To appear in IEEE Intl. Conf. on Robotics and Automation (ICRA), May 2019.

Online Multilayered Motion Planning with Dynamic Constraints for Autonomous Underwater Vehicles

Eduard Vidal[1], Mark Moll[2], Narcís Palomeras[1], Juan David Hernández[2], Marc Carreras[1], and Lydia E. Kavraki[2]

***Abstract*— Underwater robots are subject to complex hydrodynamic forces. These forces define how the vehicle moves, so it is important to consider them when planning trajectories. However, performing motion planning considering the dynamics on the robot's onboard computer is challenging due to the limited computational resources available. In this paper an efficient motion planning framework for autonomous underwater vehicles (AUVs) is presented. By introducing a loosely coupled multilayered planning design, our framework is able to generate dynamically feasible trajectories while keeping the planning time low enough for online planning. First, a fast path planner operating in a lower-dimensional projected space computes a lead path from the start to the goal configuration. Then, the lead path is used to bias the sampling of a second motion planner, which takes into account all the dynamic constraints. Furthermore, we propose a strategy for online planning that saves computational resources by generating the final trajectory only up to a finite horizon. By using the finite horizon strategy together with the multilayered approach, the sampling of the second planner focuses on regions where good quality solutions are more likely to be found, significantly reducing the planning time. To provide strong safety guarantees our framework also incorporates the conservative approximations of inevitable collision states (ICSs). Finally, we present simulations and experiments using a real underwater robot to demonstrate the capabilities of our framework.**

I. INTRODUCTION

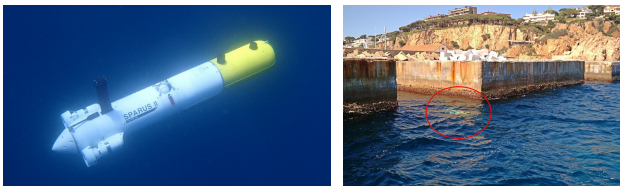
Among many other applications, autonomous underwater vehicles (AUVs) are used nowadays to perform robotic exploration and inspection [1], [2], [3]. For such tasks, where the robot navigates close to obstacles and great precision is required, it is important to carefully plan the robot trajectory taking into consideration the vehicle's dynamic constraints and safety. At the same time, the robot localization and the map of the environment, if available, are usually not accurate enough to plan all trajectories before the mission begins. In such situations, part of the mission will have to be planned online using the onboard computer. In this work we focus on this specific problem, and we are particularly interested in creating algorithms suitable for online planning.

Work at University of Girona has been supported by the GIRONA1000 and 3DAUV projects, under the grant agreements DPI2017-86372-C3-2-R and DPI2015-73978-JIN respectively, and by the Spanish Government through the FPU14/05493 PhD grant to E. Vidal.

[1]E. Vidal, N. Palomeras, and M. Carreras are members of the Underwater Robotics Research Center (CIRS), University of Girona, Spain. eduard.vidalgarcia@udg.edu, npalomer@silver.udg.edu, and marc.carreras@udg.edu.

Work at Rice University has been partially supported by NSF IIS 1317849.

[2]M. Moll, J. D. Hernández, and L. E. Kavraki are members of the Kavraki Lab, Rice University, Houston, TX, USA. mmoll@rice.edu, juandhv@rice.edu and kavraki@rice.edu.



(a) (b)

Fig. 1. Our approach has been validated using the Sparus II AUV, a torpedo-shaped underwater robot with partial hovering capabilities (a). Experimental data has been obtained in a breakwater structure, shown in (b).

Different strategies have been proposed to generate online robot trajectories. If during the mission the deviations with respect to a preplanned trajectory are expected to be small, reactive algorithms such as potential fields can be enough [4]. Alternatively, an online geometric path planner can be used. For instance, Dubins paths have been used in the past in the underwater domain [5] with great success. If the vehicle's dynamics must be considered, state lattices provide a convenient framework to plan feasible trajectories online, although they restrict the possible maneuvers of the vehicle [6]. Finally, it is also possible to use a sampling-based kinodynamic planner, but often this approach is too slow for online planning. In order to speed up the computations, some authors have proposed multilayered planning solutions [7]. We review the state of the art in motion planning with constraints and multilayered planning in section II.

