Combined Mission and Motion Planning to Enhance Autonomy of Underwater Vehicles Operating in the Littoral Zone

Erion Plaku James McMahon

***Abstract*— To enhance the mission and motion-planning capabilities of autonomous underwater vehicles (AUVs) operating in the littoral zone, this paper presents a computational approach that automatically plans collision-free and dynamically-feasible motions that enable an AUV to carry out missions expressed in Linear Temporal Logic (LTL). The key aspect of the proposed approach is its use of roadmap abstractions in configuration space to guide the expansion of a tree of feasible motions in the state space. This makes it possible to effectively deal with challenges imposed by the vehicle dynamics and the need to operate in the littoral zone, which is characterized by confined waterways, shallow water, complex ocean floor topography, varying currents, and miscellaneous obstacles. Simulations experiments with accurate AUV models carrying out missions in the littoral zone show considerable speedup over related work.**

I. Introduction

As maritime operations shift from the open ocean to the littoral zone, it becomes increasingly important to enhance the mission and motion-planning capabilities of AUVs. Operating in the littoral zone brings new challenges which are not adequately addressed by existing approaches. As the littoral zone may have small waterways that restrict the transit of multiple vessels through narrow passages or very shallow waterways that reduce the minimal altitude of the AUV, it requires the AUV to operate close to the ocean floor where the topography may have sudden changes in elevation and large objects may clutter the field. Accounting for obstacles and changes in the topography of the ocean floor requires the AUV to be able to quickly plan feasible motions.

The demand to enhance the capabilities of AUVs operating in the littoral zone becomes more pressing when coupled with critical missions, such as monitoring harbors or searching for mines and other objects of interest. AUV operations, however, are currently limited to simple missions given in terms of following a set of predefined waypoints or covering an area of interest by following a specific motion pattern. Conventional approaches to motion planning for AUVs based on potential fields, numeric optimizations, A*, or genetic algorithms [2], [8], [16], [20], [24] have generally focused on largely unobstructed and flat environments where dynamics and collision avoidance do not present significant problems. As a result, while conventional approaches are able to efficiently solve for optimal paths in 2D environments, when dealing with the challenges arising from the dynamics and the need to operate close to the ocean floor in constrained 3D environments, these approaches become inefficient. Moreover, such approaches cannot take into account missions given as LTL formulas.

To enhance the mission and motion-planning capabilities of AUVs, this paper proposes a computational approach that makes it possible to express maritime operations in LTL and automatically plan collision-free and dynamically-feasible motions to carry out such missions. As missions are characterized by events occurring across a time span, the LTL formalism allows for the combination of these events with logical connectives ($\wedge$ and, $\vee$ or, $\neg$ not) and temporal operators ($\Box$ always, $\Diamond$ eventually, $\cup$ until, $\bigcirc$ next). As an illustration, the mission of inspecting several areas of interest while avoiding collisions can be expressed in LTL as

$$\Box\pi_{\textit{safe}} \wedge \Diamond\pi_{A_1} \wedge \ldots \wedge \Diamond\pi_{A_n}$$

where $\pi_{\textit{safe}}$ denotes the proposition "no collisions" and π_{A_i} denotes "AUV inspected area A_i." As another example, searching A_1 or A_2 before A_3 or A_4 can be expressed as

$$\Box\pi_{\textit{safe}} \wedge ((\neg\pi_{A_3} \wedge \neg\pi_{A_4}) \cup ((\pi_{A_1} \vee \pi_{A_2}) \wedge \bigcirc(\pi_{A_3} \vee \pi_{A_4})))$$

The expressive power of LTL makes it possible to consider sophisticated missions. This, however, makes planning even more challenging. The discrete aspects, which relate to determining the propositions that need to be satisfied to obtain a discrete solution, are intertwined with the continuous aspects, which need to determine collision-free and dynamically-feasible motion plans to implement the discrete solutions.

As a result, approaches that compute discrete solutions without taking into account the intertwined dependencies with the continuous aspects have been limited in scope and capability [5], [7], [13], [14], [21]. Such approaches rely on the limiting assumption that a motion controller would be available to implement *any* discrete solution. However, due to dynamics, obstacles, complex ocean floor topography, and drift caused by currents, only some discrete solutions could be feasible. Indeed, determining which discrete solutions are feasible is one of the major challenges when dealing with the combined planning problem. As a result of such strong requirement on the availability of motion controllers to implement any discrete solution the applicability of these approaches is limited to 2D environments, disk robot shapes, low-dimensional state spaces, and simplified dynamics.

