risk value, which depends on the zone under collision. However, another alternative is to limit such evaluation to the directions defined by the possible maneuvers of the vehicle at the considered time. To do this, let us define three vectors in the straight and lateral motion directions. Now, instead of



(a) (b)

Figure 12: (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) 2D risk vectors. (b) 3D risk vectors.

Figure 13: (a) Sparus II AUV navigating between two obstacles (black). An example of the risk vectors, and how the risk decreases as moves away from the vehicle can be observed (red means high risk, while blue low risk). (a) Extension of risk vectors for 3D motions.

checking for complete zones, only points along those vectors will be checked for collision, assigning risk values with the same principle as done with risk zones, *i.e.,* moving away from the vehicle decreases the risk (see Fig. 13). This approach is here called *risk vectors*. It is computationally a less expensive alternative, given that checking collision for single points is more efficient than doing the check for zones (multiple points), especially when using Octomaps to represent the environment.

Both the risk zones and the risk vectors have also been extended for 3D motions. In the case of the risk zones, the areas around the vehicle become volumetric shapes of increasing size (*e.g.,* rectangular prisms, or cylinders), whereas with the risk vectors it is enough to add the corresponding vectors for the vertical motion (see Figs. 11b, 13b). These alternatives to quantify the risk associated with a path have been evaluated and compared with each other. The results of this analysis will be discussed in Section 4.4.

4.2.2 Opportunistic State Checking

When conducting an autonomous mission without initial information of the surroundings, the AUV is required to incrementally map and continuously (re)plan collision-free paths according to its partial knowledge of the environment. A significant number of configurations (sampled or obtained after expanding the tree)

are located in unexplored regions of the environment. In these situations, it is not only impossible but also unnecessary to attempt to determine if a configuration is at risk of collision. We propose to treat any configuration that is out of the explored area as safe (collision-free and minimum risk). This strategy was previously introduced by the authors as *opportunistic state checking* (Hernández et al., 2016c).

In this incremental mapping and planning approach, the tree expansion is periodically interleaved with updating the map, and such parts initially assumed as safe must be verified and discarded if found under collision as the vehicle explores the environment. The next section describes and analyses two alternative mechanisms to check and reshape the tree.

4.2.3 Reuse of the Last Best Known Solution

The strategy presented in this section is the third main characteristic of the proposed framework and its *planning* module. It allows an incremental and tree-based path planner, such as the RRT* algorithm, to replan or reshape the solution path according to the new environment information perceived during the mission. However, in order to fully understand the benefits associated with this strategy, this section firstly explains another alternative that was initially used, but later discarded due to its high computational cost.

Pruning the Tree

One alternative for dealing with partially known environments is to grow a single tree of collision-free configurations (as any RRT variant does), while periodically pruning the tree. This process allows discarding those branches that result under collision after updating the map, thus obtaining a valid solution at *anytime* that is requested. This strategy has been used with approaches based on RRT (Bekris and Kavraki, 2007) and RRT* (Karaman et al., 2011). At a first stage of this work, the *planning* module used a modified RRT* algorithm that was based on this pruning approach.

Like other RRT-based algorithms, the initially used variant consisted of two procedures, `sample` and `extend`. However, the former one had two main modifications (see Algorithm 4). The first one was intended to correct the tree according to the new elements discovered in the environment. To do so, and before sampling new configurations, the `updateTree` procedure traversed the tree using a depth-first search (DFS) algorithm to check if any node or edge was under collision (line 2). If a new collision was detected, the corresponding subtree (*i.e.,* those branches with nodes or edges under collision) was discarded. However, if the tree root was one of the nodes under collision, or if the path from the vehicle's current configuration to the root was not feasible, the *planning module* informed the *mission handler* to cancel the current path followed by the controller, and then it started again planning a new path from the vehicle's current position. This latter situation occurred because the tree root always corresponded to the configuration (or position) that the vehicle was moving towards.

The purpose of the second modification in the `sample` procedure was to make the RRT* behave as an *anytime* algorithm. To do this, if the new configuration q_{new} resulting from the tree expansion met the specified minimum distance to the goal (line 7), it was updated as the best available solution (line 8). Furthermore, if the *mission handler* had requested a new waypoint after concluding the tree expansion, and there was at least one available solution stored in the list (line 10), the planner selected the solution with the minimum associated cost, sent the *mission handler* the configuration connected to the root of that solution (line 13), and pruned the tree in such a way that the configuration sent became the new tree root (line 14). During this *pruning* process, subtrees connected to the former root (excepting the one that corresponds to the new root) were discarded. The `extend` procedure, on the other hand, remained as originally proposed in (Karaman and Frazzoli, 2011).

Reuse of the Last Best Known Solution

The previously explained approach used a tree of configurations that is periodically traversed, checked

Algorithm 4: SampleAnytimeRRT*

Input:
q_{start} : Start configuration.
q_{goal} : Goal configuration.
$\mathcal{C}$: C-Space.

```
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 ≠ TRAPPED then
7              if dist(q_new, q_goal) < ε_goal then
8                  updateBestSolution(q_new)
9                  solution_found ← true
10     if solution_found and waypoint_req then
11         result_path ←getBestSolution()
12         new_root ← result_path[1]
13         sendWaypoint(new_root)
14         pruneTree(new_root)
```

and pruned as the vehicle moves and explores the environment. The main objective was to preserve the information about collision-free areas and known paths, while discarding those that become invalid. One alternative for not conserving the whole tree, is 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.

In this iterative and *anytime* computation scheme, using the 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. This also permits to eliminate time-consuming pruning routines by avoiding checking subtrees, in which many of their configurations have probably become invalid because of the *opportunistic state checking* explained before. Although this strategy is the one used for the tests and results discussed in Section 5, a further comparison with the tree-pruning strategy will be presented in Section 4.4.3.

Finally, in order to illustrate the behavior of both strategies, *i.e.,* pruning the tree and the reuse of the last best known solution, Figure 14 describes a test scenario in which a vehicle simultaneously maps and plans collision-free paths in an unexplored environment. The vehicle is assumed to be equipped with a forward-looking range sensor of limited field of view (FoV) to navigate in an environment that resembles a breakwater. The structure is composed of a series of concrete blocks, which are separated by gaps. The task consisted in a start-to-goal query that required navigating from one side of the blocks to the other.

4.3 Pipeline for Online Planning Feasible and Safe Paths

Previous sections have presented three new features: an optimization function that combines collision risk and path length; a strategy to avoid unnecessary and expensive state checking routines; and the reuse of the last best known solution as a starting point for an anytime planning approach. These features endow an AUV with the capability to simultaneously map and plan feasible and safe paths online through unexplored environments. To fully understand how such characteristics work together, Algorithm 5 presents the execution pipeline to incrementally solve a start-to-goal query. This pipeline has allowed conducting autonomous missions for the intended AUV applications. For this reason, the newly proposed features represent the main contributions to the *planning* module presented in this section.

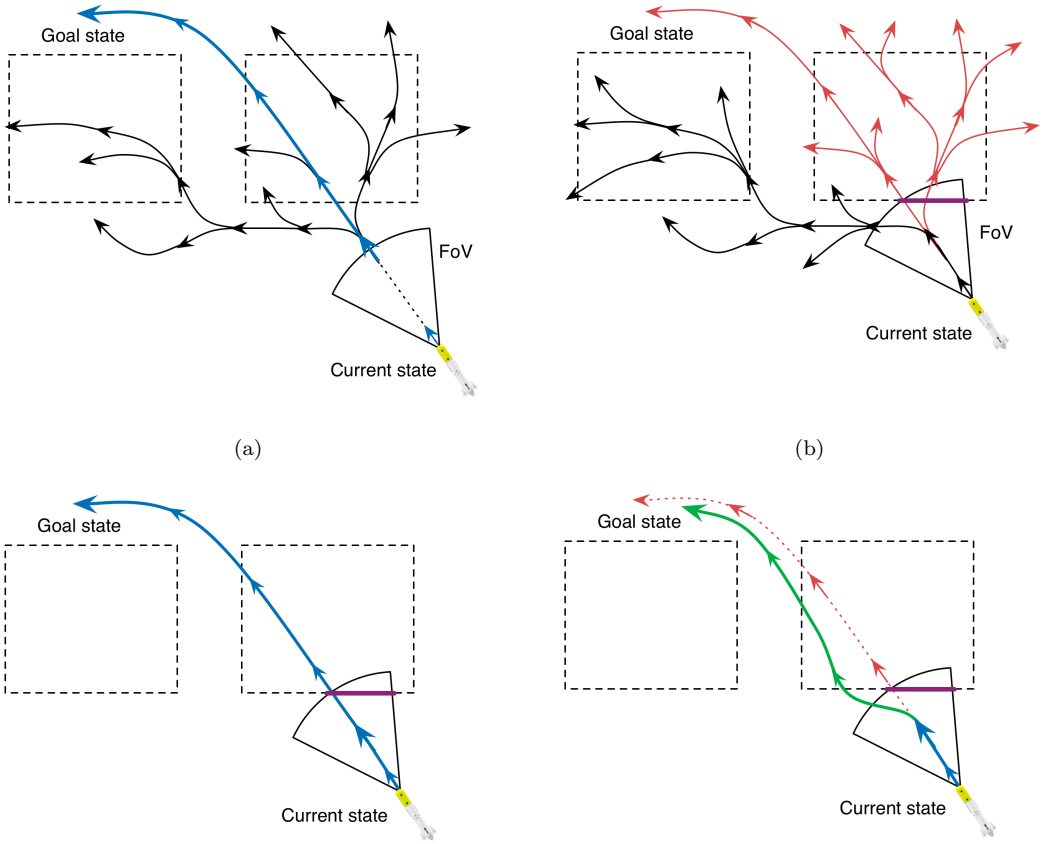


Figure 14: A test scenario in which a vehicle simultaneously maps and plans collision-free paths in an unexplored environment. The vehicle is assumed to be equipped with a forward-looking range sensor of limited FoV to navigate in an environment that contains a series of obstacles, which are separated by gaps. The task consists in a start-to-goal query that requires navigating from one side of the obstacles to the other. (a) During the first part of the task when no obstacle has been detected, the solution path corresponds to the shortest one from the vehicle's current state to the goal state (see nodes and branches in blue). (b) Once an obstacle is inside the vehicle's sensor FoV (in purple), different strategies could be used to guarantee that the solution path is still valid. With the tree-pruning strategy the tree is rechecked for collision. Nodes and edges that are in collision, as well as the subtrees attached to them, are discarded. This strategy may occasionally lead to loss of valuable information about reaching the goal, even if only part of the previous solution was affected (see parts of the tree in red). (c) An alternative is keeping the *last best known solution* (in blue), and then reusing it as an starting point in the next planning cycle, (d) which will discard the affected parts, while attempting to reconnect to other valid parts of the previous solution (in green).

