Autonomous Underwater Vehicle Mine Countermeasures Mission Planning via the Physical Traveling Salesman Problem

James McMahon and Erion Plaku

***Abstract*— Mine Countermeasures (MCM) present a challenging motion-planning problem when taking into account ocean currents, complex bathymetry, and vehicle dynamics. In an effort to improve autonomous underwater vehicle (AUV) motion planning for MCM, this work presents a Physical Traveling Salesmen Problem (PTSP) approach as a way to quickly and robustly generate motions for the reacquisition and identification portion of a MCM mission while considering the temporally and spatially complex marine environment as well as the nonlinear dynamics of the AUV. By leveraging TSP solvers as well as sampling-based motion planning, this work shows that it is possible to efficiently compute low-cost, dynamically-feasible, and collision-free trajectories.**

I. Introduction

The Office of Naval Research Naval S&T strategic plan holds Mine Countermeasures (MCM) as a significant challenge for the U.S. Navy when trying to achieve and maintain undersea dominance [11]. The ability to robustly and quickly detect and classify underwater mines is key in order to maintain the safety of both naval and commercial vessels. In this context, autonomous underwater vehicles (AUVs) are vital to this need in order to operate in hazardous areas.

MCM missions involving AUVs typically occur over several phases. First, an initial survey is conducted in which an AUV uses acoustic sensors to detect mine-like objects within a specified region. In a fully autonomous system, detection and classification algorithms are run on board in order to generate a list of potential targets. Given the list of potential targets, the AUV then reacquires data either through additional acoustic sensing or via optical based sensors. The reacquisition phase is for identification purposes and is critical in confirming with high certainty whether or not a target is a mine. Once the targets are reacquired and a final identification can be made, any mines are then neutralized.

This paper investigates the stage associated with reacquisition and identification (RID), where the AUV must re-inspect targets in order to collect data for identification purposes. Given the target list, the AUV must plan a trajectory that re-inspects each target. The problem is similar to the traveling salesman problem (TSP) in that there are a series of targets that need to be inspected while minimizing the distance traveled by the AUV. Simply using a TSP solver to generate an inspection ordering, however, would not be sufficient since the TSP solver does not take into account the dynamics of the AUV as well as the drift caused by the time-varying ocean currents. Building on prior work [4], [14], this paper addresses such challenges by combining sampling-based motion planning with a TSP solver. Sampling-based motion planning leverages the idea of searching for a solution by expanding a motion tree consisting of collision-free and dynamically-feasible motions. A TSP solver guides the motion-tree expansion by providing low-cost tours which indicate the order in which to inspect the targets. As the motion tree is expanded along the selected tour, information is also gathered regarding the progress being made. This information is used to compute new alternative tours when making progress along the current tour becomes difficult due to the constraints imposed by the vehicle dynamics or the drift caused by the ocean currents. This interplay between sampling-based motion planning and TSP solvers gives the approach the flexibility to discover new alternative tours and effectively compute a low-cost, collision-free, and dynamically-feasible motion trajectory that reaches each target. Experiments show considerable improvements over related work both in terms of runtime and solution costs.

The authors are with the Dept. of Electrical Engineering and Computer Science, Catholic University of America, Washington DC 20064. J. McMahon is also with the U.S. Naval Research Laboratory, Washington, DC 20375.

II. Related Work

Related work has generally approached the RID planning problem as a TSP variant without taking the vehicle dynamics into account or the drift caused by the ocean currents. The work in [5] frames the problem of autonomous data collection as the prize-collecting TSP variant with neighbors (PC-TSP) and proposes an approach based on self-organizing maps. The work in [15] also frames the RID planning problem as a TSP variant and proposes a genetic algorithm to solve it suitable for an underactuated robot in two dimensions. The work in [6] also uses a genetic algorithm but instead of framing the RID mission as a TSP it decides how to transition between MCM survey regions.

Other work uses the planning domain definition language (PDDL) [7] in combination with a probabilistic roadmap (PRM) [9] to model the inspection task of an underwater structure using an AUV [2]. This approach is further extended to include semantic based knowledge representations for the MCM problem in order to plan both surveys and reacquisition missions [12].

While these approaches investigate the problem of optimal (or near optimal) ordering of inspection tasks for MCM, RID, or the inspection of underwater structures, they do not fully take into account the dynamics of the AUV or the interactions with the environment such as the effects of strong ocean currents. Our work develops an approach for RID planning

978-0-933957-43-5 ©2015 MTS

that takes into account the dynamics of the vehicle as well as the effects of time-varying ocean currents.