To address the challenges imposed by the intertwined dependencies of the discrete and continuous aspects of the combined planning problem, the proposed approach simultaneously conducts the search in the discrete space of LTL sequences and the continuous space of feasible motions. The

The authors are with the Department of Electrical Engineering and Computer Science at Catholic University of America, Washington, DC 20064. James McMahon is also with the Naval Research Laboratory, Washington, DC 20375. Erion Plaku is the corresponding author.

proposed approach builds on the framework of Plaku [17], [18], which combined discrete search with sampling-based motion planning to compute collision-free and dynamically-feasible trajectories that satisfy LTL specifications.

The proposed approach significantly enhances the previous framework [17], [18] by improving the computational efficiency and enhancing the application from ground vehicles to AUVs, taking into account drift caused by ocean currents and complex ocean floor topography. The proposed approach first uses standard techniques to convert an LTL formula into an automaton. The key aspect of the proposed approach is the introduction of roadmap abstractions to capture the connectivity of the free configuration space and, in conjunction with the automaton, to effectively guide the expansion of a tree of feasible motions in the state space. Simulation experiments with accurate AUV models carrying out missions in the littoral zone show considerable speedup over related work.

We also note that recent optimal* approaches based on μ-calculus specifications [10] are generally not applicable to the problems studied in this paper. In particular, these optimal approaches require rewiring of the tree branches, which can be done only in limited cases due to the two-point boundary value problem [12]. Due to constraints imposed by the dynamics and drift forces generated from the ocean currents, rewiring is generally not possible as it would leave gaps in the solution. Moreover, the optimal approaches come with a significant increase in computational cost, rendering them impractical for the problems studied in this paper.

II. Problem Formulation

A. AUV Simulator

This paper uses the MOOS-IvP framework [3] in order to obtain a 3D AUV simulator that robustly and accurately models the underlying vehicle dynamics. The MOOS-IvP simulator propagates the vehicle state dynamics based on a set of control inputs and external drift forces caused by the ocean currents, i.e.,

$$s_{new} \leftarrow \text{SIMULATOR}(s, u, \textit{drift}, dt),$$

where s_{new} is the new state obtained by applying the control input u and the external forces *drift* to the current state s and propagating the motion dynamics for dt time steps. The MOOS-IvP AUV simulator has been tested in many applications and shown to be robust and accurate [3].

B. Environment Information

A map of the ocean floor as well as obstacle geometries and positions are provided as input. The map should indicate the elevation $h(x, y)$ at each position (x, y). The user also provides the maximum distance d_{max} the AUV is allowed to be above the ocean floor. In other words, at position (x, y), the AUV can be between $h(x, y)$ and $h(x, y) + d_{max}$. This maximum distance is usually set to a small value so that the AUV operates close to the ocean floor. Operation close to the ocean floor is required to provide high quality acoustic bathymetric data along with detecting objects of interest.

*Optimality is achieved under certain assumptions in a probabilistic sense as computational time tends to infinity.

C. From Co-Safe LTL to Finite Automata

Since LTL planning is PSPACE-complete [19], as in prior work [17], [18], this paper considers co-safe LTL, which is satisfied by finite sequences of discrete states. Co-safe LTL formulas can be translated into Deterministic Finite Automata (DFA) [15]. Fig. 1 shows an example.

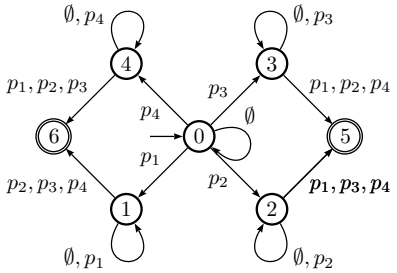


Fig. 1. The mission "visit any two of the regions p_1, p_2, p_3, p_4" encoded as a finite automaton and as a co-safe LTL formula $\bigvee_{i \neq j}(\Diamond p_i \wedge \Diamond p_j)$