Algorithm 5: incSolveStart2Goal(q_{start}, q_{goal})

Input:
q_{start}: Start configuration.
q_{goal}: Goal configuration.

```
1  begin
2      planner ← RRT*()
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())
```

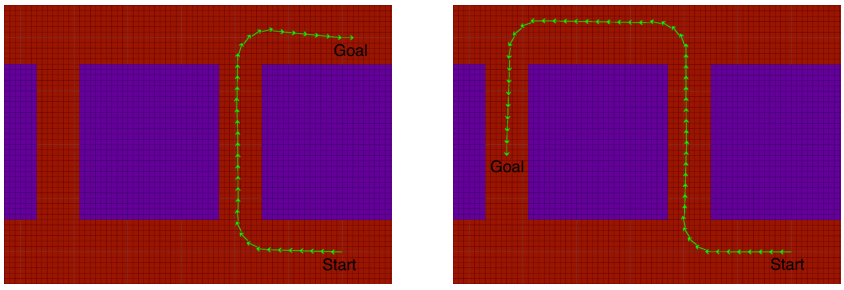
The pipeline presented in Algorithm 5 has as main input parameters the start and goal configurations (position and orientation) of the query to be solved. To initialize the incremental solving routine, the pipeline selects the RRT* algorithm as the planner that computes paths for a Dubins (or extended 3D Dubins) vehicle, sets an empty list as the *last best known solution*, and defines q_{new_start} as the starting configuration that will change as the vehicle conducts the mission (lines 2-4). To incrementally find a valid path to the goal (line 5), the pipeline's 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 or improve the 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. 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 requiring that a new path should be found from the 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 if the previous (*last best known*) solution is not in collision with an updated version of the map, the *planning* module will reuse 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).

4.4 Simulation Results of Incremental Mapping and Motion Planning in Unexplored Environments

To validate the proposed motion planning framework, this section presents simulations of the Sparus II AUV conducting autonomous missions in the virtual breakwater structure scenario. This simulated environment corresponds to a real-world structure that is located in the external and open area of a harbor (see Fig. 5). In this scenario, start and goal configurations were located in opposite sides of the shown breakwater structure, so that the Sparus II AUV had to move amidst the concrete blocks. All queries were defined to conduct missions at a constant depth, thus the motion was restricted to a 2D task.

For these tests, the vehicle was assumed to be equipped with a mechanically-scanning profiling sonar to



(a) Task1 (b) Task2

Figure 15: Solutions to start-to-goal queries using an RRT* algorithm with Dubins curves and risk functions. The solution paths maintain a safe distance from nearby obstacles (areas in purple).

perceive and detect the surroundings. The sonar was located to cover a scan sector in the horizontal plane along the vehicle's direction of motion. From the software perspective, the Sparus II used the component oriented layer-based architecture for autonomy (COLA2) (Palomeras et al., 2012), a control architecture fully integrated with the robot operating system (ROS). This software architecture also makes use of the open motion planning library (OMPL) that offers a convenient framework that can be adapted to specific planning problems (Sucan et al., 2012). Further details about the vehicle's hardware and software characteristics can be found in Appendix A.

4.4.1 Comparison of Risk Functions

Before evaluating the framework's capabilities over unexplored environments, this section firstly assesses and establishes the best alternative to estimate the risk associated with a path. This analysis is done over a fully known and mapped scenario. Once this has been determined, tests over an unexplored environment prove the advantages of using *opportunistic state check* strategy. Then, with this strategy, additional tests compare both the *pruning* tree and the *reuse of the last best known solution* approaches, thus allowing to demonstrate the latter one is the most efficient alternative for reaching an anytime computation approach.

Evaluation of the Risk Functions over Fully Mapped Environments: for these initial tests, an Octomap of the surroundings was assumed to be available *a priori*. Over this map, different start-to-goal queries were solved by using the alternatives to estimate the risk presented in Section 4.2.1. Figure 15 depicts some examples of the solutions paths obtained. It can be observed how the path maintains a safe distance from nearby obstacles (zones in purple), in contrast to what occurs when only path length criterion was considered (see Fig. 12).

Although all of the proposed approaches to estimate risk of the path can generate similar results, their associated computation times differ considerably. Figure 16 presents the average computation time required over 100 runs to solve $Task1$ and $Task2$ (shown in Fig. 15) by each approach. It can be observed that using only the *path length* is clearly the least expensive method, while *path length + clearance* and *risk vectors* are the most and least expensive, respectively, when including the risk of the path.

Evaluation of the Risk Functions over Unknown Environments: from Fig. 16, it can be concluded that the best computational alternative to include the risk associated with a path is the *risk vectors* approach. However, when dealing with unexplored environments, this approach may not permit estimating the risk correctly. In some cases, for instance, if an obstacle is not completely represented in the map so that the *risk vectors* do not coincide with the available partial information, the configuration can be incorrectly assumed as safe, while the *risk zones* will correctly estimate the risk (see Figure 17). For this reason, the best alternative when dealing with unexplored (unknown) environments is the *risk zones* approach.

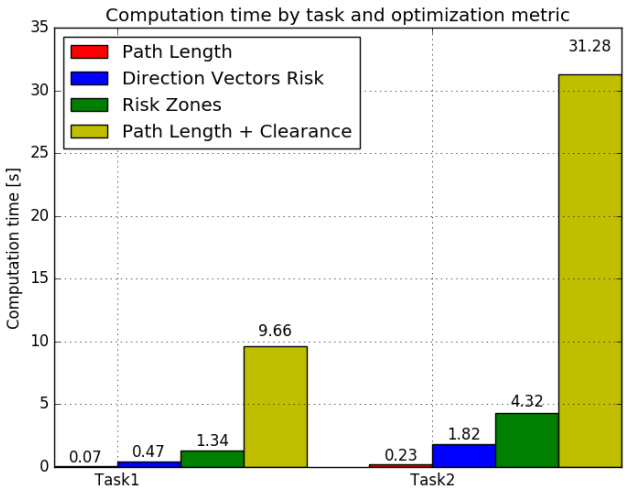
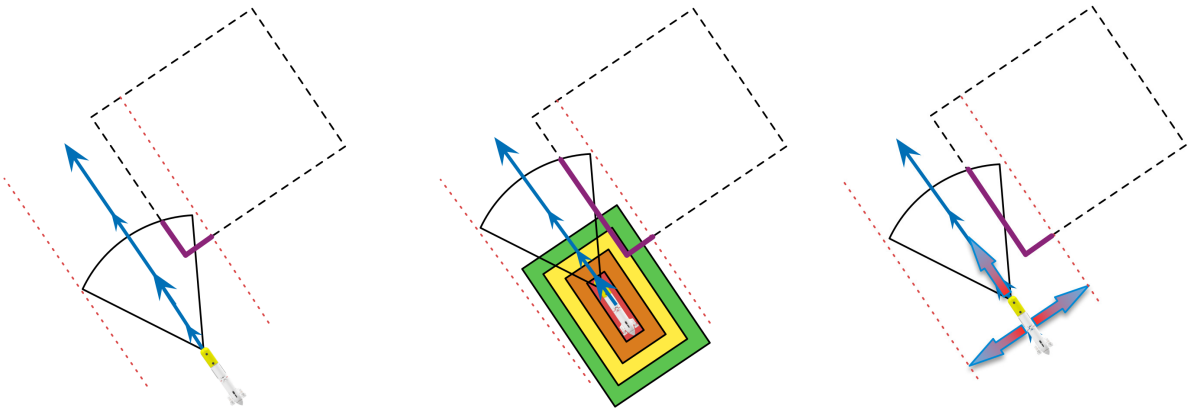


Figure 16: Average computation time, over 100 runs, required to solve $Task1$ and $Task2$ (Fig. 15) using different approaches to include risk of the path as the optimization criterion.

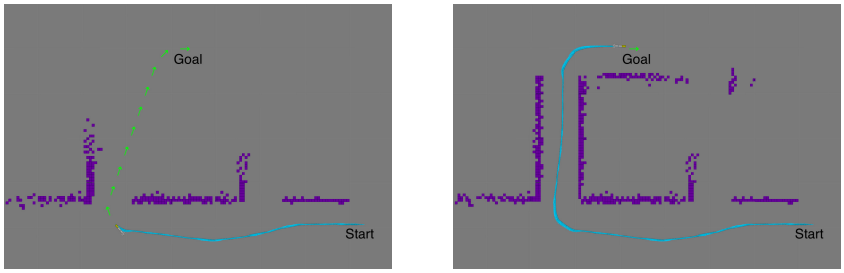


(a) (b) (c)

Figure 17: (a) A test scenario in which a vehicle incrementally discovers and maps an unexplored environment. Blue represents the path that the vehicle is following, and purple represents the partial information of the obstacle (dashed lines). (b) In cases in which obstacles are not completely represented in the map, the *risk vectors* may not coincide with the available partial information, thus assuming the configuration as safe. (c) *Risk zones*, on the other hand, correctly estimate the risk associated with the available obstacle information.

4.4.2 The Benefits of Opportunistic State Checking

Section 4.2.2 introduced the *opportunistic state checking* strategy as one of the mechanisms that can contribute to overcome this situation. To validate its efficiency, a start-to-goal query was solved and simulated 15 times, but now without assuming any previous map of the breakwater structure scenario (Fig. 18 presents the execution of one those tests).



(a) (b)

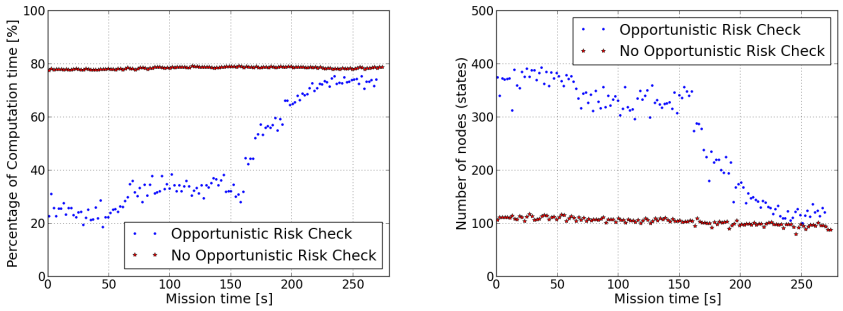
Figure 18: A simulated Sparus II in an equivalent virtual scenario of the breakwater structure. The vehicle starts a mission by submerging to a specified depth. It then maps and solves a start-to-goal query simultaneously. (a) Equipped with a scanning profiler, it incrementally builds a representation of the environment, while continuously reshaping the solution path, (b) to finally approach to the specified goal configuration.