III. Problem Formulation

Let $\mathcal{E}$ denote the 3D environment where the AUV operates. The environment is often defined by a bounding box and the bathymetry map which provides the depth information. Let $\mathcal{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_n\}$, $\mathcal{R}_i \subseteq \mathcal{E}$, denote the target regions that the AUV should visit. The user can also specify a set of restricted or forbidden regions, denoted by $\mathcal{O} = \{\mathcal{O}_1, \ldots, \mathcal{O}_m\}$, $\mathcal{O}_j \subseteq \mathcal{E}$, which the AUV should always avoid. Clearly, the restricted and target regions should be disjoint, i.e., $(\mathcal{R}_1 \cup \ldots \cup \mathcal{R}_n) \cap (\mathcal{O}_1 \cup \ldots \cup \mathcal{O}_m) = \emptyset$. The objective is then to compute a low-cost and dynamically-feasible trajectory that visits each target region in $\mathcal{R}$ while avoiding the restricted regions in $\mathcal{O}$.

This paper relies on the MOOS-IvP framework [1] in order to obtain an accurate 3D simulation of the vehicle dynamics even when considering the drift caused by the time-varying ocean currents. The simulated vehicle corresponds to an OceanServer Iver2 whose state $s \in \mathcal{S}$ is maintained in terms of position, depth, orientation, velocity, and buoyancy rate, where $\mathcal{S}$ denotes the state space. The AUV is controlled by setting the desired heading, depth, and speed, where $\mathcal{U}$ denotes the control space. Internally, MOOS-IvP uses PID controllers to choose actuator and thrust values that steer the vehicle along the desired heading, depth, and speed values prescribed by the control input $u \in \mathcal{U}$. The drift caused by the time-varying ocean currents is provided by a function of the form $\textsc{drift}(x, y, d, t)$ which indicates the drift forces at position (x, y), depth d, and time t. This makes the approach general as it can be used with any drift model.

The MOOS-IvP simulator obtains the new state of the vehicle by simulating the vehicle dynamics for one time step based on the current state, control, and drift forces. This is captured by a function of the form

$$s_{\text{new}} \leftarrow \textsc{simulate}(s, u, f, dt), \tag{1}$$

where $s_{\text{new}} \in \mathcal{S}$ is the new state, $s \in \mathcal{S}$ is the current state, $u \in \mathcal{U}$ is the input control, f represents the drift forces, and dt is the time step. The new state is considered valid if the vehicle does not collide with any of the forbidden regions or the ocean floor, which is computed by a function of the form $\textsc{collision} : \mathcal{S} \rightarrow \{\texttt{true}, \texttt{false}\}$. The PQP package [10] can be used to efficiently implement collision checking.

Starting at a time t, a state $s \in S$, and applying a sequence of input controls $\langle u_0, u_1, \ldots, u_{\ell-1} \rangle$ gives rise to a motion trajectory $\zeta : \{0, 1, \ldots, \ell\} \rightarrow S$ where

$$\zeta(0) \leftarrow s \text{ and } \zeta(i+1) \leftarrow \textsc{simulate}(\zeta(i), u_i, f_i, dt), \tag{2}$$

where $f_i \leftarrow \textsc{drift}(\textsc{pos}(\zeta(i)), \textsc{depth}(\zeta(i)), t + i * dt)$ represents the drift forces caused by the time-varying ocean currents. Let $\textsc{target} : \mathcal{S} \rightarrow \mathcal{R} \cup \{\emptyset\}$ determine which target if any the vehicle has reached at state $s \in \mathcal{S}$. More specifically,

$$\textsc{target}(s) = \begin{cases} \{\mathcal{R}_i\}, & \text{if } \textsc{posd}(s) \in \mathcal{R}_i \\ \emptyset, & \text{if } \textsc{posd}(s) \notin \bigcup_{\mathcal{R}_i \in \mathcal{R}} \mathcal{R}_i, \end{cases} \tag{3}$$

where $\textsc{posd}(s)$ denotes the position and depth defined by the state s. A motion trajectory $\zeta : \{0, 1, \ldots, \ell\} \rightarrow \mathcal{S}$ reaches zero or more targets, i.e.,

$$\textsc{targets}(\zeta) = \textsc{target}(\zeta(0)) \cup \ldots \cup \textsc{target}(\zeta(\ell)). \tag{4}$$

The RID planning problem can then be defined as follows.