Let Π denote the set of propositions. A DFA is a tuple $\mathcal{A} = (Z, Q, \delta, z_{init}, \textit{Accept})$, where Z is a finite set of states, $Q = 2^{\Pi}$ is the input alphabet corresponding to the discrete space, $\delta : Z \times Q \to Z$ is the transition function, $z_{init} \in Z$ is the initial state, and $\textit{Accept} \subseteq Z$ is the set of accepting states. The state obtained by running $\mathcal{A}$ on $\sigma = [q_i]_{i=1}^{n}$, $q_i \in Q$, starting from the state z is defined as

$$\mathcal{A}([q_i]_{i=1}^{n}, z) = \begin{cases} z, & n = 0 \\ \delta(\mathcal{A}([q_i]_{i=1}^{n-1}, z), q_n), & n > 0. \end{cases}$$

$\mathcal{A}$ accepts σ iff $\mathcal{A}(\sigma, z_{init}) \in \textit{Accept}$.

To facilitate presentation, let *Reject* denote all rejecting states of $\mathcal{A}$, i.e., states that cannot reach an accepting state. Let $\delta(z)$ denote all the non-rejecting states connected by a single transition from z, i.e.,

$$\delta(z) = \{\delta(z, q) : q \in Q\} - \textit{Reject}.$$

Let $\textit{props}(z_{from}, z_{to})$ denote all the propositions $\pi_{i_1}, \ldots, \pi_{i_k}$ labeling the transition from z_{from} to z_{to}. As an example, referring to Fig. 1, $\textit{props}(4, 6) = \{p_1, p_2, p_3\}$.

D. Discrete Semantics in the Continuous State Space

The semantics of a proposition $\pi \in \Pi$ is defined over the continuous state space S by a function $\text{HOLDS}_\pi : S \to \{\texttt{true}, \texttt{false}\}$, which indicates the continuous states that satisfy π. This interpretation provides a mapping from the continuous state space S to the discrete space Q, i.e.,

$$\textit{qstate}(s) = \{\pi : \pi \in \Pi \wedge \text{HOLDS}_\pi(s) = \texttt{true}\}.$$

For convenience, the inverse mapping from propositions $\pi \in \Pi$ to states $s \in S$ is also defined, i.e.,

$$\textit{sregion}(\pi) = \{s : s \in S \wedge \text{HOLDS}_\pi(s) = \texttt{true}\}.$$

Moreover, as the continuous state changes according to a motion trajectory $\zeta : [0, T] \to S$, parametrized by time, the

discrete state may also change. In this way, ζ maps to a sequence of discrete states, $qstates(\zeta) = [q_i]_{i=1}^{n}$, $q_i \neq q_{i+1}$. As a result of this mapping, ζ is said to satisfy a specification given by $\mathcal{A}$ iff $\mathcal{A}$ accepts $qstates(\zeta)$.

E. Problem Statement

Given an AUV simulator, environment information, proposition functions, and a co-safe LTL formula translated into a DFA $\mathcal{A}$, automatically compute a dynamically-feasible trajectory $\zeta : [0, T] \rightarrow S$ that enables the AUV to carry out the mission, i.e., $\mathcal{A}$ accepts $qstates(\zeta)$, while avoiding collisions and operating close to the ocean floor.

III. Methodology

A. Roadmap Abstraction

Let us first consider a simplified version of the problem that ignores the underlying dynamics of the vehicle. The vehicle motions in this simplified version are defined over the vehicle's configuration space, denoted as C, which accounts for translations and rotations, but not for velocities, accelerations, curvature, and other constraints related to dynamics.

1) Roadmap Construction: Motivated by the success of the Probabilistic RoadMap (PRM) [11] in dealing with motion-planning problems in configuration spaces, the proposed approach constructs a roadmap $RM = (V_{RM}, E_{RM})$ to capture the connectivity of the free configuration space C_{free}. The roadmap is constructed by sampling collision-free configurations and connecting neighboring configurations via local paths, where a distance metric $\rho : C \times C \rightarrow R^{\geq 0}$ defines the distance between two configurations. Any PRM variant [6], [9], [11], [22] can be used for the roadmap construction. Since the roadmap will be used to define heuristic costs based on shortest paths it is important to allow cycles during roadmap construction. Nearest neighbors are computed using efficient data structures [4], [23]. Since the configuration space C is of lower dimensionality than the state space S and the motions are less constrained in C, the roadmap construction takes only a fraction of the overall running time.

