Motion Planning for an Underwater Mobile Manipulator by Exploiting Loose Coupling

Dina Youakim[1], Andrew Dornbush[2], Maxim Likhachev[2], and Pere Ridao[1]

[1]Computer Vision & Robotics Institute, University of Girona
[2]The Robotics Institute, Carnegie Mellon University

***Abstract*— Intervention Autonomous Underwater Vehicle or I-AUV has recently started to grab researchers attention in the last 20 years. Only three I-AUVs have demonstrated autonomous manipulation skills: ALIVE, SAUVIM and GIRONA 500. While prior systems rely on variations of the task-priority redundancy control framework, our recent research showed preliminary results using motion planning for floating-based intervention in the presence of obstacles. With the increasing need for autonomously performing more complex manipulation tasks, two main challenges need to be addressed: the high-dimensionality of the system, and the motion coordination between the mobile base and the working arm. The latter challenge is of high importance if accurate execution is required, especially considering the floating nature of the AUV and the control challenges that come with it. Our approach relies on exploiting the loose coupling between the AUV and the arm. In particular we present an approach based on MR-MHA* (Multi-Representation, Multi-Heuristic A*), and we show how it can generate efficient trajectories by exploiting decoupling. We show for the first time the use of a search-based planner on a high-dimensional underwater manipulator. In addition, we support our claims with experimental analysis of the generated trajectories with respect to various metrics in different environments. Furthermore, we demonstrate the ability of our approach to conduct a full intervention mission in a realistic simulated underwater intervention environment.**

I. Introduction

Nowadays AUVs are mostly used for survey missions, while many existing applications require intervention, like the maintenance of permanent observatories and pipes, the deployment and recovery of benthic stations, or the search and recovery of black-boxes. Currently, these tasks use work-class ROVs deployed from vessels equipped with dynamic positioning, leaving such solutions expensive to adopt.
In addition, performing such tasks, requires operating the I-AUV in unknown, and potentially cluttered and dynamic environments (e.g. near complex structures on the sea floor or underwater caves), and therefore are more exposed to collisions.

In the "Object Search And Recovery" application, the first result based on floating manipulation was achieved in 2009 in the SAUVIM project, where the position of the object of interest was known only roughly *a priori*. A similar experiment in a water tank was carried out in 2012 by RAUVI project. Later on, during the TRIDENT project, the experiment was extended to a harbour environment using a 7-DOF arm endowed with a 3 fingered hand. In contrast with the autonomous trials, innovative recent work [1] has been carried out to bring the ROVs to a new level, where a humanoid diving robot outfitted with human vision, haptic force feedback and an artificial brain, has been successfully used for recovery missions.
In the "Inspection Maintenance and Repair" application for the offshore industry, representative tasks usually performed by ROVs, are tasks such as "Valve Turning" and "Connector Plug/unplug". In previous work, such tasks have been automated with different approaches:
Fixed-Base Manipulation: The first fully autonomous intervention at sea, was demonstrated by ALIVE 1.5 Ton I-AUV. A mechanical scanning imaging sonar was used to locate a sub-sea panel, and visual servo-ing techniques were used for docking the vehicle using 2 hydraulic grasps. Once the vehicle was docked, a hydraulic 7-DOF manipulator was used to open/close a valve. A similar approach was proposed in the TRITON project. An active localization strategy employing a Sum-of-Gaussian filter and range measurements to the sub-sea panel was used to discover its position and then, to home onto it. Next, visual servo-ing methods based on the *a priori* known appearance of the panel were used to autonomously dock the robot into a funnel-shaped docking station. Besides demonstrating the valve turning task, a first autonomous demonstration of a connector plug/unplug operation was also carried out.
Free-Floating Base Manipulation: In this approach, the manipulation task takes places while the AUV is floating, hence, the arm motion may potentially disturb the AUV pose. The first autonomous free-floating valve-turning was carried out based on a Learning-by-Demonstration paradigm within the PANDORA project. More recently, a task priority redundancy control approach at kinematics level [2], has been used by the authors for the same purpose.

The previously developed solutions relied on variations of the task priority redundancy control framework. As a result these approaches are not able to address the operation in proximity of obstacles as would be expected in a real sea environment. As a consequence, using a motion planning based solution is becoming a necessity to address such a challenge.
To the best of our knowledge, only preliminary results have been shown using motion planning for underwater intervention in our previous work [3].