Definition 1: Given

- a 3D operational environment $\mathcal{E}$,
- the forbidden regions $\mathcal{O} = \{\mathcal{O}_1, \ldots, \mathcal{O}_m\}$,
- a list of detected target regions $\mathcal{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_n\}$,
- a model of the ocean currents as a function of the form $\textsc{drift}(x, y, d, t)$,
- the state space $\mathcal{S}$, the control space $\mathcal{U}$, a function $\textsc{simulate}(s, u, f, dt)$ to simulate the vehicle dynamics,
- and an initial state $s_{\text{init}} \in \mathcal{S}$

compute a sequence of control inputs $\langle u_0, u_1, \ldots, u_{\ell-1} \rangle$ such that the dynamically-feasible trajectory $\zeta : \{0, 1, \ldots, \ell\} \rightarrow \mathcal{S}$ obtained by applying the controls in sequence starting at the state s is collision-free and reaches all the targets, i.e.,

$$\textsc{targets}(\zeta) = \mathcal{R} \text{ and} \tag{5}$$

$$\forall i \in \{0, 1, \ldots, \ell\} : \textsc{collision}(\zeta(i)) = \texttt{false}. \tag{6}$$

The planner will also seeks to reduce the distance traveled by the AUV.

IV. Method

In order to effectively compute a low-cost, collision-free, and dynamically-feasible motion trajectory that inspects all the target regions, the proposed approach couples sampling-based motion planning with a TSP solver. The TSP solver operates over a simplified, discrete, representation of the problem. Solutions obtained by the TSP solver guide sampling-based motion planning as it expands a motion tree in the state space. More specifically, the discrete abstraction is obtained by constructing a navigation roadmap which captures the connectivity of the environment. Drawing from the success of the Probabilistic RoadMap (PRM) [9], the navigation roadmap is constructed by sampling collision-free waypoints and connecting neighboring waypoints with collision-free paths. Using the navigation roadmap and the list of targets reached along each branch, the motion tree is partitioned into equivalent groups. The TSP solver searches over the navigation roadmap in order to compute a low-cost tour for each group. Using the tour as a heuristic, preference is given to groups associated with low-cost tours as such expansions promote the generation of low-cost trajectories. Sampling-based motion planning seeks to add new branches to the motion tree from the vertices associated with the selected group along its tour. Unsuccessful expansions are penalized in order to promote expansions along alternative tours associated with other groups. The process of selecting an equivalent group and expanding the motion tree along its tour is repeated until a solution is found or an upper bound on runtime is reached. The rest of the section describes the main steps of the proposed approach in more detail.

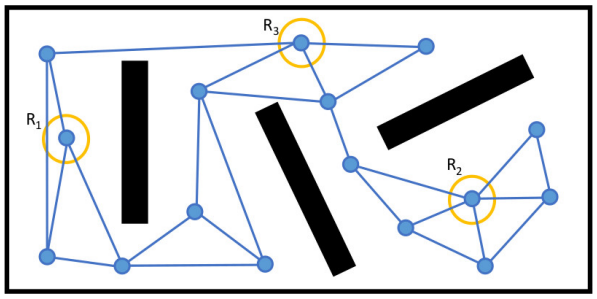

Fig. 1. Illustration of a roadmap. Targets are labeled with co-located roadmap vertices

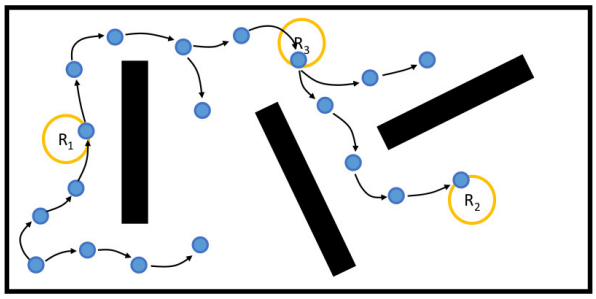

Fig. 2. Illustration of the motion tree. Edges between vertices represent collision-free and dynamically-feasible trajectories