2) Implicit Partition of the State Space Induced by the Roadmap: The roadmap and the distance metric ρ induce an implicit partition of the state space S into equivalence classes where roadmap configurations acts as centers of Voronoi sites. In particular, let $cfg(s) \in C$ denote the configuration corresponding to the state $s \in S$. For example, $cfg(s)$ could correspond to the position and orientation components of s. The state s is then associated with the roadmap configuration $c \in V_{RM}$ closest to $cfg(s)$ according to ρ, i.e.,

$$NearestCfgRM(s) = \arg \min_{c \in V_{RM}} \rho(cfg(s), c).$$

Conversely, the site labeled with $c \in V_{RM}$ is defined as

$$sregion(c) = \{s : s \in S \wedge c = NearestCfgRM(s)\}.$$

This partition, as described later in the section, is particularly useful when determining the region from which to expand the search in the state space S.

3) Mapping Roadmap Configurations to Propositions: Each roadmap configuration c is mapped to a discrete state based on the propositions that are satisfied by c, i.e.,

$$qstate(c) = \{\pi : \pi \in \Pi \wedge \textsc{holds}_{\pi}(c) = \texttt{true}\}.$$

This mapping is computed only once when c is added to the roadmap and is then retrieved when needed. Note that HOLDS has been overloaded to take as input configurations $c \in C$ in addition to states $s \in S$.

Conversely, each proposition $\pi \in \Pi$ is associated with the roadmap configurations that satisfy it, i.e.,

$$cfgs(\pi) = \{c : c \in RM \wedge \textsc{holds}_{\pi}(c) = \texttt{true}\}.$$

From an implementation perspective, when c is added to the roadmap, it is also added to $cfgs(\pi)$ if $\textsc{holds}_{\pi}(c) = \texttt{true}$.

To ensure sufficient sampling of the propositions, additional sampling may take place during roadmap construction so that $cfgs(\pi)$ is nonempty. Based on the problem under consideration, it may be easier to sample directly from the regions associated with the propositions. For example, propositions in the experiments in this paper correspond to areas of interest in the environment. As such, a configuration can be generated by sampling a position inside the area and then a random orientation.

4) Heuristic Costs based on Shortest Paths in the Roadmap: The roadmap is used to define heuristic costs of the form $hcost(c, \pi_{i_1}, \ldots, \pi_{i_k})$ in terms of the length of the shortest path in the roadmap from c to a roadmap configuration c' satisfying one of the propositions $\pi_{i_1}, \ldots, \pi_{i_k}$, i.e., $c' \in \cup_{j=1}^{k} cfgs(\pi_{i_j})$. This heuristic cost serves as an estimate on the difficulty of expanding the motion tree $\mathcal{T}$ from tree vertices in $sregion(c)$ to reach states that satisfy one of the propositions $\pi_{i_1}, \ldots, \pi_{i_k}$. Dijkstra's shortest-path algorithm is used for the computation of the heuristic cost, where the weight of each edge $(c_i, c_j) \in E_{RM}$ is defined as $\rho(c_i, c_j)$.

B. Overall Search

The approach uses the roadmap abstraction in conjunction with the automaton $\mathcal{A}$ to guide the search in the state space S. A tree data structure $\mathcal{T}$ is used as the basis for conducting the search in S. The tree starts at the initial state s_{init} and is incrementally expanded by adding new collision-free and dynamically-feasible trajectories as branches.

The tree vertices are partitioned according to their corresponding automaton states. More specifically, let $traj(v)$ denote the trajectory obtained by concatenating the trajectories associated with the edges connecting the root of $\mathcal{T}$ to the vertex v. The vertex v keeps track of the automaton state obtained by running $qstates(traj(v))$ on $\mathcal{A}$, denoted as $astate(v)$. The computation of $astate(v)$ is done incrementally when checking for collisions the trajectory from the parent of v to v, as described in section III-B.2. This allows grouping the tree vertices according to their automaton states, i.e.,

$$TreeVertices(z) = \{v : v \in \mathcal{T} \wedge z \in Z \wedge z = astate(v)\}.$$

To speed up computation, the approach keeps track of the automaton states reached by the tree vertices, referred to as

active automaton states and defined as

$$\textit{ActiveAStates} = \{z : z \in Z \wedge |\textit{TreeVertices}(z)| > 0\}.$$

Note that a solution is obtained if the tree vertices reach an accepting automaton states, i.e., $\textit{ActiveAStates} \cap \textit{Accept} \neq \emptyset$.