In this paper we present a novel motion planning framework that takes into consideration the dynamic model of the vehicle, generating trajectories that are feasible according to the provided model, and utilizing the full dynamic range of the vehicle. A multilayered planning scheme is proposed, where a kinematically feasible path (generated using the asymptotic optimal rapidly-exploring random tree (RRT*) planner, [8]) is used to bias the search for a trajectory that is dynamically feasible and safe (generated using the stable sparse-RRT (SST) planner, [9]). As demonstrated in the results section, the SST planner on its own may be too slow for online planning with complex dynamics, so our multilayered framework aims to speed up the planning computation to make it suitable for online planning. Strong safety guarantees are also imposed by using the concept of inevitable collision states (ICSs). To the best of the authors' knowledge, this work presents for the first time a motion planning framework that combines the SST planner with the RRT* path planner with the goal of reducing the planning

time. Furthermore, it is also the first time that safety contingency maneuvers are applied to autonomous underwater vehicles. The multilayered planner scheme is shown through simulation and experiments to be a practical and efficient scheme for underwater navigation in challenging scenarios. The details of the algorithm can be found in section III, and then sections IV and V present the results using the Sparus II AUV (see Fig. 1) and conclusions.

II. RELATED WORK

This section reviews related work on motion planning for underwater robots, starting by geometrical approaches, continuing by planning on state lattices, kinodynamic planning and finally, multilayered motion planning. It also analyzes the applicability of all methods to our underwater motion planning problem according to our requirements: we want a fast kinodynamic motion planner with asymptotic optimality.

Hernández et al. [5] presented a online *geometric path planning* framework for underwater robots that approximates the robot trajectories by Dubins paths. Lin et al. [10] also used Dubins paths for aerial vehicles. As an alternative, splines have also been used in motion planning by Connors et al. [11]. However, when geometric path planners are used, it is seldom possible to follow the path with zero error. For instance, Dubins paths present sudden changes in angular velocity from section to section, which are only achievable by a system capable of infinite angular acceleration. Our goal in this work is to overcome this limitation by considering the vehicle's dynamics in the planning stage.

Another set of work relevant to underwater robotics is *planning in state lattices*. When using a state lattice the motion planning problem is solved as an unbounded graph search where the vertices and edges of the graph are generated according to a reduced set of precomputed motion primitives. The state lattice can be geometric or can include also the vehicle dynamics. Relevant examples to our underwater planning problem can be found in Pivtoraiko et al. [12], Likhachef et al. [6], and Hent et al. [13], [14]. However, using a reduced set of motions can be undesirable because some of the real capabilities of the robot are lost. Since we want to use the full dynamic range of our underwater vehicle, state lattices are not suitable for our application.

Alternatively, there are many *kinodynamic motion planners* available that do not restrict the state space. Most of them do not offer asymptotic optimality guarantees, but some of them do, at the expense of longer computational times. A kinodynamic version of the asymptotic optimal rapidly-exploring random tree (RRT*) was proposed by Webb et al. [15]. They use a fixed-final-state free-final-time controller that optimally connects any pair of states. The disadvantage of their method is that it was designed for systems with linear dynamics. If the dynamics are not linear, under some assumptions their approach can still be used by linearizing the system. However, this approach is not suitable for underwater robots due to their highly nonlinear dynamics. Hauser et al. [16] proposed a method to obtain asymptotic optimality from any feasible kinodynamic planner. They augment the state variable with the cost, and then the planning problem is repeatedly solved imposing that the solution must have a better cost at each iteration. Although this method guarantees asymptotic optimality, it is not very computationally efficient, rendering it unusable for online planning. Finally, the stable sparse-RRT (SST) planner by Li et al. [9] is a tree based motion planner that divides the motions in two categories, active and inactive, according to the cost of the path. Eventually the best motions remain in the tree while the least efficient disappear, providing also asymptotic optimality guarantees.

Since this work presents a *multilayered planning* approach, it is also important to review the literature about multilayered motion planning. Sequeira et al. [17] proposed a two layer path planning approach for underwater vehicles. In their proposal, a geometric high level planner (HLP) computes a set of consecutive waypoints from the start configuration to the goal configuration, and then the low level planner (LLP) implements a potential field technique to generate the vehicle configurations between waypoints. Alternatively, Arinaga et al. [18] also proposed a two layer path planner. Their global planner uses a connectivity graph to find the path class with minimum cost, and then, in the geometric local motion planning step, a smooth path that connects two configurations is computed. However, both approaches do not take into account the dynamics of the vehicle.