A. Discrete Abstraction

The discrete abstraction is obtained via a roadmap which is represented as a weighted graph $RM = (V_{RM}, E_{RM})$. The roadmap is constructed by sampling-collision free waypoints and connecting neighboring waypoints with collision-free paths. An illustration is provided in Fig. 1. The roadmap does not take the vehicle dynamics into account but serves instead to guide the motion-tree expansion. Initially, a waypoint $p_{\mathcal{R}_i}$ from each target region $\mathcal{R}_i \in \mathcal{R}$ is added to the set of roadmap vertices V_{RM}. The waypoint $p_{\mathcal{R}_i}$ can be selected as the centroid of $\mathcal{R}_i$ or some other point inside $\mathcal{R}_i$. The roadmap is further populated by using random sampling to generate a number of collision-free waypoints. More specifically, a position (x, y) is generated uniformly at random inside the bounding box associated with the environment $\mathcal{E}$. A depth d is generated uniformly at random in $[0, d_{\max})$ where $d_{\max}$ denotes the maximum depth at position (x, y) as provided by the bathymetry map associated with the environment $\mathcal{E}$. The point $p = (x, y, d)$ is added as a roadmap vertex when it is outside the forbidden regions $\mathcal{O}$. This process is repeated until the roadmap reaches the user-defined number of vertices.

After populating the roadmap, attempts are made to connect each $p \in V_{RM}$ to its k-nearest neighbors. If the straight-line segment (p, p_{neigh}) is not in collision with the forbidden regions, then (p, p_{neigh}) is added as a roadmap edge to E_{RM}. The cost of the edge, denoted by $\text{COST}(p, p_{\text{neigh}})$, is defined as the Euclidean distance from p to p_{neigh}.

B. Motion Tree

The motion tree $\mathcal{T}$ is rooted at the initial state s_{init} and is incrementally expanded by adding collision-free and dynamically-feasible trajectories as branches. An illustration is shown in Fig. 2. Each vertex $v \in \mathcal{T}$ is associated with a collision-free state, denoted by $v.s$. Each edge $(v, v') \in \mathcal{T}$ is labeled with a tuple $\langle u, f, t \rangle$ to denote the motion from $v.s$ to $v'.s$, which is obtained by simulating the vehicle dynamics when applying the control input u to the state $v.s$ for one time step where f represents the drift forces at time t, i.e., $v'.s = \text{SIMULATE}(v.s, u, f, dt)$.

Let $\zeta_{\mathcal{T}}(v)$ denote the trajectory from the root of $\mathcal{T}$ to the vertex $v \in \mathcal{T}$. Such trajectory is obtained by concatenating the motions associated with the edges that connect the root vertex to v. The trajectory $\zeta_{\mathcal{T}}(v)$ constitutes a solution if it reaches all the targets, i.e., $\text{TARGETS}(\zeta_{\mathcal{T}}(v)) = \mathcal{R}$. In order to speed up the computation, the targets reached by $\zeta_{\mathcal{T}}(v)$ are stored as a field in the data structure representing v, i.e., $v.\text{targets} = \text{TARGETS}(\zeta_{\mathcal{T}}(v))$.

In order to take advantage of the navigation roadmap $RM = (V_{RM}, E_{RM})$, each vertex $v \in \mathcal{T}$ is mapped to the nearest waypoint in the roadmap, i.e.,

$$v.\text{waypt} = \underset{p \in V_{RM}}{\text{argmin}} \, ||p - \text{POSD}(v.s)||_2. \tag{7}$$

Using the targets and the navigation roadmap, the motion tree $\mathcal{T}$ is partitioned into equivalent groups. Two vertices v and v' belong to the same group if they map to the same roadmap waypoint and the trajectories $\zeta_{\mathcal{T}}(v)$ and $\zeta_{\mathcal{T}}(v')$ have reached the same targets. In this way, a set of targets $\mathcal{Q} \subseteq \mathcal{R}$ and a waypoint $p \in V_{RM}$ define an equivalent group as

$$\text{group}_{\langle \mathcal{Q}, p \rangle} = \{v : \quad v \in \mathcal{T} \text{ and } v.\text{targets} = \mathcal{Q} \text{ and} \tag{8}$$

$$v.\text{waypt} = p\}, \tag{9}$$

which groups together all the tree vertices that map to the waypoint p and whose trajectories in the motion tree have reached all and only the targets in $\mathcal{Q}$. This induces a partition of the motion tree into equivalent groups, i.e.,

$$\text{groups} = \{\text{group}_{\langle \mathcal{Q}, p \rangle} : \quad \mathcal{Q} \subseteq \mathcal{R} \text{ and } p \in V_{RM} \text{ and} \tag{10}$$

$$|\text{groups}_{\langle \mathcal{Q}, p \rangle}| > 0\}. \tag{11}$$

These groups, as described next, are used to effectively expand the motion tree so that it reaches all the targets in $\mathcal{R}$.