The search proceeds incrementally as follows:

1) an automaton state z_{from} is selected from *ActiveAStates*;
2) an automaton state z_{to} is selected from $\delta(z_{from})$;
3) attempts are made to expand $\mathcal{T}$ from $\textit{TreeVertices}(z_{from})$ toward z_{to}.

These steps are repeated until a solution is obtained or an upper bound on computational time is exceeded. A description of each of these steps follows.

1) Selecting Automaton States: The selection of the automaton state z_{from} from *ActiveAStates* determines the tree vertices $\textit{TreeVertices}(z_{from})$ from which to expand the search. The probability of selecting z_{from} is defined as

$$\textit{prob}(z_{from}) = 2^{-d(z_{from})} / \sum_{z' \in \textit{ActiveAStates}} 2^{-d(z')},$$

where $d(z)$ denotes the minimum number of automaton transitions to reach an accepting state in $\mathcal{A}$ from z. Since the overall objective is to compute a trajectory that reaches an accepting automaton state, the selection of z_{from} is biased toward automaton states that are close to accepting states.

The selection of z_{to} aims to promote expansion toward unreached automaton states ($|\textit{TreeVertices}(z_{to})| = 0$) in order to explore new regions. Similar to the bias in the selection of z_{from}, preference is given to those unreached automaton states that are close to accepting states. Another criterion is that z_{to} should be a non-rejecting automaton state connected by a single transition from z_{from}, i.e., $z_{to} \in \delta(z_{from})$. Taking these criteria into account, z_{to} is selected according to the probability distribution

$$\textit{prob}(z_{to}) = 2^{-d(z_{to})} / \sum_{z' \in \delta(z_{from}) - \textit{ActiveAStates}} 2^{-d(z')}.$$

2) Tree Expansion: The tree expansion from $\textit{TreeVertices}(z_{from})$ adds new collision-free and dynamically trajectories aiming to reach z_{to}. Since $z_{to} \in \delta(z_{from})$ the automaton transition from z_{from} to z_{to} determines the propositions that need to be satisfied in order to reach z_{to}, which, as described in section II-C, are denoted as $\textit{props}(z_{from}, z_{to})$. Recalling again the example in Fig. 1, the automaton state z_6 can be reached from z_4 by satisfying one of the propositions p_1, p_2, p_3. Then, the objective is to expand $\textit{TreeVertices}(z_{from})$ towards states $s \in S$ such that $\textsc{holds}_\pi(s) = \texttt{true}$ for some $\pi \in \textit{props}(z_{from}, z_{to})$.

In this way, the expansion from $\textit{TreeVertices}(z_{from})$ toward z_{to} gives rise to a motion-planning problem. The proposed approach uses the roadmap abstraction to effectively guide the tree expansion. To speed up computation, $\textit{TreeVertices}(z)$ are further grouped according to the nearest configuration in the roadmap RM, i.e.,

$$\begin{aligned}\textit{TreeVertices}(z, c) \quad = \quad & \{v : v \in \textit{TreeVertices}(z) \wedge \\ & c = \textit{NearestCfgRM}(\textit{sstate}(v))\},\end{aligned}$$

where $\textit{sstate}(v)$ denotes the state in S associated with v. The approach also keeps track of all the active roadmap configurations, i.e.,

$$\textit{ActiveCfgs}(z) = \{c : c \in RM \wedge |\textit{TreeVertices}(z, c)| > 0\}.$$

From an implementation perspective, each time a vertex v is added to $\mathcal{T}$, it is also added to the corresponding $\textit{TreeVertices}(z, c)$. Moreover, c is added to $\textit{ActiveCfgs}(z)$ if not already there.