Palmieri et al. [19] proposed the use of the Theta* path planner to generate a lead path to bias the search of a rapidly-exploring random tree (RRT) planner. However, since the second planner is a RRT, no asymptotic optimality guarantees are provided. Herbert et al. [20] proposed a motion planning algorithm that plans a trajectory using simplified dynamics. In their approach, the generated trajectory is an approximation of the actual robot trajectory. Then, tracking and safety controllers are used to follow the approximate trajectory with a bounded error that depends on the worst-case disturbance. However, this means that it is required to collision check all the path for the worst-case disturbance, which is too restrictive in the underwater domain, specially when planning taking into account water currents.

Finally, Plaku et al. ([21], [22] and later on [7]) presented a multilayered approach where two path planners are tightly integrated. The first planner computes a lead, which it is defined in their work as a sequence of decomposition regions that starts and ends at regions associated with the start and goal states. In their work, the lead can be understood as a rough sketch of the cells to be visited to go from the start to the goal. Then, this lead is passed to the second planner. If the second planner does not find a trajectory within the given lead, the first planner is executed to recompute an alternative lead. In their proposal, the first planner uses discrete search, and the second planner is a sampling-based motion planner. However, the proposed combination of planners does not guarantee asymptotic optimality, and the discrete planner can become slow in problems with high dimensionality.

III. MULTILAYERED PATH PLANNING WITH DYNAMIC CONSTRAINTS

The proposed motion planning framework aims to plan trajectories which satisfy the dynamic model of an underwater vehicle while taking into account safety and performance. We consider two cases: offline planning and online planning. In offline planning the map is fully known in advance, and only one planning iteration takes place. In online planning the workspace representation evolves over time and the robot has to dynamically recompute the best trajectory to reach the goal. Multiple planning iterations take place in the robot computer, so there are often time/computing constraints.

In this work we have developed a fast motion planning framework suitable for both offline and online planning. This section presents the main parts of the proposed framework. Please, refer to the Appendix section for further details.

A. Sampling around a lead path

First we introduce the concept of sampling around a lead path. Consider a robot in a workspace W. The robot uses its sensors to update an evolving workspace representation $s(W,t)$ where t is the elapsed time. Let Q be the state space of the robot, n_Q be its order, and $q \in Q$ be a state. For each pair of a states we define a metric $\rho(q_1, q_2)$ which obeys the triangle inequality. Consider now a second state space Q' with $n_{Q'} < n_Q$ and $q' \in Q'$ where we define a projection operation $proj(q) \rightarrow q'$. For Q' we also have a metric $\rho'(q'_1, q'_2)$ which satisfies the Lipschitz condition $\rho'(proj(q_1), proj(q_2)) < M\rho(q_1, q_2)$. For us, a lead path is a path in Q' defined by $q'_l = lead\_path(t)$ with $t \in [t_1, t_2]$. We say that a state q is around the lead path if there exists a q'_l in the lead path such that $\rho'(q'_l, proj(q)) \leq d$, where d is the maximum allowed distance around the lead path. In our implementation, we sample around the lead path by first sampling a random point q' along the lead path. Then we randomly sample a point within the ball of radius d centered on q'. Finally we lift the random sample to $q \in Q$ while preserving the distance.

The idea of sampling around a lead path is really powerful as it enables a two layered scheme for motion planning for the problem considered in this paper. Furthermore, if sampling around the lead path is used in combination with uniform sampling, the optimality and probabilistic completeness guarantees of the second planner are not lost.

B. Selection of motion planners

We have selected the asymptotic optimal rapidly-exploring random tree (RRT*) motion planner [8] to generate the lead path because of its simplicity, flexibility, speed and because we expect it to scale better than grid based algorithms when the number of dimensions increase.

To generate the final trajectory we have selected the stable sparse-RRT (SST) planner [9] because of its asymptotic near-optimal (and even optimal) guarantees. According to our experiments, the SST planner by itself may not be fast enough to compute good quality trajectories online. In this work we aim to increase its performance using the proposed multilayered framework. In the results section we provide a quantitative comparison where our framework outperforms the SST planner, enabling its use in online planning.

It is important to clarify that the RRT* planner is only used to generate the geometric lead path. It is not possible to use RRT* to solve the considered motion planning problem with the dynamics of the vehicle because it is not possible to implement the required reconnection (tree rewiring) step.