C. Expanding the Motion Tree

Initially, the motion tree $\mathcal{T}$ contains only the root vertex v_{init} which is associated with the initial state s_{init}. There is also only one group, namely $\text{group}_{\langle v_{\text{init}}.\text{targets}, v_{\text{init}}.\text{waypt} \rangle}$. Additional groups are created when new vertices are added to $\mathcal{T}$. The motion-tree expansion is driven by two procedures: (i) selecting $\text{group}_{\langle \mathcal{Q}, p \rangle}$ from groups, and (ii) expanding $\mathcal{T}$ from the vertices associated with $\text{group}_{\langle \mathcal{Q}, p \rangle}$.

To effectively expand $\mathcal{T}$, the TSP solver is invoked when $\text{group}_{\langle \mathcal{Q}, p \rangle}$ is first created in order to compute a low-cost tour, denoted by $\text{tour}_{\langle \mathcal{Q}, p \rangle}$. Specifically, the TSP solver searches the navigation roadmap $RM = (V_{RM}, E_{RM})$ to compute a low-cost tour that starts at p and reaches all the remaining targets, namely $\mathcal{R} \setminus \mathcal{Q}$. During the selection

process, preference is given to those groups associated with a low-cost tour since expansions from such groups could quickly reach all the remaining targets. Preference is also given to groups that have not been selected frequently in the past in order to avoid overexploration. In fact, a weight $w_{\langle \mathcal{Q},p \rangle}$ is defined as

$$w_{\langle \mathcal{Q},p \rangle} = \frac{\alpha^{\mathrm{sel}_{\langle \mathcal{Q},p \rangle}}}{\mathrm{cost}\big(\mathrm{tour}_{\langle \mathcal{Q},p \rangle}\big)}, \tag{12}$$

where $0 < \alpha < 1$ and $\mathrm{sel}_{\langle \mathcal{Q},p \rangle}$ denotes the number of times $\mathrm{group}_{\langle \mathcal{Q},p \rangle}$ has been previously selected. The group with the maximum weight is then selected for expansion, i.e.,

$$\underset{\mathrm{group}_{\langle \mathcal{Q},p \rangle} \in \mathrm{groups}}{\mathrm{argmax}} \quad w_{\langle \mathcal{Q},p \rangle}. \tag{13}$$

After selecting $\mathrm{group}_{\langle \mathcal{Q},p \rangle}$, the objective is to expand $\mathcal{T}$ along $\mathrm{tour}_{\langle \mathcal{Q},p \rangle}$ so that $\mathcal{T}$ can quickly reach the remaining targets. To promote such expansions, a point q is sampled uniformly at random along $\mathrm{tour}_{\langle \mathcal{Q},p \rangle}$. The vertex v closest to q from the vertices in $\mathrm{group}_{\langle \mathcal{Q},p \rangle}$ is selected and an attempt is made to generate a collision-free and dynamically-feasible motion trajectory from v to q. More specifically, PID controllers available in the MOOS-IvP simulator are used to steer the AUV from v to q by setting the desired heading and depth to point to q. After each simulation step, the new state is checked for collisions. If it collides with any of the forbidden regions, the tree expansion from v stops, and the method goes back to selecting an equivalent group. If the new state is not in collision, then it is added as a new vertex, v_{new}, to $\mathcal{T}$. When v_{new} is added to $\mathcal{T}$, it is also added to the corresponding group, namely $\mathrm{group}_{\langle v_{\mathrm{new}}.\mathrm{targets}, v_{\mathrm{new}}.\mathrm{waypt} \rangle}$. If the group does not exist, it is created and added to groups. When the group is first created, the TSP solver is also invoked to compute the low-cost tour over the navigation roadmap. The expansion from v terminates after a maximum number of steps or getting near q.

The planner proceeds in this way by selecting and expanding a group until a solution is found or the runtime limit is reached. By using the low-cost tours computed by the TSP solver to guide the motion-tree expansion, the planner is able to effectively compute a low-cost, collision-free, and dynamically-feasible motion trajectory that reaches all the target regions.