The solutions for queries such as the one presented in Fig. 18 were obtained with and without the *opportunistic state checking* strategy. Although in both cases the framework succeeded in conducting the task, Fig. 19a demonstrates that without it almost 80% of the total computation time is dedicated to risk checking routines over the whole mission. When using *opportunistic state checking*, its associated computation time increases as the environment is progressively explored, but is still computationally more efficient. This is especially noticeable when a mission does not require exploring and mapping the entire 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, but also improves the workspace exploration and the path quality. Fig. 19b presents the effect of using *opportunistic state checking*, in which it can be observed how a low computation time in the first part of the mission, when the environment has not been explored, also enables the construction of a tree with a higher number of nodes.

4.4.3 Anytime Motion Planning in Unexplored Environments

Section 4.2.3 presented a *pruning* tree scheme and the *reuse of the last best known solution*. Both are alternative replanning approaches to make the framework capable of providing valid solutions at *anytime*. In order to prove the latter one is the best option, the execution pipeline presented in Section 4.3 has been used with both approaches to run different simulations.

Two different simulated scenarios were used to compare the *pruning* tree scheme and the *reuse of the last best known solution*. The first simulated scenario is the breakwater structure. Over this scenario, two different start-to-goal queries were tested (see Fig. 20). After executing these missions 10 times, it was observed that the number of successful attempts is greater and the mean of cancelled maneuvers (over the total successful missions) is smaller when *reusing the last best known solution* with *opportunistic state checking* (see Table 2). Such cancelled maneuvers correspond to situations where the calculated path was unfeasible, thus requiring a new valid path. It is important to notice that the execution of a simulated mission in the breakwater structure scenario can last on average 2 minutes. Given that the planning time was set to $1s$, each of these missions required approximately 120 executions of the proposed planner, for a total of 240 for both $Task1$ and $Task2$. In the case of the simulated missions in the underwater canyon, the missions can last on average 4 minutes. In this case, the planning time was set to $1.2s$, which means approximately 288 executions of the



(a) (b)

Figure 19: Incidence of *Opportunistic State Checking* approach when solving query presented in Fig. 18. (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, the computation time 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.

Table 2: Comparison of solving the tasks shown in Figs. 20 and 21 using *pruning-tree* scheme and the *reuse of the last best known solution*. The execution of each simulated mission in the breakwater structure scenario lasted on average 2 minutes, with a planning time set to $1s$, requiring, approximately, 120 planning executions, for a total of 240 for both $Task1$ and $Task2$. In the case of the simulated missions in the underwater canyon, the missions lasted on average 4 minutes, with a planning time set to $1.2s$, thus requiring approximately 288 planning executions.

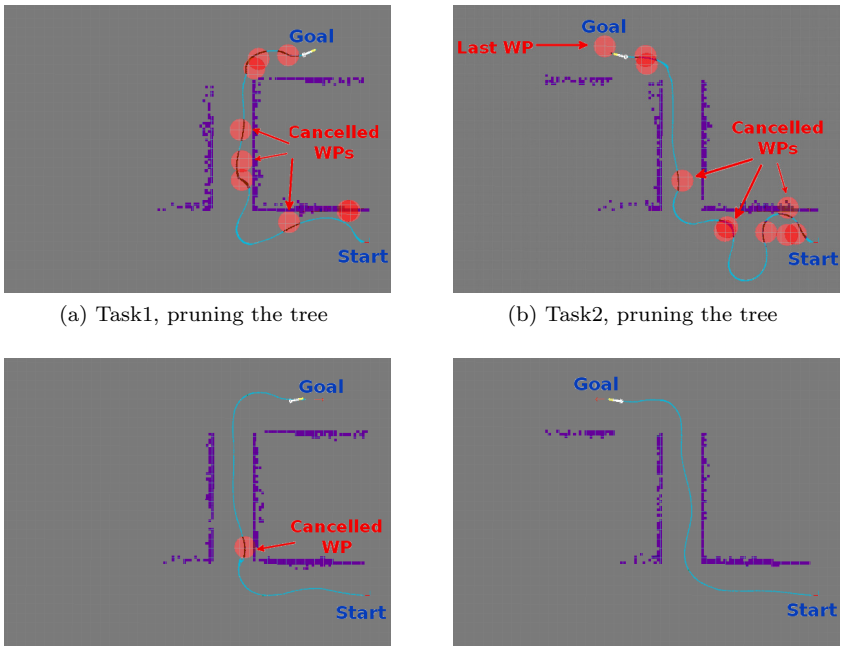
	Virtual Scenario 1: Breakwater Structure (Fig. 20)				**Virtual Scenario 2: Sea Rocks (Fig. 21)**	
	Task 1 (10 missions)		Task 2 (10 missions)		Task 3 (10 missions)	
	# Successful missions	Mean of cancel. maneuvers	# Successful missions	Mean of cancel. maneuvers	# Successful missions	Mean of cancelled maneuvers
Pruning Tree	7	8.85	7	6.42	6	6.6
Reuse Last Sol.	10	0.3	9	0.1	7	1.14

planner. We consider that 500 executions is a representative sample of the planner performance.

The second test scenario resembles a natural environment composed of rocky formations, which create an underwater canyon (see Figs. 21a, 21b). One start-to-goal query was defined in a way that the vehicle was required to travel through the canyon (see Figs. 21c, 21d). Once again, it was observed that the number of successful attempts is greater and the mean of cancelled maneuvers is smaller in the case of *reusing the last best known solution* (see Table 2).

5 Results in Real-World Scenarios

This section presents an extensive evaluation of the proposed framework in real-world scenarios through both simulations and in-water trials. Experiments were conducted with the Sparus II AUV (see Appendix A for hardware and software details). These experiments are separated into three different scenarios: 1) planning constant-depth paths to move through artificial marine structures; 2) planning constant-depth paths to move through natural marine formations; and 3) planning variable-depth paths to move through confined marine environments. It is important to notice that the these scenarios do not include long-distance missions, and they were not conducted in areas where the currents that can affect drastically the motion of the vehicle. The following sections will discuss in detail the results obtained for each of these scenarios.



(a) Task1, pruning the tree

(b) Task2, pruning the tree

(c) Task1, reuse last solution

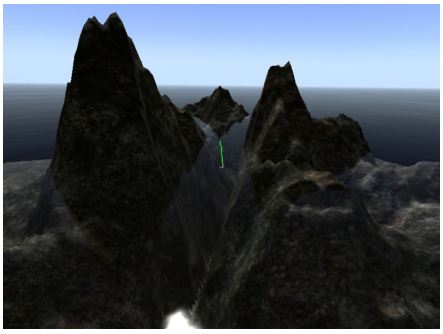
(d) Task2, reuse last solution

Figure 20: Comparison of the *pruning* tree scheme and the *reuse of the last best known solution* when solving start-to-goal queries with the *pruning-tree* scheme and the *reuse of the last best known solution* approaches. Cancelled waypoints (WPs) are presented as circles in red, and correspond to WPs that may lead the vehicle to a collision, thus requiring the path to be completely replanned.

5.1 Planning AUV Paths in Artificial Marine Structures

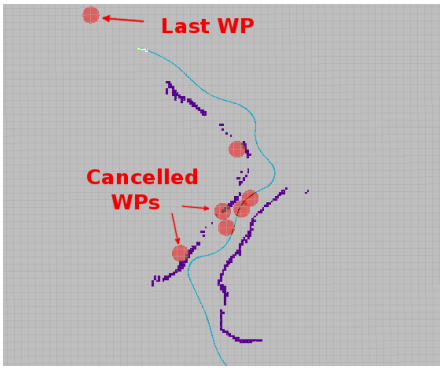
Nowadays, there are a wide range of man-made marine constructions, from large commercial harbors to offshore platforms. These artificial structures often have a regular and known shape, which could be used to preplan trajectories to travel through them. However, as was explained in Section 1, there are different factors such as low visibility and limited navigation accuracy, which make it difficult to correctly follow such precalculated paths. An example of this kind of structure is the breakwater mentioned in previous sections. An example mission could be one that makes an AUV to move amidst the concrete blocks. The origin for the north-east-depth (NED) inertial system is defined by a latitude-longitude coordinate, which can be obtained from Google Maps ©. Likewise, a start-to-goal query can be specified in such a way that the AUV is required to traverse the blocks from the outer to the inner area, and vice versa. Although these goals could be defined in advance, this preliminary information does not include unknown obstacles that the vehicle may encounter during the mission execution. However, the proposed approach has the capabilities to avoid colliding with such unexplored obstacles, and in case that such obstacles do not allow to complete the start-to-goal mission, the vehicle will stop.

If a map of the surroundings is not available a priori, the proposed framework requires the vehicle to be equipped with exteroceptive sensors. These sensors allow detecting the surroundings to incrementally create a map online. For this test scenario, a set of four echosounders and one mechanical-scanning imaging sonar were located within the vehicle's payload (front) area, all of them pointing in the horizontal plane (see Fig. 22). Three of the echosounders were separated by 45°, with the central one looking forward and parallel to the vehicle's direction of motion, while the fourth one was perpendicular to the central one (see Fig. 22a). The imagining sonar was set up to cover a scan sector in the vehicle's direction of motion (see Fig. 22b). Since both sensors cover the horizontal plane, missions with this configuration were executed at a constant depth. For more details about the complete Sparus II's hardware configuration, the interested reader is

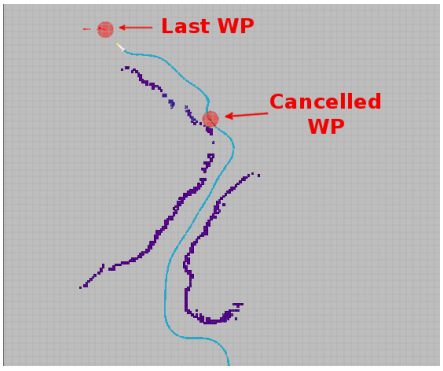


(a) Sea rocks scenario

(b) Navigating through the canyon

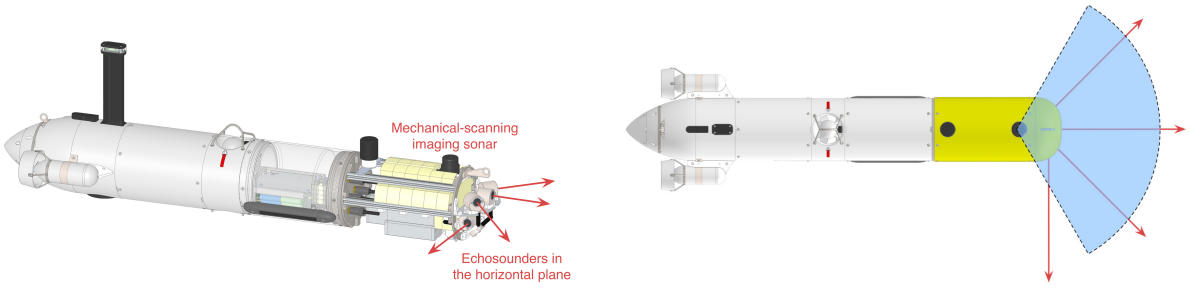


(c) Pruning the tree



(d) Reuse last solution

Figure 21: Comparison of the *pruning* tree scheme and the *reuse of the last best known solution* when solving start-to-goal queries. The simulated scenario resembles a natural environment. Cancelled waypoints (WPs) are presented as circles in red, and correspond to WPs that may lead the vehicle to a collision, thus requiring the path to be completely replanned.