C. Cost functions

Both the RRT* planner and the SST planner select the best path according to a cost function. Let $cost(q_1, q_2, s(W,t))$ be the cost of a robot motion between q_1 and q_2 according to the workspace representation at time t. For the RRT* planner, which computes the lead path, we propose a cost function (1) that reflects both the path length and the amount of occupied space around according to the integral expression:

$$cost_{RRT*}(q'_1, q'_2, s(W,t)) = \int_{q'_1}^{q'_2} occupied\_space\_around(q', s(W,t))dq', \quad (1)$$

where the function $occupied\_space\_around(q', s(W,t))$ returns the area of the occupied cells around the configuration q' according to the current representation of the workspace.

The proposed cost function for the SST motion planner (2) is similar to the one used in the RRT*, but an extra term has been added to take into account the elapsed time (with the purpose of penalizing slow movements):

$$cost_{SST}(q_1, q_2, s(W,t)) = K\ elapsed\_time(q_1, q_2) + \int_{q_1}^{q_2} occupied\_space\_around(q, s(W,t))dq, \quad (2)$$

where K is a user defined constant that reflects how much the planner has to optimize for faster trajectories and $elapsed\_time(q_1, q_2)$ returns the duration of the motion.

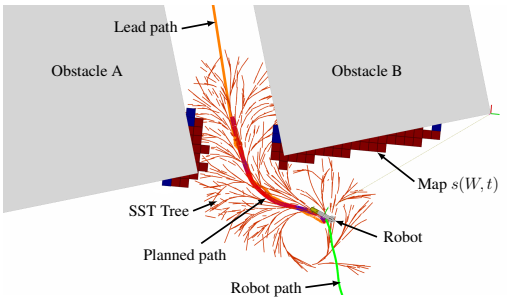


Fig. 2. The lead path helps reducing sampling in areas that are less likely to produce a good quality solution. This figure shows how the sampling takes place around the lead path. The sampling distance around the lead path should depend on the maximum possible deviation between the lead path and the final path.

D. Safety through inevitable collision states

As introduced in the previous section, the cost functions used in both the RRT* and SST motion planners take into account the occupied space around the motions. This means that motions that are further away from the obstacles will

be preferred. However, as this section highlights this can not be used as the only safety measure and other criteria are required to avoid collisions.

When planning a path from a start state to a goal state it is important to take into account whether the workspace representation $s(W, t)$ will evolve over time (online planning, more than one planning iteration) or not (offline planning, only one planning iteration), because it has performance and safety implications.

To understand the implications it is important to introduce the concept of an inevitable collision state (ICS). Intuitively, an ICS is a state where, no matter what the robot does from there, it will always end up in collision [23]. It is difficult to precisely determine if a state is an ICS, but in [24] a method was proposed to compute a superset of all the ICSs by collision checking a reduced set of contingency maneuvers, which are a set of simple maneuvers such as braking or turning. If from a state q there exists at least one contingency maneuver that is collision free, then the state q is not an ICS.

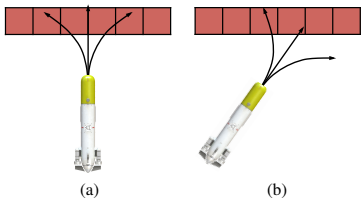


Fig. 3. In (a), although the robot is in a collision free state, all contingency maneuvers collide with the environment, so we consider this state to be an ICS. In (b), however, one of the contingency maneuvers is collision free, meaning that at least there is a maneuver that avoids the collision.

Since this work presents offline and online results, both scenarios will be discussed. In the *offline planning* scenario the initial planner plans a lead path from the projected start state to the projected goal state, and then the second planner uses the lead path to focus the sampling where a good solution is more likely to be found, providing a performance increase over using the second planner with only uniform sampling. Regarding safety, during planning it is enough to check the states for collision. Even if some of them are ICSs, they never become part of the final solution because it is impossible to find a collision free path from an ICS to the goal state.

In the *online planning* scenario it is possible to plan the whole path as in the offline case. However, in this work we propose an alternative to avoid unnecessary computations by planning only up to a finite horizon (see Fig. 2). In the presented framework, the initial planner computes a lead path from the projected robot state to the projected goal state. Then this lead path is trimmed up to a defined horizon. Finally the second planner only plans up to where the trimmed lead path ends. The horizon distance depends on the vehicle dynamics (for systems with complex dynamics, the horizon will have to be increased to ensure the goal is always reachable). Regarding safety, in the online case the proposed framework checks that the robot never ends in an ICS between planning iterations. If the planning time is set to T seconds, the SST motion planner checks all states at exactly time T and discards those which are ICSs, because there is where we expect the robot to be at the beginning of the next planning iteration.