V. Experiments and Results

The planner is tested in simulation over a riverine environment corresponding to the Patuxent river emptying into the Chesapeake Bay, as showing in Fig. 3. The operational area of the vehicle is $12km \times 9km$ with a maximum depth of $45m$. The scene uses historical data from the Chesapeake Bay, the bathymetry is obtained from the National Geophysical Data Center [3], and time-varying currents in three dimensions are obtained by the Chesapeake Bay Operational Forecast System [8]. The scene represents a challenging and realistic environment characterized by complex bathymetry and dynamic time-varying ocean currents. In the experiments, the AUV must reacquire target information on previously detected targets. Fig. 4 shows a solution obtained by the planner.

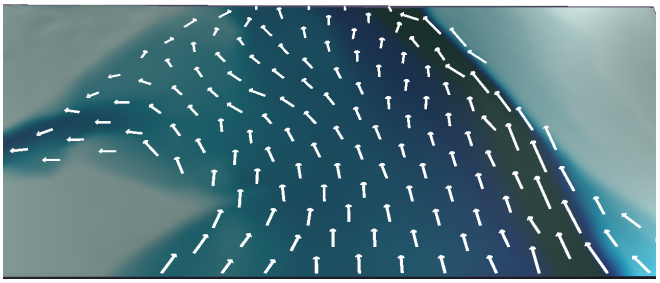

Fig. 3. Chesapeake Bay Scene with ocean currents shown in white, bathymetry shown in blue, and land shown in gray

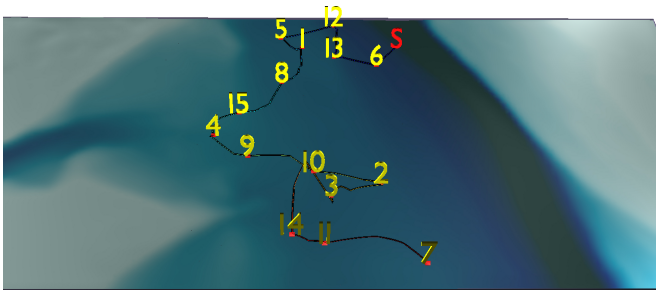

Fig. 4. Example of a collision-free and dynamically-feasible solution trajectory with 15 targets, starting at 'S'.

To test the performance of the planner, experiments were carried out by varying the number and the placement of the targets. For a given number of targets n, sixty problem instances were generated, denoted by $\mathcal{I}_n$. A problem instance in $\mathcal{I}_n$ is obtained by placing n targets at random within the environment $\mathcal{E}$ while ensuring a minimum separation distance among them. Results report on the mean runtime and solution cost which is measured as the distance traveled by the AUV. Runtime measures everything from reading the input file to reporting that a solution is found. Experiments were run on an Intel Core i7 (CPU: 1.90GHz, RAM: 4GB) using Ubuntu 14.04 and GNU g++-4.8.2.

Fig. 5 shows the results in comparison to previous work [13] which used Linear Temporal Logic (LTL) to specify multiple targets. LTL provides an expressive logical model which combines propositions with logical (AND, OR, NOT) and temporal (NEXT, ALWAYS, EVENTUALLY) connectives. In LTL, the problem of reaching multiple targets can be expressed as "eventually each region will be reached." The LTL planner takes the LTL specification and converts it into an automaton which encodes all the possible permutations in which the targets can be reached.

Results demonstrate significant improvements both in terms of the runtime and solution cost. In fact, the runtime of the LTL planner increases rapidly with the number of targets, timing out (set at $120s$ per run) at problem instances with 6 or more targets. This is due to the exponential dependency associated with the search conducted by the LTL planner which is over the product of the navigation roadmap with