The tree expansion proceeds incrementally for several iterations. During each iteration, a roadmap configuration c is selected from $\textit{ActiveCfgs}(z_{from})$. The selection is based on $\textit{hcost}(c, \pi_1, \ldots, \pi_k)$ where $\{\pi_1, \ldots, \pi_k\} = \textit{props}(z_{from}, z_{to})$. In particular, the procedure selects the configuration c in $\textit{ActiveCfgs}(z_{from})$ with the lowest $\textit{hcost}(c, \pi_1, \ldots, \pi_k)$. As described in section III-A.4, $\textit{hcost}(c, \pi_1, \ldots, \pi_k)$ is initially computed as the length of the shortest-path in the roadmap from c to a configuration satisfying one of the propositions $\pi_1, \ldots, \pi_k$. As such, the selection strategy provides a greedy component necessary to effectively expand the search toward z_{to}. To balance the greedy exploitation with methodical exploration, $\textit{hcost}(c, \pi_1, \ldots, \pi_k)$ is increased after each selection. If the heuristic cost is not increased, then it is likely that future expansions from z_{from} will again select c. Increasing the heuristic cost, for example, by doubling it, promotes selection of other configurations in future iterations. In this way, the tree expansion has the flexibility to make rapid progress toward z_{to} while effectively discovering new ways to reach it. A hash map is used to keep track of the heuristic costs using $\langle c, \pi_1, \ldots, \pi_k \rangle$ as keys.

After selecting c from $\textit{ActiveCfgs}(z_{from})$, a vertex v is selected from $\textit{TreeVertices}(z_{from}, c)$. A simple strategy that has been shown to work well in practice is to select v uniformly at random. A control input u is then sampled also uniformly at random. A collision-free and dynamically-feasible trajectory is then obtained by applying the control input u for several time steps starting from $\textit{sstate}(v)$ and running the AUV simulator to obtain the new state. The simulation is run until a collision is found or a maximum number of steps is exceeded. Intermediate states along the generated trajectory and the final state are added to $\mathcal{T}$. The other data structures that keep track of groups of tree vertices are updated accordingly.

The approach also uses motion controllers to generate better trajectories than those obtained by applying random controls. In particular, PID controllers are used to adjust the heading and steer the AUV toward selected target positions. To expand the search toward z_{to}, the target is often selected near configurations associated with the roadmap path from c to configurations satisfying propositions enabling the transition from z_{from} to z_{to}. At other times, the target is selected uniformly at random from the entire workspace to expand the search along new directions. After selecting the target position, the PID controllers are invoked until the trajectory comes close to the target, a collision is found, or a maximum number of steps is exceeded.

IV. Experiments and Results

The approach is tested on a simulated environment using an accurate and robust AUV simulator, as described in Section II-A. Several LTL formulas are used to specify different missions.

A. Experimental Setup

1) Ocean Floor: A simulated ocean floor map is created by adding random peaks and valleys. A uniform grid is imposed over (x, y) and the center of each peak and valley is sampled at random. The height of each grid cell is determined based on the distance to the closest peak or valley. The height map is then converted to a triangular mesh. As shown in Fig. 2, this setting provides a challenging environment due to changes in topography. In the experiments, the dimensions of the grid are set to $1.5km \times 1.5km$, which correspond to realistic settings for AUV missions.

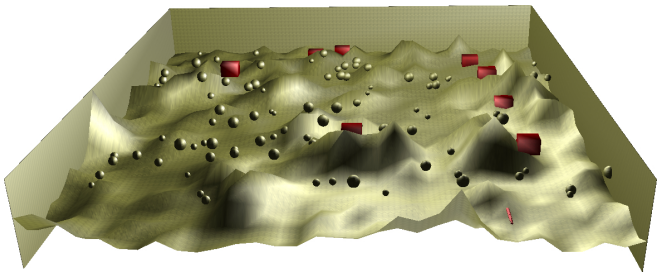


Fig. 2. One of the environments used in the experiments. The ocean floor and the obstacles are shown in gold. The areas of interest are shown as red boxes. The AUV in its initial state is shown in red.

The AUV is $4.93m$ long and has a diameter of $0.53m$. An illustration is shown in Fig. 3. The AUV moves at a

Fig. 3. AUV model.

maximum speed of $1.5m/s$. The AUV will be restricted to an altitude of $2m$ to $10m$ from the ocean floor. Operation this close to the ocean floor is required in order to provide high quality acoustic bathymetric data along with detecting objects of interest. The environment is populated with 100 obstacles placed in random positions to test the ability of the approach to avoid collisions.

2) Ocean Currents: The drift caused by ocean currents is modeled as vector fields. The drift magnitude becomes smaller as the depth increases since currents are stronger closer to the surface. For the experiments, we used publicly-available data from the National Data Buoy Center [1] describing the ocean currents for the Chesapeake bay channel.