In our implementation we have defined three contingency maneuvers, shown in Fig. 3: braking to a complete stop, braking to a complete stop while turning right and braking to a complete stop while turning left. While checking for only one of the contingency maneuver is sufficient to guarantee no collision, having three maneuvers helps us avoid discarding too many safe motions. Contingency maneuvers are only checked in the SST planner, since the RRT* planner does not account for the full dynamic state of the vehicle.

IV. RESULTS

To test the proposed motion planning framework the Sparus II AUV has been used (see Fig. 1, [25]). It is a torpedo-shaped robot with partial hovering capabilities. The surge, heave and yaw degrees of freedom (DOFs) are actuated, while sway, roll and pitch DOFs are not actuated.

Two different scenarios have been used. The first scenario consists of a series of breakwater concrete blocks (see Fig. 1) located outside the harbor of St. Feliu de Guíxols, Girona, Spain. Each block spans an area of 12x12 meters, and the distance between blocks is 5 meters. It is a challenging scenario to test motion planning algorithms due to its narrow passages and water currents. The second scenario is a 100 meters long narrow passage, similar to a canyon, between the coast cliffs of St. Feliu and a large boulder (see Fig. 6).

A. Simulated results

We have compared the performance of the SST planner with uniform sampling against the proposed framework. A set of 100 experiments, each of them with a duration of 90 seconds, has been performed. Figure 4 represents the box plots of the path cost (from Eq. (2)) as a function of the planning time. It can be seen that after the same amount of planning time has been elapsed, the cost of the solution provided by our framework is lower than for the SST planner alone, specially at the beginning, which is of particular interest for online planning. Figure 5 graphically shows the evolution of the planning tree in both cases after 10 seconds. It can be seen that the framework has already converged to a solution close to the optimal while the SST planner has not converged in all cases due to the large area that it has to explore. This experiment can be classified as offline planning because the map is known and only one planning iteration is performed to compute the trajectory from the start to the goal configurations.

The second simulated experiment we have performed consists in planning online in a complex non-structured scenario. The goal of this simulation was to test the framework for online planning before our real experiments. Figure 6 shows a top view of the environment and the robot trajectory. The robot trajectory is smooth and the robot is never close to the obstacles. The robot speed was close to the maximum allowed speed of 0.3 m/s at all time.

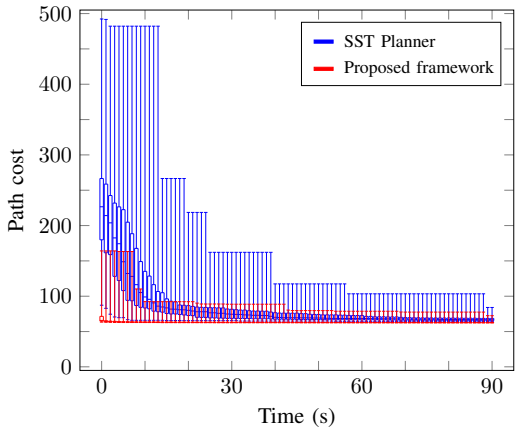


Fig. 4. This plot represents the solution cost as a function of the planning time. A box plot is represented every second. The proposed framework provides a clear convergence advantage.

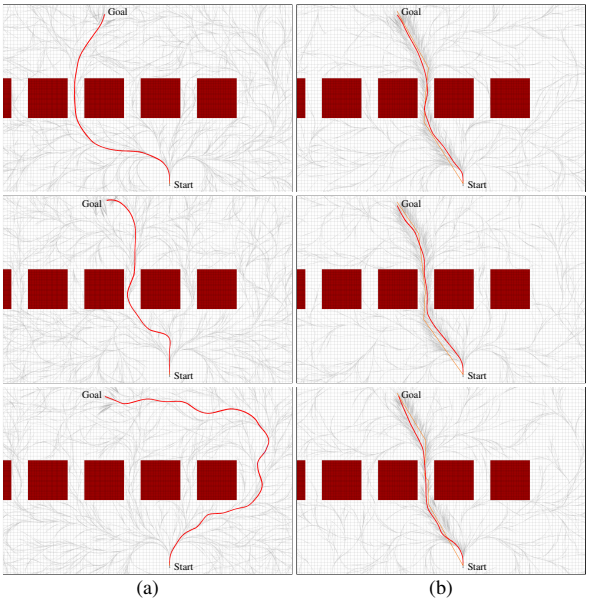


Fig. 5. Comparison between (a) the SST planner and (b) the proposed framework after 10 seconds of planning. In the proposed framework the solution converges faster to the optimal solution. This behavior is representative of what happens in other environments.