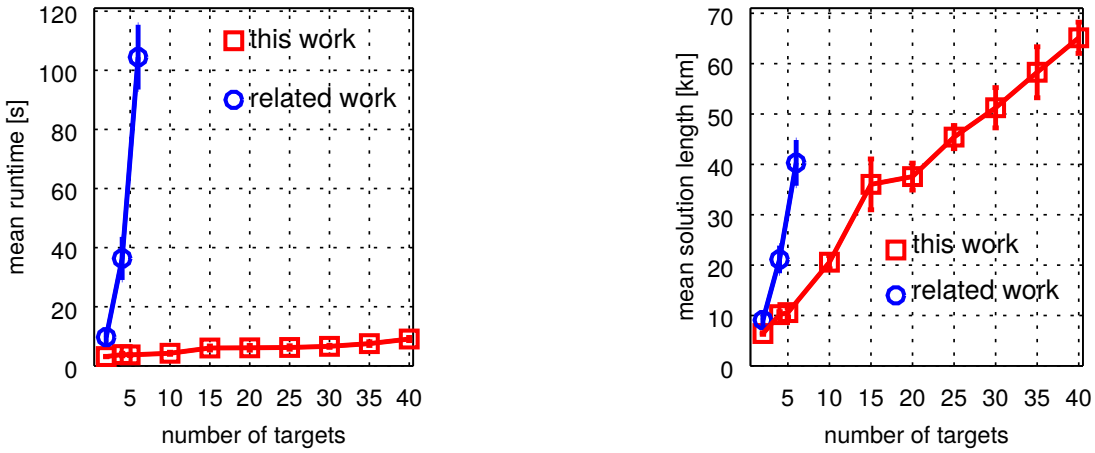

Fig. 5. Mean runtime and solution trajectory length for the proposed approach vs. related work [13]. The graph for related work shows no results after 6 targets due to the planner timing out.

the automaton representing the LTL formula. In contrast, the proposed approach relies on a TSP solver to efficiently search the navigation roadmap and compute low-cost tours that effectively guide the motion-tree expansion.

VI. Discussion

This paper focused on Mine Countermeasure (MCM) mission planning for autonomous underwater vehicles. By combining sampling-based motion planning with a TSP solver we show that when compared to related work there is a significant speed up in planning time as well as reduction in overall trajectory cost. This is an active project and future research involves developing techniques to bias the search to expand the motion tree towards targets with high uncertainty in the probability of classification as well as considering finite resources with respect to time and fuel.

References

[1] 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.

[2] M. Cashmore, M. Fox, T. Larkworthy, D. Long, and D. Magazzeni, "Auv mission control via temporal planning," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 6535–6541.

[3] N. G. D. Center, "U.S. coastal relief model - southeast atlantic [nov, 2014]," 1999.

[4] S. Edelkamp and E. Plaku, "Multi-goal motion planning with physics-based game engines," in *IEEE Conference on Computational Intelligence and Games*, 2014, pp. 115–122.

[5] J. Faigl and G. A. Hollinger, "Unifying multi-goal path planning for autonomous data collection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 2937–2942.

[6] C. Fang and S. Anstee, "Coverage path planning for harbour seabed surveys using an autonomous underwater vehicle," in *Proc. MTS/IEEE OCEANS*, May 2010, pp. 1–8.

[7] M. Ghallab, A. Howe, C. Knoblock, C. McDermott, A. Ram, D. Veloso, M. Weld, and D. Wilkins, "PDDL—the planning domain definition language," 1998.

[8] T. F. Gross, K. T. Bosley, and K. W. Hess, *The Chesapeake Bay operational forecast system (CBOFS): technical documentation*. US Department of Commerce, National Oceanic and Atmospheric Administration, National Ocean Service, Office of Coast Survey, Coast Center for Operational Oceanographic Products and Services, 2000.

[9] 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.

[10] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha, "Fast proximity queries with swept sphere volumes," in *IEEE International Conference on Robotics and Automation*, 2000, pp. 3719–3726.

[11] ONR, "Naval science and technology strategic plan," Arlington, VA, 2015.

[12] G. Papadimitriou and D. Lane, "Semantic based knowledge representation and adaptive mission planning for mcm missions using AUVs," in *OCEANS*, 2014, pp. 1–8.

[13] E. Plaku and J. McMahon, "Motion planning and decision making for underwater vehicles operating in constrained environments in the littoral," in *LNCS Towards Autonomous Robotic Systems*, 2014, vol. 8069, pp. 328–339.

[14] S. Rashidian, E. Plaku, and S. Edelkamp, "Motion planning with rigid-body dynamics for generalized traveling salesman tours," in *ACM SIGGRAPH Motion in Games*, 2014, pp. 87–96.

[15] M. Wiig, T. Krogstad, and O. Midtgaard, "Autonomous identification planning for mine countermeasures," in *IEEE/OES Autonomous Underwater Vehicles*, 2012, pp. 1–8.