(a) (b)

Figure 22: Exteroceptive sensors configuration for the Sparus II AUV to conduct autonomous missions in the breakwater structure. (a) The vehicle was equipped with four echosounders and one mechanical-scanning imaging sonar, all of them pointing in the horizontal plane. (b) Top view, echosounders beams direction and imagining sonar scan sector.

referred to Appendix A.

During the first simulation and in-water trials conducted over this test scenario, the Sparus II AUV only used the echosounders to perceive the environment. Nevertheless, the four echosounders beams were insufficient to rapidly establish the validity of the path along the direction of motion, which triggered multiple replanning maneuvers. Although the vehicle was capable of finding a solution path, it required multiple attempts to complete one full successful mission. Furthermore, this limitation generated unexpected and non-desired trajectories. To avoid such situations, the Sparus II used both the echosounders and the mechanical-scanning imaging sonar to incrementally build the map. For each beam position within the scan sector, the imaging sonar provided an array of intensities. From these intensity values, those that were over a specified threshold represented the obstacles detected. Once those values had been identified, they were transformed into ranges or distances to the obstacles. However, since the imaging sonar has a vertical beamwidth of $35°$, the maximum range was limited to $10m$ to avoid false-positive detections from the sea bottom. The echosounders, on the other hand, were set up with a maximum range of $20m$, since they have a smaller beamwidth of $10°$.

This payload configuration was used in different sea trials, and allowed the vehicle to traverse the breakwater structure with great repeatability. To demonstrate this, the Sparus II conducted real-world missions that included multiple and successive start-to-goal queries. Figure 23 depicts one of these missions. 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. For this mission, the Sparus II navigated with a constant surge speed $u = 0.5m/s$, a maximum turning rate $r_{\max} = 0.3rad/s$, and at a constant depth of $2m$.

The mission was composed of three different and consecutive start-to-goal queries, and assumed the environment as unexplored. Both the NED frame origin and the different goals coordinates when obtained from Google Maps © 2017. For the first query, the vehicle was initially in the inner area of the breakwater structure, and the goal was a coordinate in the outer area. Hence, the only possible solution path required the AUV to navigate through one of the four-meter gaps (see Fig. 23a). The second query was set back in the inner area but in a different coordinate, thus requiring to move through to one of the gaps once again (see Fig. 23b). Likewise, the third and last query was defined to conclude the mission in the outer area (see Figs. 23c).

Along the mission, the Sparus II did not surface to obtain GPS fixes. Figure 23d depicts the vehicle's trajectory and the Octomap built overlapped with a satellite imagine of the breakwater structure. Although the Octomap is coherent with respect to the real-world structure, there are some differences due to the

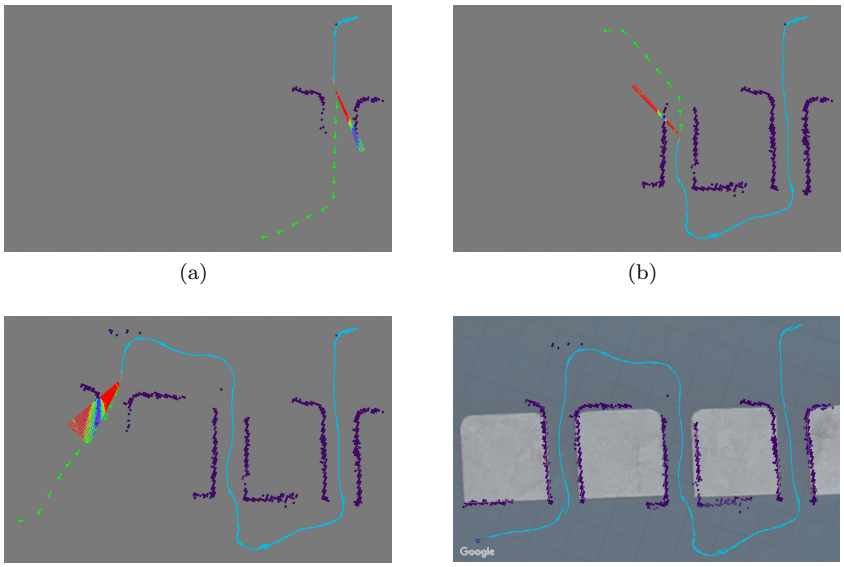


Figure 23: The Sparus II AUV used the motion planning framework to navigate through a breakwater structure without a preliminary map of the surroundings. (a), (b), (c) The mission required traversing the breakwater structure multiple times by solving successive start-to-goal queries. (d) The vehicle's trajectory is drawn in light blue, while the map built with the imaging sonar data is presented in purple. This information is overlapped with a satellite image (Google Maps © 2017).

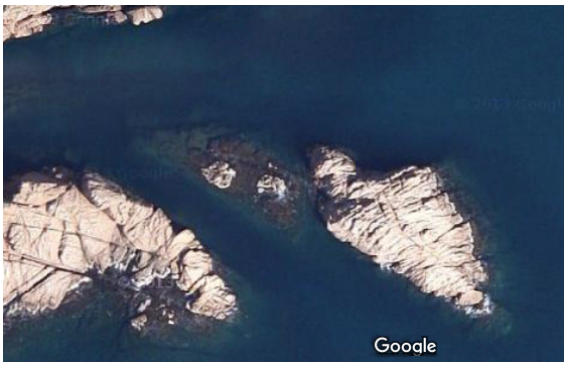
	Breakwater Structure Tests	
	Task 1: single crossing (8 attempts) # Successful attempts	**Task 2: three consecutive crossings (4 attempts)** # Successful attempts
Pruning Tree	2	0
Reuse Last Sol.	6	2

Table 3: Comparison of solving two different in-water tasks in the breakwater structure using *pruning-tree* scheme and the *reuse of the last best known solution*. Task 1 consisted in crossing once the breakwater structure by moving through one of the gaps between the concrete blocks. Task 2 consisted in crossing three consecutive times, each time using a different gap (see Fig. 23). For both tasks, the *reuse of the last best known solution* has a higher rate of success, thus proving to be a more effective approach.

accumulation of navigation error. Yet this did not prevent the AUV from successfully completing the mission. Although Figure 23 represents a successful mission, it is important to analyse the repeatability. Table 3 presents a comparison of the rate of successful attempts while using *pruning-tree* scheme and the *reuse of the last best known solution* for two different in-water tasks: single crossing, and multiple consecutive crossings. It can be observed that the *reuse of the last best known solution* proves to be a better strategy, as was already demonstrated in simulation (see Table 2).

5.2 Planning AUV Paths in Natural Marine Formations

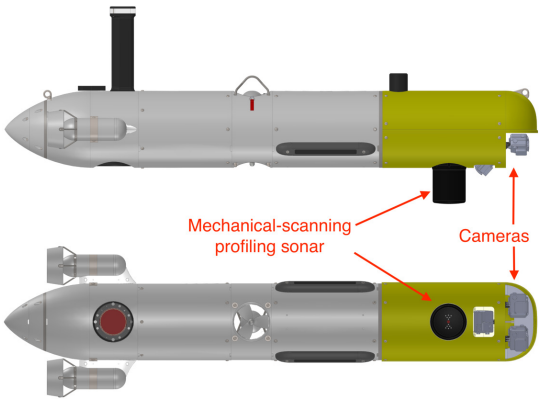
Although the mission in the breakwater structure represents a valid application example, it is also true that the obstacles in it, *i.e.,* the concrete blocks, have regular shapes. Their vertical walls, for instance, were correctly detected by a mechanical-scanning imaging sonar, despite its great beamwidth. Natural environments, on the other hand, present less regular and more challenging scenarios. To test the proposed framework under more challenging conditions, we therefore also conducted autonomous missions over a real-



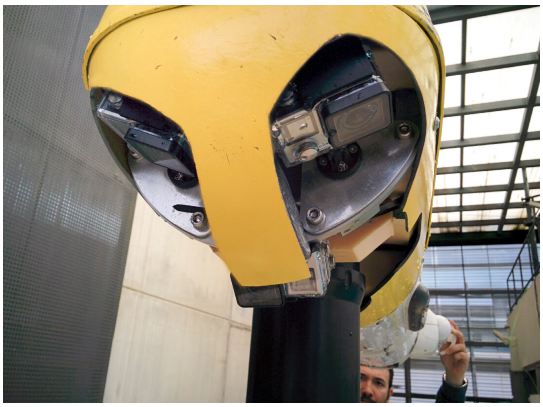
(a) Satellite view. Image credit: Map data ©2017 Google.

(b)

Figure 24: The test scenario consists of rocky formations. (a) They create an underwater canyon that can be observed in the satellite image. (b) From the inner area (*i.e.,* from the coastline towards the open sea) the canyon can be clearly observed.