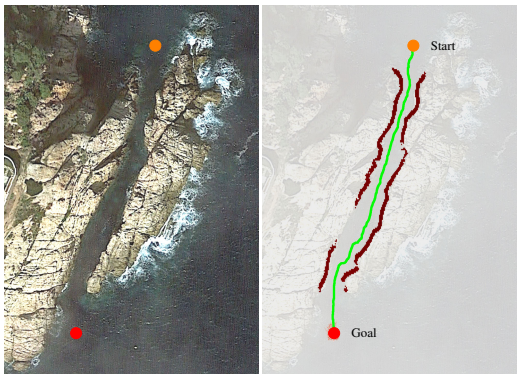


Fig. 6. Simulated experiment in a canyon scenario. The orange dot represents the initial robot position. The red dot represents the goal. The proposed framework is able to compute a safe path even in narrow passages. The total length of the trajectory is 128 meters.

Since our motion planning framework takes into account the dynamics of the system, it is possible to introduce the effects of water currents at the motion planning stage. This is achieved by considering the relative velocity of the vehicle with respect to the water in the damping term of the dynamic equations (see Eqs. (5)). The third simulated experiment we have performed shows the difference between ignoring water currents and taking them into account. Figure 7 shows the planned trajectory and the actual robot trajectory in a simple scenario, where the goal is placed between two breakwater blocks but there is a lateral water current, and it can be seen that the trajectory tracking error is much lower when currents are taken into account at the motion planning stage. For this experiment a uniform current of 0.4 m/s in the north direction is considered, although the framework can cope with any time/state parameterizable water current.

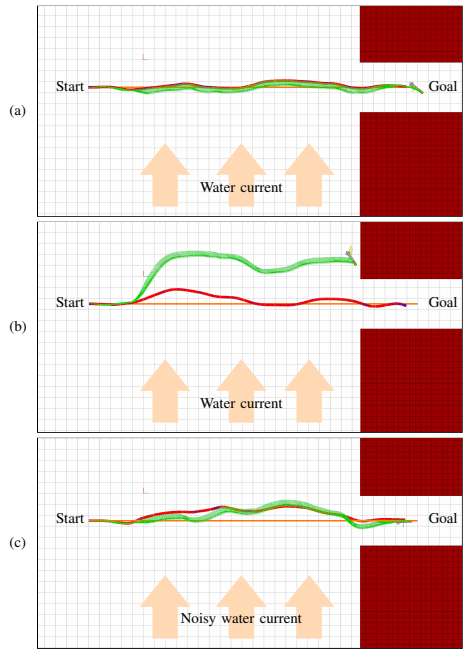


Fig. 7. In (a) the water current is fully taken into account, and in (b) it is not. There is a clear trajectory tracking error when the current is not taken into account, which depends on the parameters of the trajectory tracking controller. By taking the water current into account at the planning stage, it is possible to generate much more realistic trajectories. Since modeling the current accurately is difficult, (c) shows the robot trajectory when noise is added to the water current during the simulation. The robot trajectory deviates slightly from the the planned trajectory because of the water current mismatch, but results are much better than in (b).

B. Experimental results

Figure 8 shows real in-water experiments using the Sparus II AUV. The experiments consisted in crossing the breakwater blocks autonomously doing online planning. The robot was equipped with a mechanically scanning profiling sonar which captured exteroceptive data up to a range of 10 meters. The map was incrementally built during the experiment and replanning occurred every 5 seconds. All the experiments were performed at a depth of 2.0 meters and the requested speed was close to the maximum allowed speed of 0.3 m/s,

only decreasing for tight turns. The proposed framework was able to compute a safe path to the goal for all experiments, so no contingency maneuvers were executed to bring the robot to a safe state during the missions.