3) LTL Specification of an Inspection Mission: Each area of interest A_i defines a proposition π_i. The function $\text{HOLDS}_{\pi_i}(s)$ is `true` iff the AUV is in area A_i when its state is s. An area A_i is defined as a box, placed at random inside inside the test environment. The inspection mission, which requires the AUV to visit each area of interest, is defined by the following LTL formula:

$$\phi_1 = \bigwedge_{i=1}^{n} \Diamond \pi_i.$$

Note that the mission specification leaves it up to the approach to determine an appropriate order in which to visit the areas. Experiments were also conducted with a mission that requires visiting the areas of interest in a predefined order $A_1, \ldots, A_n$. This mission, referred to as "sequencing," is given by

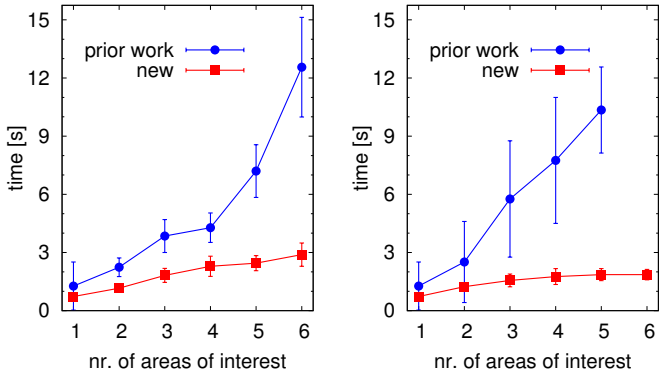
$$\phi_2 = \beta \cup (\pi_{A_1} \wedge ((\pi_{A_1} \vee \beta) \cup (\pi_{A_2} \wedge (\ldots (\pi_{A_{n-1}} \vee \beta) \cup \pi_{A_n})))),$$

where $\beta = \wedge_{i=1}^{n} \neg \pi_{A_i}$.

4) Measuring Computational Time: Experiments are run on an Intel Core i7 machine (CPU: 1.90GHz, RAM: 4GB) using Ubuntu 12.10. Code is compiled with GNU g++-4.7.2. Due to the probabilistic nature of sampling-based motion planning, the reported results are based on 30 different runs for each problem instance. The five worst and the five best runs are discarded to avoid the influence of outliers. Results report on average computational time. Standard deviations are also included as bars in the plots.

B. Results

The proposed approach is compared to our prior framework [18]. Fig. 4 summarizes the results when varying the number n of areas of interest.



(a) ϕ_1: "all" (b) ϕ_2: "sequencing"

Fig. 4. Results when varying the number of areas of interest. Bars indicate one standard deviation. Results for the proposed approach (labeled as new) include the time to construct the roadmap abstraction. The roadmap abstraction took between $0.1s - 0.4s$.

The results show considerable improvements. In fact, the approach in prior work has difficulty solving these problem instances and starts timing out as the number of areas of interest is increased (timeout set to 15s). In contrast, the proposed approach effectively solves the problem instances. Prior work relies on a workspace decomposition to guide the tree expansion, which, as the problems become more challenging, can lead the tree-based exploration along infeasible discrete paths. The lead will eventually be corrected, but it may take the approach in [18] considerable time to

do so and find other more feasible leads. In contrast, the proposed approach relies on a roadmap abstraction to capture the connectivity of the free configuration space, which more effectively guides the tree expansion in the state space.

V. Discussion

This paper focused on enhancing the mission and motion-planning capabilities of AUVs operating in the littoral zone, which is characterized by confined waterways, shallow waters, complex ocean floor topography, varying currents, and miscellaneous obstacles. The proposed approach can take into account maritime operations expressed in LTL and automatically plans collision-free and dynamically-feasible motions to carry out such missions. The key aspect of the approach is the introduction of roadmap abstractions to capture the connectivity of the free configuration space and, in conjunction with an automaton representing the LTL formula, to effectively guide the expansion of a tree of feasible motions in the state space.

In future work, we plan to enhance the proposed approach so that it can be applied to real AUVs operating in littoral environments. This would require adaptations to fit a replanning framework as well as building the map from sensory information and quickly responding to local events, such as unanticipated obstacles or changes in elevation.

Acknowledgment

The work of J. McMahon is supported in part by the Office of Naval Research, code 32.