(a)



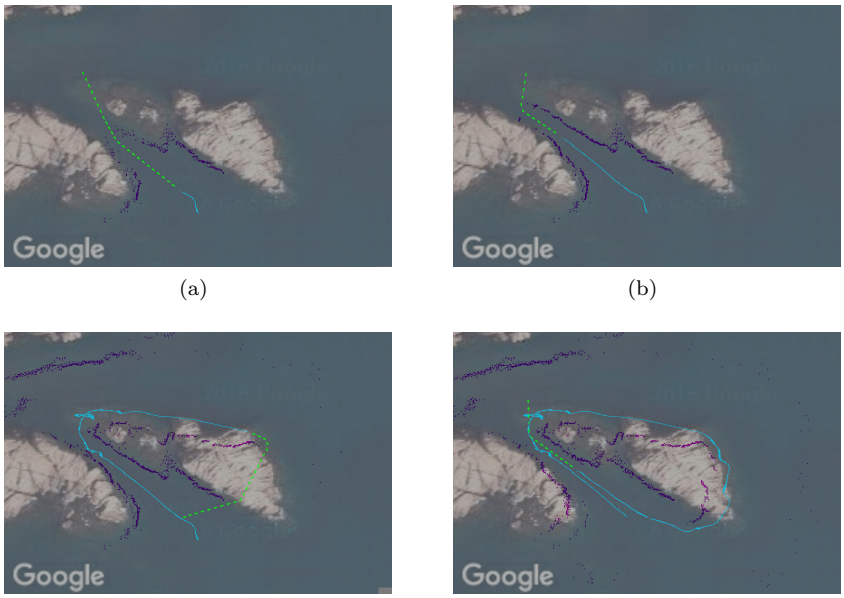
(b)

Figure 25: Exteroceptive sensors configuration for the Sparus II AUV to conduct autonomous missions in the underwater canyon. (a) The vehicle was equipped with three GoPro^{TM} Hero 4 Black edition cameras and one mechanical-scanning profiling sonar. (b) Real-world vehicle's payload.

world natural environment. The testing area is also located in Sant Feliu de Guíxols (Spain), and contains rocky formations that create an underwater canyon (see Fig. 24).

For this scenario, the Sparus II AUV used a mechanic-scanning profiling sonar, which has a smaller beamwidth of 1–2°. This sensor, which covered the horizontal plane, permitted not only to perceive the obstacles shape with more accuracy, but could also be set to a higher maximum range without being affected by false-positive detections from the bottom. The AUV was also equipped with a set of three GoPro^{TM} Hero 4 Black edition cameras. They were used to gather the optical images required to create a 3D reconstruction of the surroundings. The cameras were positioned systematically to ensure the highest possible coverage, where two of them were placed in a downward configuration at an angle of 20°, while the third camera was positioned forward looking at 40° (see Fig. 25).

Two different start-to-goal queries were defined using coordinates obtained from Google Maps ©. The first query required the Sparus II AUV to traverse the canyon towards the shore. Then, the second query goal was chosen on the outside of the rocky formation in such a way that the vehicle had to circumnavigate the outer rock.



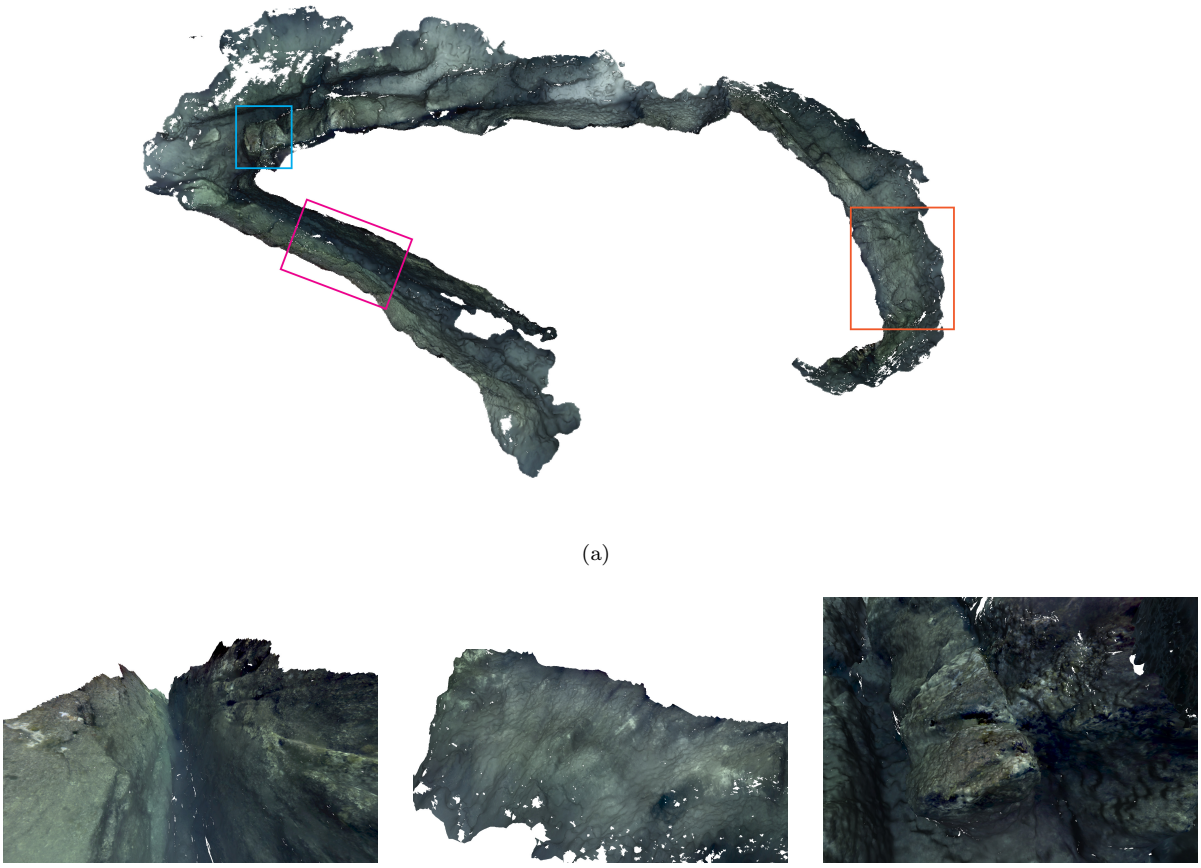
(a) (b) (c) (d)

Figure 26: The Sparus II AUV used the motion planning framework to navigate through an underwater canyon without a preliminary map of the surroundings. (a), (b) The first start-to-goal query required the Sparus II traversing an underwater canyon. (c) For the second query, the vehicle circumnavigated the outer rock. (d) The AUV partially repeated the first start-to-goal query in order to close the loop and to obtain overlapped images. Vehicle's trajectory and Octomap overlap a satellite image (Map data ©2017 Google).

Figure 26 depicts one of the inspection missions conducted in the underwater canyon. For this mission, the Sparus II navigated with a constant surge speed $u = 0.5m/s$, a maximum turning rate $r_{\max} = 0.2rad/s$, and at a constant depth of $5m$. The Sparus II not only created a map of a complex and unexplored environment, but also planned a collision-free path simultaneously and incrementally. The map and the vehicle's trajectory are shown overlapped with a satellite image. In the initial part of the mission, *i.e.,* when the vehicle traverses the canyon for the first time, the map coincides with the satellite image (see Fig. 26b); however, disparities can be clearly observed after some time (see Figs. 26c, d). Such differences are due to the accumulation of error in the navigation system that depends on the doppler velocity log (DVL), which may provide incorrect data when navigating over rocks, as occurred in this test scenario. Despite this, the vehicle succeeded in conducting the mission because both the map and the path are created online, which permits correcting or adjusting them even when moving in previously visited areas (see Fig. 26d when accessing the canyon for a second time).

In this kind of exploration, the data that is gathered along the mission can be used to create a more detailed survey of the area. In this particular mission, for instance, the images were extracted and were used to build a photo-realistic 3D model, which is depicted in Figure 27a. This 3D reconstruction allows us to better understand complex environments by generating arbitrary user-defined views (see Figs 27b–d). For more details about these 3D reconstructions, the interested reader is referred to (Hernández et al., 2016a; Hernández et al., 2016b).

As it was also explained for the in-water tests in the breakwater structure, in the underwater canyon the start-to-goal queries could be tentatively defined in advance, but this information does not include obstacles or environment limitations that the vehicle may encounter during the mission execution. For this particular case, for instance, the canyon's minimum aperture could be estimated as $8 - 10m$ if it is observed from the surface. However, the canyon gets narrower as the vehicle navigates deeper (see Figure 27b). The proposed framework has the capabilities to deal with this kind of unexplored environment, and it will stop the vehicle



(a)

(b) (c) (d)

Figure 27: (a) Top-down view of the textured 3D model with marked areas additionally depicted in magenta, orange and blue, which correspond to generated views of: (b) the underwater canyon; (c) the external side of the underwater rocky formation; (d) underwater rocks.

in case that the information detected along the mission avoids reaching the desired goal.

The main challenges associated in the experiment depicted in Figure 26 are not immediately apparent. The factor that most likely had the biggest impact on the success was the correct payload setup. Once the mechanic-scanning profiling was selected as the perception sensor, and it was correctly mounted and setup, it was possible to conduct four successful missions out of six attempts.

Finally, in order to understand the complexity associated with this mission, Figure 28 presents a visual comparison between the vehicle's trajectory, estimated by its dead-reckoning (DR) system, and the cameras' trajectory, estimated during the reconstruction (green and red, respectively). While both trajectories have a similar shape, it can be clearly observed how the one derived from the cameras is more realistic according to the rock observed in the surface (the rocky formation does not create a vertical wall, which means that the vehicle may have moved further from the visible part of the rock when navigating at $3m$ deep), while the one estimated by the AUV's dead-reckoning system seems to be colliding with the rock. This is mainly due to the accumulation of errors in the navigation system, as was already mentioned before.

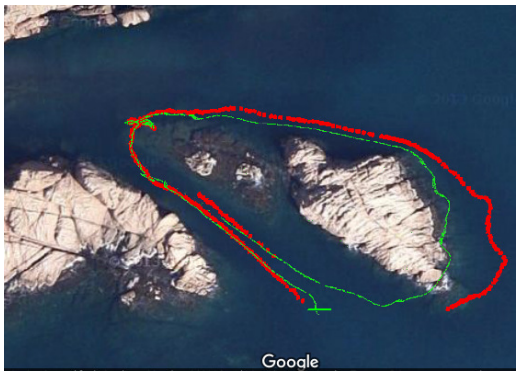
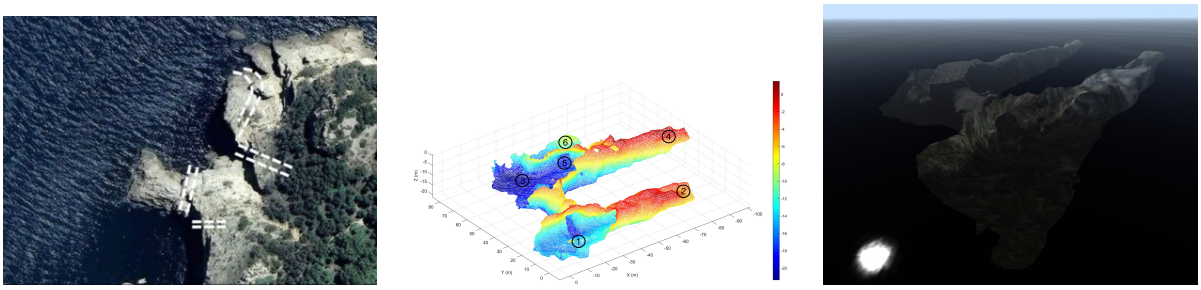


Figure 28: Vehicle's trajectory (green) calculated by its dead-reckoning system and the cameras trajectory (red) estimated by the image-based reconstruction. Both trajectories are shown overlapped with a satellite image of the test scenario (Map data ©2017 Google).