The main contribution of this work is the application of a search-based planner to an I-AUV, and further developing it to exploit the characteristics of the system, especially the loose coupling between the base and the arm. As experiments show, the planner generates more efficient trajectories in terms of path length and automatically figures out which motion set leads better to the goal. This is particularly useful in the context of floating-base manipulation where avoiding to move the AUV when the arm can do the job is preferred and vice-versa given the limitation of the arm reachability without moving the base. The simulation results show the potential of the new technique in generating shorter more efficient trajectories in different cluttered environments while maintaining the consistency of solutions (i.e. generates similar solutions to similar queries)

II. The Underwater Vehicle Manipulator System

The system used for validation (Fig. 1) consists of:

a) ***Girona500 AUV*** *[4]:* a 6 DoF, compact-size AUV with overall dimensions are of 1 m height, 1 m width, and 1.5 m length, and weighing about 140 kg weight. Its maximum operating depth is 500 m. The vehicle is passively stable in pitch and roll, making it suitable for tasks requiring a stable platform such as intervention. Its basic configuration is equipped with typical navigation sensors - DVL (Doppler Velocity Log), AHRS (Attitude and Heading Reference System), pressure gauge and USBL (Ultra-Short Baseline) - and basic survey equipment (profiler sonar, side scan sonar, video camera and sound velocity sensor).

b) ***ECA/CSIP Micro-Arm****:* a 4-DOF arm. It has 3 rotational joints actuated by electric screw drives to control the (x, y, z) position of the terminal element. A fourth actuator controls the roll (ϕ) of the end-effector. The arm is under-actuated and hence each 3D position can only be reached within a particular attitude. Full attitude control of the end-effector is only possible by the cooperative motion of the arm and the AUV.

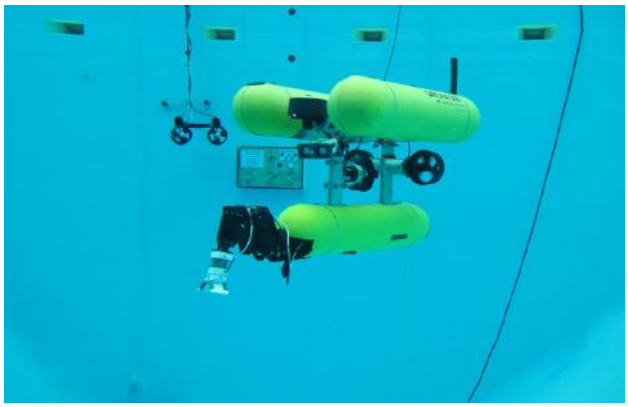


Fig. 1: Girona500 AUV mounted with the ECA/CSIP 4-DOF Micro-Arm.

Girona500 UVMS Modelling

Figure 2 presents the I-AUV kinematics chain as modelled in *MoveIt!* as an 8 DoF system. A North-East-Down $\{NED\}$ frame is used as the global reference frame, and the $\{R\}$ frame rotated with respect to the $\{NED\}$ is used for visualization purposes.

c) AUV Modelling: The vehicle is actuated only in 4-DOF ($surge$, $sway$, $heave$ and yaw). The AUV kinematics are modelled using the Denavit and Hartenberg (DH) method applied to the "equivalent" Cartesian manipulator shown in Fig. 2. The vehicle motion is modelled as three consecutive prismatic joints representing (X, Y, Z) displacements in the $\{NED\}$ frame. Followed by three rotational joints corresponding to (ϕ, θ, ψ) Euler rotations. Two of them are non-controllable (i.e. modeled as fixed joints). The ψ joint is modelled as an unlimited continuous one. The final configuration vector of the AUV is $Q = [q_1\, q_2\, q_3\, q_4]^T$ representing the controllable part of the AUV pose vector ${}^{NED}\eta = [x\, y\, z\, \psi]^T$.

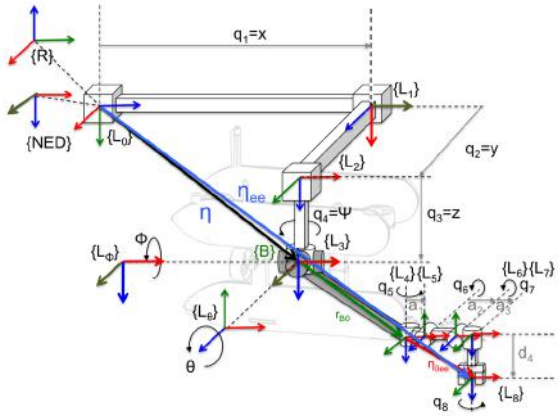


Fig. 2: I-AUV Kinematic chain Modelling as an 8 DoF system $\rightarrow$(x,y,z,ψ) of the AUV, followed by $[q_1\, q_2\, q_3\, q_4]$ of the arm

d) Arm Modelling: The arm is also modelled using the DH convention. Its links have associated frames $\{L_4, L_5, L_6, L_7, L_8\}$, with the first and the last associated to the arm base $\{0\}$ and end-effector $\{ee\}$ respectively. The arm joint variables $Q = [q_5\, q_6\, q_7\, q_8]^T$, are configured as rotational joints with their corresponding mechanical limits.