References

[1] "NOAA HF Radar National Server and Architecture Project," National Data Buoy Center. [Online]. Available: http://www.ndbc.noaa.gov/

[2] A. Alvarez, A. Caiti, and R. Onken, "Evolutionary path planning for autonomous underwater vehicles in a variable ocean," *IEEE Journal of Oceanic Engineering*, vol. 29, no. 2, pp. 418–429, 2004.

[3] M. R. Benjamin, H. Schmidt, P. M. Newman, and J. J. Leonard, "Nested Autonomy for Unmanned Marine Vehicles with MOOS-IvP," *Journal of Field Robotics*, vol. 27, no. 6, pp. 834–875, 2010.

[4] A. Beygelzimer, S. Kakade, and J. Langford, "Cover trees for nearest neighbor," in *Interantional Conference on Machince Learning*, Pittsburgh, Pennsylvania, 2006, pp. 97–104.

[5] L. Bobadilla, F. Martinez, E. Gobst, K. Gossman, and S. LaValle, "Controlling wild mobile robots using virtual gates and discrete transitions," in *American Control Conference*, Montreal, Canada, 2012.

[6] J. Denny and N. M. Amato, "Toggle PRM: A coordinated mapping of C-free and C-obstacle in arbitrary dimension," in *International Workshop on Algorithmic Foundations of Robotics*, ser. Springer Tracts in Advanced Robotics, vol. 86, Cambridge, MA, 2013, pp. 297–312.

[7] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic mobile robots," *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.

[8] C. Ho, A. Mora, and S. Saripalli, "An evaluation of sampling path strategies for an autonomous underwater vehicle," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 5328–5333.

[9] D. Hsu, J. C. Latombe, and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," *International Journal of Robotics Research*, vol. 25, no. 7, pp. 627–643, 2006.

[10] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning with deterministic μ-calculus specifications," in *American Control Conference*, Montreal, CA, 2012, pp. 735–742.

[11] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[12] H. Keller, *Numerical Methods for Two-Point Boundary-Value Problems*. New York, NY: Dover, 1992.

[13] M. Kloetzer and C. Belta, "Automatic deployment of distributed teams of robots from temporal logic specifications," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 48–61, 2010.

[14] H. Kress-Gazit, T. Wongpiromsarn, , and U. Topcu, "Correct, reactive robot control from abstraction and temporal logic specifications," *IEEE Robotics & Automation Magazine*, vol. 18, no. 3, pp. 65–74, 2011.

[15] O. Kupferman and M. Vardi, "Model checking of safety properties," *Formal methods in System Design*, vol. 19, no. 3, pp. 291–314, 2001.

[16] C. Petres, Y. Pailhas, P. Patron, Y. Petillot, J. Evans, and D. Lane, "Path planning for autonomous underwater vehicles," *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 331–341, 2007.

[17] E. Plaku, "Path planning with probabilistic roadmaps and linear temporal logic," in *International Conference on Intelligent Robots and Systems*, Vilamoura, Algarve, Portugal, 2012, in press.

[18] ——, "Planning in discrete and continuous spaces: From LTL tasks to robot motions," in *Advances in Autonomous Robotics*, ser. Lecture Notes in Computer Science, Bristol, UK, 2012, vol. 7429, pp. 331–342.

[19] A. Sistla, "Safety, liveness and fairness in temporal logic," *Formal Aspects of Computing*, vol. 6, pp. 495–511, 1994.

[20] M. Soulignac, "Feasible and optimal path planning in strong current fields," *IEEE Transactions on Robotics*, vol. 27, no. 1, pp. 89–98, 2011.

[21] M. Wu, G. Yan, Z. Lin, and Y. Lan, "Synthesis of output feedback control for motion planning based on LTL specifications," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO, 2009, pp. 5071–5075.

[22] H.-Y. Yeh, S. Thomas, D. Eppstein, and N. M. Amato, "UOBPRM: A uniformly distributed obstacle-based PRM," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura, Algarve, Portugal, 2012, pp. 2655–2662.

[23] A. Yershova and S. M. LaValle, "Improving motion planning algorithms by efficient nearest-neighbor searching," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 151–157, 2007.

[24] N. K. Yilmaz, C. Evangelinos, P. Lermusiaux, and N. M. Patrikalakis, "Path planning of autonomous underwater vehicles for adaptive sampling using mixed integer linear programming," *IEEE Journal of Oceanic Engineering*, vol. 33, no. 4, pp. 522–537, 2008.