(a) (b) (c)

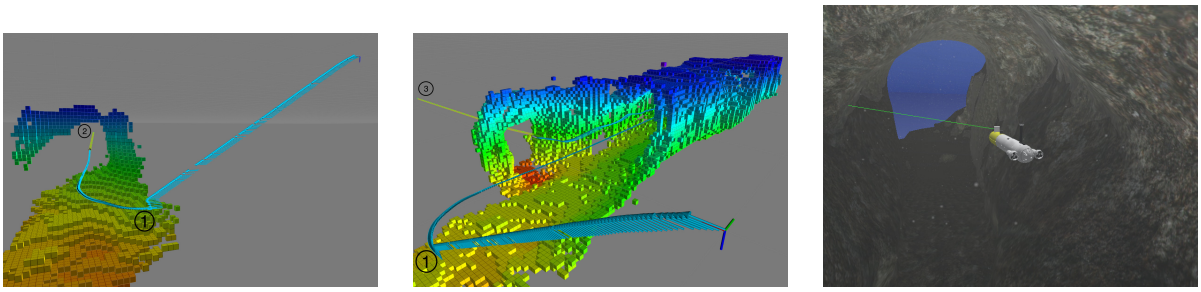
Figure 29: (a) The underwater caves complex "Coves de Cala Viuda" is located in the L'Escala, Spain (Lat: 42.10388, Lon: 3.18255). It consists of three single-branch caves and several tunnels. Their approximate positions are marked with dotted white lines. (b) Meshed map using the SNDC algorithm (Campos et al., 2013). Over the map, the approximate locations of six different goals have been marked in black. Images credit: Mallios *et al.* 2015 (Mallios et al., 2015; Mallios et al., 2017). (c) Mesh map added into UWSim as a simulation environment.

5.3 Planning AUV Paths in Confined Natural Environments

In this section we will present experimental results for a mission that requires 3D motion. One example of a mission that require a 3D motion planner is to move through confined natural environments (*e.g.,* underwater caves and tunnels). Some technical aspects that make it more difficult to conduct autonomous missions in these environments, include the navigation error associated with incorrect data provided by the DVL when traveling over rocky surfaces.

In dealing with this latter aspect, Mallios *et al.* presented the exploration of an underwater caves complex (see Fig. 29a), in which a diver guided the Sparus AUV to gather environment acoustic information (Mallios et al., 2015). To do so, the vehicle was equipped with two mechanically scanning imaging sonars to cover the horizontal and vertical planes. Such data was used to prove a scan-matching algorithm over a simultaneous localization and mapping (SLAM) framework, which allows reducing and bounding the AUV navigation error. This survey provides an extensive dataset that includes not only the sonars' raw data, but also a detailed meshed map (see Fig. 29b) (Mallios et al., 2015; Mallios et al., 2017). For more details about this survey, the interested reader is referred to the cited references.

Although the prior work is considered a significant step towards the exploration of underwater confined



(a) (b) (c)

Figure 30: Simulated mission in the caves complex. In (a), the first start-to-goal query guided the AUV closer to the complex location. The second query required the vehicle to navigate through the first single-branch cave. The third start-to-goal query guided the AUV to the second single-branch cave's entrance. In (b) the vehicle's trajectory appears in light blue, and the remaining of the calculated path towards waypoint (3) appears in yellow. This query required the vehicle not only to find a way out of the first cave, (c) but also to traverse a tunnel that connects the entrances of both single-branch caves.

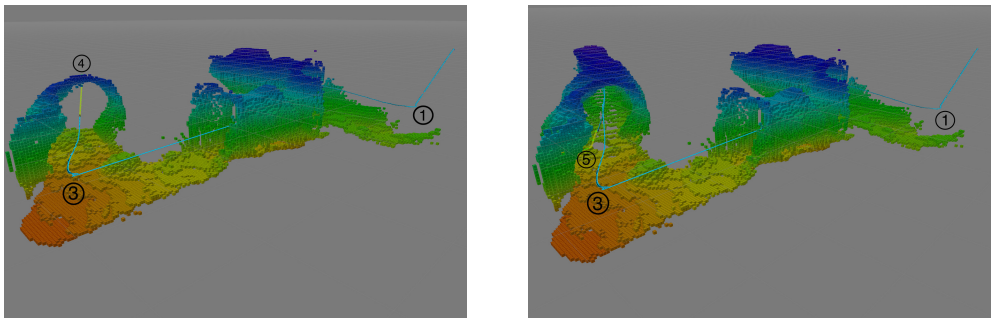
environments, there are still significant challenges in conducting such exploration missions fully autonomous. Our work contributes to coping with some of those aspects. The proposed framework, for instance, seeks to endow AUVs with additional capabilities that eliminate the need for a diver's guidance in the near future. This section presents a simulated mission over the meshed map of the cave complex. The mission consisted in solving six consecutive start-to-goal queries, where the Sparus II was not provided with any preliminary environment information, thus requiring to incrementally map and (re)plan the path to the goals. Figure 29b depicts the meshed map and the approximate locations of six different query goals.

To conduct this test mission, the meshed map was added into UWSim as a 3D simulation environment (see Fig. 29c). To perceive the surroundings, the AUV's payload was assumed to be equipped with an additional link capable of rotating 120° around the vehicle's z axis. Over this link, a forward-looking multibeam sonar was mounted. The sonar had 240 beams distributed over a total aperture of 120° around the vehicle's x axis. With this payload configuration, the simulated Sparus II was capable of gathering 3D range data of the environment along its direction of motion.

The first start-to-goal query was defined to guide the AUV closer to the caves complex location. From there, the second query sought to explore the first and biggest single-branch cave. This meant navigating to the end of the cave. Figure 30a depicts part of the execution of these two queries, including the AUV's trajectory, the calculated path, and the goals. From the second query's goal configuration, a third query was defined to take the Sparus II closer to the second single-branch cave's entrance. To accomplish this part of the mission, the AUV not only had to find a way out of the first cave, but also had to traverse a tunnel that connects the entrances of both single-branch caves. Figure 30b depicts part of the execution of this query.

Once the Sparus II AUV had crossed the tunnel, the fourth query was defined to navigate to the end of the second single-branch cave. Once this was accomplished, the fifth query was set to guide the AUV close to a third cave's entrance. Figure 31 depicts part of the execution of these two queries.

Finally, Figure 32 presents different views of the whole mission execution, where both the vehicle's trajectory and the Octomap built along the mission can be observed. Although a real-world trial was not conducted, mainly due to technical and mechanical aspects that have to be further developed, this test evaluates all the aspects that have been covered throughout this paper and represents a clear advance towards fully autonomous inspections. It is important to mention that once the values for the surge speed has been defined for each of the start-to-goal queries, this mission through the caves complex was simulated 4 times, each execution required at least 20 minutes, with a planning time of $1s$, which means at least 4800 planning queries. The mission did not fail in any of these simulations.Additional information about this test, such as



(a) Sparus II traversing the second cave

(b) Sparus II on its way back to the third cave's entrance

Figure 31: Simulated mission in the caves complex. (a) The fourth start-to-goal query required the vehicle to navigate through the second single-branch cave. (b) The fifth query was defined to guide the vehicle close to a third cave's entrance. This required the planner to find a way out of the cave. In both images, the vehicle's trajectory appears in light blue.

the distances traveled by the vehicle and the speed for each query, can be found in Appendix B.

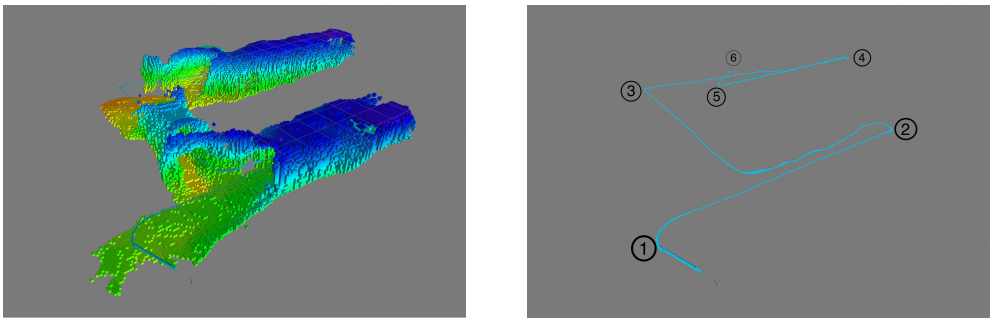
6 Conclusions and Future Work

In this paper, we present an approach to endow AUVs with the capabilities required to move through unexplored environments. To do so, we propose a computational framework that is composed of three functional modules: a mission handler, a mapping module, and a planning module. The first of them works as a high-level coordinator between the other two modules and the AUV's low-level controllers. The mapping module uses an Octomap to incrementally build a 3D map of the surroundings, which represents free, occupied, and unexplored areas. Finally, the planning module calculates feasible and safe paths for AUVs that operate under motion constraints.

To guarantee that the resulting paths are feasible, we considered the vehicle's motion capabilities. This was accomplished by using a tree-based sampling planning method and Dubins curves to characterize the AUV constant-depth motions. Such a formulation was also extended to support fully 3D motions. Furthermore, we presented and assessed multiple alternatives to evaluate the risk associated with the solution path. As a result, we established a function that combines the length and safety of the path into a single optimization objective.

In order to allow an AUV to navigate environments without an initial map, we proposed two complementary strategies that permit to efficiently reshape the solution path as the environment is incrementally explored. The first one is to *reuse the last best known solution* to avoid the need of pruning the tree of configurations. The second one is to *opportunistically check* the states' risk of collision. Both strategies sought to meet online computation limitations, thus providing a computationally efficient alternative to fully considering the sensing and motion uncertainty.

To validate the proposed approach and its characteristics, we presented an extensive evaluation in several scenarios. This included simulated and in-water trials in different environments such as artificial marine structures, natural marine structures, and confined underwater environments, as well as the autonomous survey replanning for coverage gap filling and target inspection. These experiments validated the framework's capabilities for 2D and 3D tasks in which the vehicle can gather different kind of data from the surroundings. To highlight potential applications, some of these experiments included a photo-realistic 3D reconstruction of the traveled area.