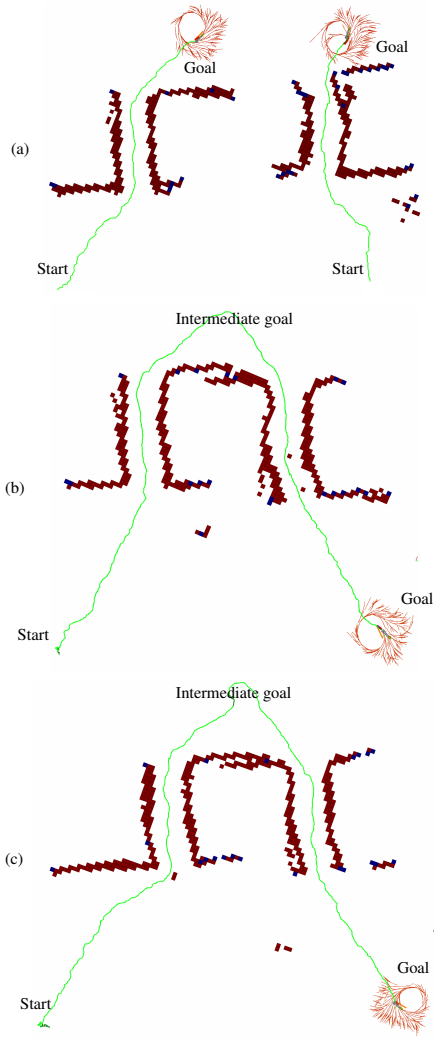


Fig. 8. In the first two experiments (a) the robot crossed the breakwater structure from different starting positions. Then, (b) and (c) show experiments where the robot crossed twice. An initial goal was set at the other side of the blocks, and then a second goal was put so that the robot returns to the same side where the mission started. Although some localization drift is present, the robot never gets close to the obstacles.

V. CONCLUSIONS

In this paper we have presented a novel motion planning framework which generates trajectories that take into account the dynamic constraints and safety of an underwater vehicle, even with a limited amount of computational resources.

To the best of the authors' knowledge, this work presents for the first time a motion planning framework that combines the stable sparse-RRT (SST) planner with the asymptotic optimal rapidly-exploring random tree (RRT*) motion planner with the goal of reducing the planning time. Furthermore, it is also the first time that safety contingency maneuvers are applied to underwater vehicles.

In addition to the previous contributions, this work also includes experimental data to evaluate the proposed planning framework. Simulations show a significant performance increase over running the SST planner on its own, achieving a faster convergence rate, which is of particular interest when performing online planning. Real experiments show that the planned trajectories are suitable for a real underwater robot, and also demonstrate that the proposed framework is able to operate within our computational budget, enabling for the first time the use of a dynamic model for planning trajectories online with the Sparus II AUV.

Finally, this framework can be easily adapted to other robots because only their dynamic model is used. It is flexible and disturbances such as water currents can be easily taken into account. Because of the aforementioned considerations, we believe this work may be relevant to the rest of the motion planning community.

APPENDIX

The state vector q used along this paper for the Sparus II AUV (using the notation of the society of naval architects and marine engineers (SNAME) [26] in Table I) is:

$$q = [x, y, \psi, u, v, r]. \tag{3}$$

TABLE I
SNAME NOTATION.

DOF	Forces and torques	Linear and angular velocities	Linear damping	Quadratic damping	Added mass
Surge	X	u	X_u	X_{uu}	$X_{\dot{u}}$
Sway	Y	v	Y_v	Y_{vv}	$Y_{\dot{v}}$
Yaw	Z	r	N_r	N_{rr}	$N_{\dot{r}}$

Taking advantage of the actuated degrees of freedom (DOFs) of the vehicle, the control variables are the surge and yaw accelerations, as defined by:

$$\mu = [\dot{u}, \dot{r}]. \tag{4}$$

At the motion planning stage, we limit the control inputs so that the planned accelerations are always achievable by the controllers of the system.

The dynamic model of the vehicle is a 2-dimensional (2D) adaptation of the 3-dimensional (3D) model found in [26]. It includes linear and quadratic damping, Coriolis and added mass terms:

$$\begin{aligned}
\dot{x} &= \cos(\psi)u - \sin(\psi)v \\
\dot{y} &= \sin(\psi)u + \cos(\psi)v \\
\dot{\psi} &= r \\
\dot{u} &= \mu[0] \\
\dot{v} &= \frac{(Y_v + Y_{vv}|v'|)v' - u(m - X_{\dot{u}})r}{m - Y_{\dot{v}}} \\
\dot{r} &= \mu[1],
\end{aligned} \tag{5}$$

where m is the mass of the vehicle and the rest of the parameters use the SNAME notation. For the damping term v' is the relative sway velocity with respect to the water.