(a) Perspective front view

(b) Perspective front view

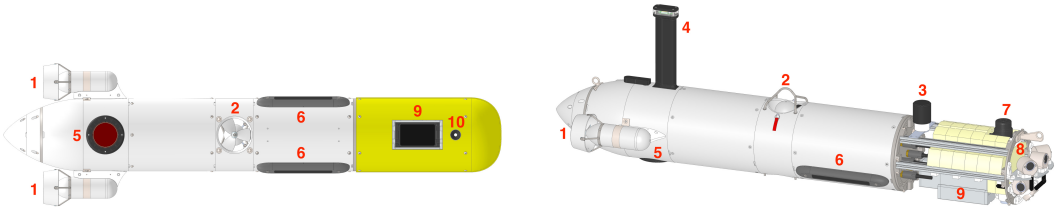
(c) Perspective back view

(d) Perspective back view

Figure 32: Simulated mission in the caves complex. The whole mission was composed of six start-to-goal queries that were executed consecutively.

This work has also provided a good overview of the main challenges and limitations that have to be overcome in order to conduct fully autonomous underwater missions. A critical one is the computational power available in underwater vehicles. While in other autonomous systems, such as terrestrial vehicles, a common solution for this limitation is the use of dedicated computers for the different functional modules (*e.g.,* mapping, planning, control, etc.), in underwater vehicles the volume dedicated for on-board computers is generally fixed, and commonly includes only one computer. In most of the commercial AUVs, such a computer is capable of estimating its position to follow predefined survey paths, but it is not intended for mapping and (re)planning paths online. Furthermore, the increase of the number of on-board computers may also limit the vehicle's autonomy. Therefore, the computational complexity, in time and space, is a critical requirement to be considered when designing and developing software control architectures for these kind of applications.

Another important challenge for conducting autonomous underwater missions is the limited accuracy of the exteroceptive and interoceptive sensors. For the former group, it was a considerable effort to conduct in-water trials in the underwater canyon. Different sensor configurations were tested for getting the best possible representation of this kind of environment. With respect to the interoceptive sensors, the DVL, which is the main sensor used for navigation, is considerably affected when navigating over rocky environments. The difficulties encountered with both kind of sensors provide an idea of the potential challenges that will have to be addressed in order to conduct in-water missions in complex scenarios such as the underwater caves complex. Furthermore, these technical issues are mainly responsible for the reduced effectiveness when moving from simulation to in-water trials. Many of the failed attempts were caused by navigation or perception errors, which are not directly related to the planning framework. Such sensor errors include noisy DVL measurements when navigating over rocky environments, thus providing incorrect vehicle's velocities estimation, magnetometer bias increase when navigating amidst the concrete blocks, and false-positive detection of obstacles which is mainly due to the imaging sonar aperture.



(a) Bottom view (b) 3D view

Figure 33: Sparus II AUV: bottom and 3D views. Different hardware parts can be observed, including the thrusters (1,2), the acoustic modem (3), the Wi-Fi and GPS (4), as well as interoceptive and exteroceptive sensors such as the DVL (5), side-scan sonar (6), mechanically-scanning imaging sonar (7), single-beam echosounders (8), multibeam sonar (9), and an optical camera (10).

There are several directions we plan to explore in future work. First, we plan to further develop the high-level planning layer to automatically select goal configurations or goal regions that the vehicle should navigate towards. The order of the exploration, the strategy to efficiently cover a desired area, dealing with dead ends are decisions all need to be addressed by this high-level planning layer. Second, we plan to explore how we can more accurately account for and exploit the vehicle's dynamics during planning while preserving the fast planning times.

Appendices

A Experimental Platform: The Sparus II AUV

The experimental validation of the framework proposed in this paper was carried out with the Sparus II AUV. This underwater vehicle is a torpedo-shaped AUV with hovering capabilities, which has been designed and developed at the Underwater Vision and Robotics Research Center (CIRS)[3]. The vehicle is rated for depths up to $200m$, and is equipped with three thrusters; two of them are located in the back, and are used for motion on the horizontal plane; the third one is located in the middle, and is dedicated to vertical motion. This implies that the AUV can be actuated in surge, heave and yaw degrees of freedom (DOF). Furthermore, the Sparus II 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.

The Sparus II AUV also has communication devices such as an acoustic modem for underwater communication with other vehicles or surface stations (*e.g.,* by using an ultra-short baseline (USBL) system), and a Wi-Fi antenna that can be used when the AUV is at surface. Moreover, the vehicle includes a configurable payload area in the front, which contains a set of exteroceptive sensors to perceive and detect the surroundings. This latter group of sensors can be modified according to the mission's requirements, and may include optical cameras, single-beam echosounders, mechanical-scanning (profiler and imaging) sonars, multibeam sonars, etc. Figure 33 depicts different views of the Sparus II AUV, including one where a possible payload configuration can be observed.

As far as software is concerned, the Sparus II AUV is controlled through the component oriented layer-based architecture for autonomy (COLA2) (Palomeras et al., 2012), which is a control architecture that is completely integrated with the robot operating system (ROS). Besides operating aboard real robots, COLA2

[3]CIRS is part of the Computer Vision and Robotics Institute (ViCOROB) in Girona (Spain)

can interact with the underwater simulator (UWSim), which can import 3D environment models. UWSim is the result of a joint effort between our group and the group from University of Jaume I (Castellon, Spain), thus allowing to simulate the Sparus-II's sensors and dynamics with high fidelity (Prats et al., 2012; Carreras et al., 2018). Furthermore, the use of ROS allows for easy integration of third party tools, such as the open motion planning library (OMPL) which offers a convenient framework that can be adapted to specific motion planning problems (Sucan et al., 2012).

B Start-to-goal Queries over the Underwater Caves Complex "Coves de Cala Viuda".

Detailed information about the underwater caves complex "Coves de Cala Viuda" can be found in (Mallios et al., 2015; Mallios et al., 2017). However, this appendix provides the exact values of the start and goal configurations, as well as the vehicle's speed used in the simulation test presented in Section 5.3 (see Figure 32). For the first start-to-goal query $q_{goal_1} = [20.0, 0.0, 12.0, 0.0]$. From there, the second query used $q_{goal_2} = [18.0, 55.0, 8.0, 1.57]$. For these two queries, the planner limited the vehicle to navigate at a constant surge speed $v = 0.5m/s$, a maximum ascending speed $d_{ascend} = 0.2m/s$, and a maximum descending speed $d_{descend} = 0.18m/s$.

From the second query's goal configuration, a third query was defined with $q_{goal_3} = [80.0, 25.0, 16.0, 0.23]$. For this query, the planner limited the vehicle to navigate at a constant surge speed $v = 0.3m/s$, while keeping the previous maximum ascending/descending speeds ($d_{ascend} = 0.2m/s$ and $d_{descend} = 0.18m/s$). This decrease of v permitted the vehicle to conduct maneuvers with a smaller turning radius, especially required to leave the first cave.

The fourth query was defined with $q_{goal_4} = [60.0, 87.0, 9.0, 1.82]$. To do so, the speed constrains were kept as established for the previous query. Once this was accomplished, the fifth query was set with $q_{goal_5} = [82.0, 47.0, 17.0, 0.0]$. This latter query required the planner to find a way out of the second cave, which is considerably narrower than the first one. To cope with this latter situation, the planner used a lower surge speed, $v = 0.1m/s$, which allowed the vehicle to turn back with a smaller turning radius.

Acknowledgments

J.D. Hernández, E. Vidal, N. Palomeras, and M. Carreras have been supported by the EXCELLABUST and ARCHROV Projects under the Grant agreements H2020-283TWINN-2015, CSA, ID: 691980 and DPI2014-57746-C3-3-R respectively.

M. Moll and L.E. Kavraki have been supported supported in part by NSF IIS-1317849 and NSF IIS-1718478.

References

Alvarez, A., Caiti, A., and Onken, R. (2004). Evolutionary Path Planning for Autonomous Underwater Vehicles in a Variable Ocean. *IEEE Journal of Oceanic Engineering*, 29(2):418–429.

Arinaga, S., Nakajima, S., Okabe, H., Ono, A., and Kanayama, Y. (1996). A motion planning method for an AUV. In *Symposium on Autonomous Underwater Vehicle Technology*, pages 477–484.

Bekris, K. E. and Kavraki, L. E. (2007). Greedy but Safe Replanning under Kinodynamic Constraints. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 704–710.

Bingham, B., Foley, B., Singh, H., Camilli, R., Delaporta, K., Eustice, R., Mallios, A., Mindell, D., Roman, C., and Sakellariou, D. (2010). Robotic tools for deep water archaeology: Surveying an ancient shipwreck with an autonomous underwater vehicle. *Journal of Field Robotics*, 27(6):702–717.

Blackmore, L. and Williams, B. (2006). A probabilistic approach to optimal robust path planning with obstacles. *American Control Conference*, page 7 pp.

Caldwell, C. V., Dunlap, D. D., and Collins, E. G. (2010). Motion planning for an autonomous Underwater Vehicle via Sampling Based Model Predictive Control. In *MTS/IEEE OCEANS*, pages 1–6.

Campos, R., García, R., Alliez, P., and Yvinec, M. (2013). Splat-based surface reconstruction from defect-laden point sets. *Graphical Models*, 75(6):346–361.

Cao, J., Cao, J., Zeng, Z., and Lian, L. (2016). Optimal path planning of underwater glider in 3D dubins motion with minimal energy consumption. In *MTS/IEEE OCEANS*, pages 1–7. IEEE.

Carreras, M., Hernández, J. D., Vidal, E., Palomeras, N., Ribas, D., and Ridao, P. (2018). Sparus II AUVA Hovering Vehicle for Seabed Inspection. *IEEE Journal of Oceanic Engineering*, 43(2):344 – 355.

Carroll, K. P., McClaran, S. R., Nelson, E. L., Barnett, D. M., Friesen, D. K., and William, G. N. (1992). AUV path planning: an A* approach to path planning with consideration of variable vehicle speeds and multiple, overlapped with, time-dependent exclusion zones. In *Symposium on Autonomous Underwater Vehicle Technology*, pages 79–84.

Cheng, C.-T., Fallahi, K., Leung, H., and Tse, C. K. (2010). An AUVs path planner using genetic algorithms with a deterministic crossover operator. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2995–3000.

Dubins, L. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3):497–516.

Englot, B. and Hover, F. S. (2013). Three-dimensional coverage planning for an underwater inspection robot. *The International Journal of Robotics Research*, 32:1048–1073.

Fang, C. and Anstee, S. (2010). Coverage path planning for harbour seabed surveys using an autonomous underwater vehicle. In *MTS/IEEE OCEANS*, Sydney.

Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, Ltd, Chichester, UK.

Galceran, E., Campos, R., Palomeras, N., Ribas, D., Carreras, M., and Ridao, P. (2014). 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*.

Galceran, E. and Carreras, M. (2012). Efficient seabed coverage path planning for ASVs and AUVs. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 88–93.

Galceran, E. and Carreras, M. (2013). Planning Coverage Paths on Bathymetric Maps for In-Detail Inspection of the Ocean Floor. In *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe.

Galceran, E., Nagappa, S., Carreras, M., Ridao, P., and Palomer, A. (2013). Uncertainty-driven survey path planning for bathymetric mapping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6006–6012.

Grasmueck, M., Eberli, G. P., Viggiano, D. A., Correa, T., Rathwell, G., and Luo, J. (2006). Autonomous underwater vehicle (AUV) mapping reveals coral mound distribution, morphology, and oceanography in deep water of the Straits of Florida. *Geophysical Research Letters*, 33(23):L23616.

Heo, Y. J. and Chung, W. K. (2013). RRT-based path planning with kinematic constraints of AUV in underwater structured environment. In *International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 523–525.

Hernández, J. D., Istenic, K., Gracias, N., García, R., Ridao, P., and Carreras, M. (2016a). Autonomous Seabed Inspection for Environmental Monitoring. In Reis, Luís Paulo and Moreira, António Paulo and Lima, Pedro U. and Montano, Luis and Muñoz-Martinez, V., editor, *ROBOT2015: Second Iberian Robotics Conference*, pages 27–39. Springer International Publishing, Lisbon, Portugal.

Hernández, J. D., Istenič, K., Gracias, N., Palomeras, N., Campos, R., Vidal, E., García, R., and Carreras, M. (2016b). Autonomous Underwater Navigation and Optical Mapping in Unknown Natural Environments. *Sensors*, 16(8):1174.

Hernández, J. D., Moll, M., Vidal, E., Carreras, M., and Kavraki, L. E. (2016c). Planning Feasible and Safe Paths Online for Autonomous Underwater Vehicles in Unknown Environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1313–1320, Daejeon, Korea.

Hernández, J. D., Vidal, E., Greer, J., Fiasco, R., Jaussaud, P., Carreras, M., and García, R. (2017). AUV online mission replanning for gap filling and target inspection. In *MTS/IEEE OCEANS*, pages 1–4, Aberdeen.

Hernández, J. D., Vidal, E., Vallicrosa, G., Galceran, E., and Carreras, M. (2015). Online path planning for autonomous underwater vehicles in unknown environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1152–1157, Seattle. IEEE.

Hollinger, G. A., Englot, B., Hover, F. S., Mitra, U., and Sukhatme, G. S. (2013). Active planning for underwater inspection and the benefit of adaptivity. *The International Journal of Robotics Research*, 32(1):3–18.

Hong-jian, W., Xin-qia, B., Xu, Z., Ming-yu, F., and Juan, L. (2004). Two approaches for autonomous underwater vehicle global path planning in large range ocean environment. In *International Conference on Intelligent Mechatronics and Automation (ICIMA)*, pages 224–227.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). OctoMap: an efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206.

Houts, S. E., Rock, S. M., and McEwen, R. (2012). Aggressive terrain following for motion-constrained AUVs. In *IEEE/OES Autonomous Underwater Vehicles (AUV)*, pages 1–7.

Hover, F. S., Eustice, R. M., Kim, A., Englot, B., Johannsson, H., Kaess, M., and Leonard, J. J. (2012). Advanced perception, navigation and planning for autonomous in-water ship hull inspection. *The International Journal of Robotics Research*, 31(12):1445–1464.

Hsu, D., Latombe, J.-C., and Motwani, R. (1999). Path planning in expansive configuration spaces. *International Journal of Computational Geometry & Applications*, 09(04n05):495–512.

Hurtós, N., Ribas, D., Cufí, X., Petillot, Y., and Salvi, J. (2015). Fourier-based Registration for Robust Forward-looking Sonar Mosaicing in Low-visibility Underwater Environments. *Journal of Field Robotics*, 32(1):123–151.

Karaman, S. and Frazzoli, E. (2011). Sampling-based Algorithms for Optimal Motion Planning. *The International Journal of Robotics Research*, 30(7):846–894.

Karaman, S., Walter, M. R., Perez, A., Frazzoli, E., and Teller, S. (2011). Anytime Motion Planning using the RRT*. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1478–1483.

Kim, H., Lee, T., and Chung, H. (2012). Any-angle path planning with limit-cycle circle set for marine surface vehicle. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2275–2280.

Kuffner, J. J. and LaValle, S. M. (2000). RRT-connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 995–1001.

Kuwata, Y., Karaman, S., Teo, J., Frazzoli, E., How, J. P., and Fiore, G. (2009). Real-Time Motion Planning With Applications to Autonomous Urban Driving. *IEEE Transactions on Control Systems Technology*, 17(5):1105–1118.

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press, Cambridge.

LaValle, S. M. and Kuffner, J. J. (2001). Randomized Kinodynamic Planning. *The International Journal of Robotics Research*, 20(5):378–400.

Luders, B. D., Karaman, S., Frazzoli, E., and How, J. P. (2010). Bounds on tracking error using closed-loop rapidly-exploring random trees. In *American Control Conference (ACC)*.

Luders, B. D., Karaman, S., and How, J. P. (2013). Robust Sampling-based Motion Planning with Asymptotic Optimality Guarantees. In *AIAA Guidance, Navigation, and Control (GNC)*.

Maki, T., Mizushima, H., Kondo, H., Ura, T., Sakamaki, T., and Yanagisawa, M. (2007). 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.

Mallios, A., Ridao, P., Ribas, D., Carreras, M., and Camilli, R. (2015). Toward autonomous exploration in confined underwater environments. *Journal of Field Robotics*, 33(7):994–1012.

Mallios, A., Vidal, E., Campos, R., and Carreras, M. (2017). Underwater caves sonar and vision data set. *International Journal of Robotic Research*, 0(0):1–5.

McMahon, J. and Plaku, E. (2016). Mission and Motion Planning for Autonomous Underwater Vehicles Operating in Spatially and Temporally Complex Environments. *IEEE Journal of Oceanic Engineering*, 41(4):893–912.

Mišković, N. (2010). *Use of Self-Oscillations in Guidance and Control of Marine Vessels*. PhD thesis, University of Zagreb.

Murthy, K. and Rock, S. (2010). Spline-based trajectory planning techniques for benthic AUV operations. In *IEEE/OES Autonomous Underwater Vehicles (AUV)*, pages 1–9.

Pairet, È., Hernández, J. D., Lahijanian, M., and Carreras, M. (2018). Uncertainty-based Online Mapping and Motion Planning for Marine Robotics Guidance. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page to appear.

Palomeras, N., El-Fakdi, A., Carreras, M., and Ridao, P. (2012). COLA2: A Control Architecture for AUVs. *IEEE Journal of Oceanic Engineering*, 37(4):695–716.

Petillot, Y., Ruiz, I. T., and Lane, D. M. (2001). Underwater vehicle obstacle avoidance and path planning using a multi-beam forward looking sonar. *IEEE Journal of Oceanic Engineering*, 26(2):240–251.

Petres, C., Pailhas, Y., Patron, P., Petillot, Y., Evans, J., and Lane, D. (2007). Path Planning for Autonomous Underwater Vehicles. *IEEE Transactions on Robotics*, 23(2):331–341.

Poppinga, J., Birk, A., Pathak, K., and Vaskevicius, N. (2011). Fast 6-DOF path planning for Autonomous Underwater Vehicles (AUV) based on 3D plane mapping. In *IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 345–350.

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

Qu, Z. and Yuan, H. (2009). Optimal real-time collision-free motion planning for autonomous underwater vehicles in a 3D underwater space. *IET Control Theory & Applications*, 3(6):712–721.

Rao, D. and Williams, S. (2009). Large-scale path planning for Underwater Gliders in ocean currents. In *Australasian Conference on Robotics and Automation (ACRA)*.

Sequeira, J. and Ribeiro, M. (1994). A two level approach for underwater path planning. In *MTS/IEEE OCEANS*, volume 2, pages II/87 – II/91, Brest.

Shkel, A. M. and Lumelsky, V. (2001). Classification of the Dubins set. *Robotics and Autonomous Systems*, 34(4):179–202.

Stack, J. and Smith, C. (2003). Combining random and data-driven coverage planning for underwater mine detection. In *MTS/IEEE OCEANS*, San Diego, CA.

Sucan, I. A. (2011). *Task and motion planning for mobile manipulators*. Phd's thesis, Rice University.

Sucan, I. A., Moll, M., and Kavraki, L. E. (2012). The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82.

Sugihara, K. and Yuh, J. (1997). GA-based motion planning for underwater robotic vehicles. In *10th International Symp. on Unmanned Untethered Submersible Technology*, pages 406–415, Durham, NH, USA.

Tan, C., Sutton, R., and Chudley, J. (2004). An incremental stochastic motion planning technique for autonomous underwater vehicles. In *IFAC Control Applications in Marine Systems Conference*, pages 483–488.

Tsianos, K. I., Sucan, I. A., and Kavraki, L. E. (2007). Sampling-based robot motion planning: Towards realistic applications. *Computer Science Review*, 1(1):2–11.

Wadoo, S. and Kachroo, P. (2010). *Autonomous Underwater Vehicles: Modeling, Control Design and Simulation*. CRC Press.

Warren, C. (1990). A technique for autonomous underwater vehicle route planning. *IEEE Journal of Oceanic Engineering*, 15(3):199–204.

Wehbe, B., Shammas, E., Zeaiter, J., and Asmar, D. (2014). Dynamic modeling and path planning of a hybrid autonomous underwater vehicle. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 729–734.

Wei Li, Farrell, J., Shuo Pang, and Arrieta, R. (2006). Moth-inspired chemical plume tracing on an autonomous underwater vehicle. *IEEE Transactions on Robotics*, 22(2):292–307.

Williams, D. P. (2010). On optimal AUV track-spacing for underwater mine detection. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4755–4762.

Yan, Z., Zhao, Y., Chen, T., and Deng, C. (2012). 3D path planning for AUV based on circle searching. In *MTS/IEEE OCEANS*, pages 1–6.

Yang, K. and Sukkarieh, S. (2008). 3D smooth path planning for a UAV in cluttered natural environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 794–800.

Ying, N., Eicher, L., Xunzhang, W., Seet, G. G., and Lau, M. W. (2000). Real-time 3D path planning for sensor-based underwater robotics vehicles in unknown environment. In *MTS/IEEE OCEANS*, volume 3, pages 2051–2058. IEEE.