REFERENCES

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

[2] E. Galceran, R. Campos, N. Palomeras, D. Ribas, M. Carreras, and P. Ridao, "Coverage Path Planning with Real-time Replanning and Surface Reconstruction for Inspection of Three-dimensional Underwater Structures using Autonomous Underwater Vehicles," *Journal of Field Robotics (JFR)*, vol. 32, no. 7, pp. 952–983, 2015.

[3] E. Vidal, J. D. Hernández, K. Istenic, and M. Carreras, "Online View Planning for Inspecting Unexplored Underwater Structures," *IEEE Robotics and Automation Letters (RA-L)*, vol. 99, no. 3, pp. 1436–1443, 2017.

[4] H. T. Chiang, N. Malone, K. Lesser, M. Oishi, and L. Tapia, "Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2347–2354, 2015.

[5] J. D. Hernández, M. Moll, E. Vidal, M. Carreras, and L. E. Kavraki, "Planning feasible and safe paths online for autonomous underwater vehicles in unknown environments," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 1313–1320, 2016.

[6] M. Likhachev and D. Ferguson, "Planning Long Dynamically Feasible Maneuvers for Autonomous Vehicles," *International Journal of Robotics Research (IJRR)*, vol. 28, no. 8, pp. 933–945, 2009.

[7] E. Plaku, "Region-guided and sampling-based tree search for motion planning with dynamics," *IEEE Transactions on Robotics (T-RO)*, vol. 31, no. 3, pp. 723–735, 2015.

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

[9] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *International Journal of Robotics Research (IJRR)*, vol. 35, no. 5, pp. 528–564, 2016.

[10] Y. Lin and S. Saripalli, "Path planning using 3D Dubins Curve for Unmanned Aerial Vehicles," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 296–304, 2014.

[11] J. Connors and G. Elkaim, "Analysis of a spline based, obstacle avoiding path planning algorithm," *IEEE Vehicular Technology Conference*, pp. 2565–2569, 2007.

[12] M. Pivtoraiko and A. Kelly, "Differentially constrained motion replanning using state lattices with graduated fidelity," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 2611–2616, 2008.

[13] L. Heng, L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Autonomous obstacle avoidance and maneuvering on a vision-guided mav using on-board processing," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2472–2477, 2011.

[14] L. Heng, A. Gotovos, A. Krause, and M. Pollefeys, "Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1071–1078, 2015.

[15] D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5054–5061, 2013.

[16] K. Hauser and Y. Zhou, "Asymptotically optimal planning by feasible kinodynamic planning in a state-cost space," *IEEE Transactions on Robotics (T-RO)*, vol. 32, no. 6, pp. 1431–1443, 2016.

[17] J. Sequeira and M. Ribeiro, "A two level approach for underwater path planning," *MTS/IEEE Oceans*, vol. 2, pp. 87–91, 1994.

[18] S. Arinaga, S. Nakajima, H. Okabe, A. Ono, and Y. Kanayama, "A motion planning method for an AUV," *Symposium on Autonomous Underwater Vehicle Technology*, pp. 477–484, 1996.

[19] L. Palmieri, S. Koenig, and K. O. Arras, "RRT-based nonholonomic motion planning using any-angle path biasing," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2775–2781, 2016.

[20] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Fastrack: A modular framework for fast and guaranteed safe motion planning," in *IEEE Conference on Decision and Control (CDC)*, pp. 1517–1522, 2017.

[21] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Discrete search leading continuous exploration for kinodynamic motion planning," *Robotics: Science and Systems (RSS)*, pp. 326–333, 2007.

[22] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Motion planning with dynamics by a synergistic combination of layers of planning," *IEEE Transactions on Robotics (T-RO)*, vol. 26, no. 3, pp. 469–482, 2010.

[23] T. Fraichard and H. Asama, "Inevitable collision states - a step towards safer robots?," *Advanced Robotics, Taylor & Francis*, vol. 18, no. 10, pp. 1001–1024, 2004.

[24] K. E. Bekris and L. E. Kavraki, "Greedy but safe replanning under kinodynamic constraints," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 704–710, 2007.

[25] M. Carreras, J. D. Hernández, E. Vidal, N. Palomeras, D. Ribas, and P. Ridao, "Sparus II AUV-A Hovering Vehicle for Seabed Inspection," *IEEE Journal of Oceanic Engineering (JOE)*, vol. PP, no. 99, pp. 1–12, 2018.

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