III. Multi-Representation Multi-Heuristic A* For an I-AUV

Search-based planners formulate the motion planning problem as a graph search to find an optimal path given certain criteria (commonly path length). They use common notations, as used throughout this paper, where: S denotes the finite set of states of the domain, $c(s, s')$ denotes the cost of the edge between states s and $s' \in S$; if there is no such edge, then $c(s, s') = \infty$. $Succ(s) := [s' \in S \mid c(s, s') \neq \infty$], denotes the set of all successors of s. (s_{start}, s_{goal}) pair represents the start and goal states of a given query. $c^*(s, s')$ denotes the cost of an optimal path from state s to s', $g(s)$ denotes the current best path cost from s_{start} to s, and $h(s)$ denotes the heuristic for state s, which is an estimate of the best path cost from s to s_{goal}. A heuristic is admissible if it

never overestimates the best path cost to s_{goal} and consistent if it satisfies, $h(s_{goal}) = 0$ and $h(s) \leqslant h(s') + c(s, s'), \forall s$, s' such that $s' \in Succ(s)$ and $s \neq s_{goal}$. An OPEN set is a priority queue, typically implemented as a heap, of states eligible for expansion, ordered by estimated path cost. A CLOSED set holds those states that have already been expanded. A typical search-based algorithm runs first by picking the minimum estimated-cost node from the OPEN queue, and expands this node by generating its successors. For each successor $s' \in Succ(s)$, if a less costly path to s' is found through this new expansion, its $g(s')$ is lowered and, if it has not been expanded yet, s' is added to the OPEN queue.

While optimal search-based planning approaches (e.g. A* [5]) are not feasible in high-dimensional planning, sub-optimal ones scale dramatically better. In our work, we show how to adapt recently developed *Muti-Representation, Multi-Heuristic A** that has been shown effective for high-dimensional planning in systems that can be decoupled under several lower-dimensional representations. In the following, we briefly describe Multi-Heuristic A* and then its Multi-Representation version, followed by the contribution of this work.

A. Multi-Heuristic A* Background

Multiple sub-optimal versions have been developed such as "Weighted A*" (WA*) [6] and its anytime variants [7], [8]. They compromise optimality with speed, and guarantee that the solution suboptimality is bounded by w (inflation of heuristic) if $h(s)$ is consistent.

While WA* has been shown to provide a speedup in many domains, it is heavily dependent on the heuristic function. Heuristics subject to local minima or where the estimated heuristic grows away from the realistic cost of the path to the goal, will dramatically degrade WA*'s performance [9].

Multi-Heuristic A* (MHA*) [10], [11], [12] is a recently developed search algorithm that builds on the observation that designing a single admissible and consistent heuristic that capturing all the complexity of a given domain is quite challenging. Instead of relying on a single heuristic, it incorporates several, possibly-inadmissible heuristics, and run multiple searches. MHA* enables the search to simultaneously use the guiding power of each heuristic around local minima, by exploiting their combined effort during the search, where each heuristic may be useful in certain parts of the search space.

Two variants have been proposed: "Independent MHA*" (IMHA*) which uses independent g and h values for each search, and "Shared MHA*" (SMHA*) which uses different h values but a single g value across all the searches. IMHA* works by independently exploring different heuristics through running $n + 1$ searches simultaneously, each with its own OPEN queue. In addition to the n searches, an *anchor search* is run. The anchor search is a single WA* with a consistent heuristic function h_0. A best-first expansion is performed from queues $OPEN_i$ $i = 1..n$, as long as $OPEN_i.Minkey() \leq w_2 * OPEN_0.Minkey()$, where w_2 is a used as a factor to prioritize an inadmissible search i over the anchor search. If the condition is violated, the anchor search is expanded instead. When a state s of $search_i$ is expanded, the generated successors $s' \in Succ(s)$ gets inserted in the corresponding $OPEN_i$ if it has not been expanded before by this i^{th} search. SMHA* extends IMHA* by sharing the current path of a given state among all the searches, i.e. if a better path is discovered by any of the searches, its information gets updated in all their corresponding queues. The expansion condition is similar to IMHA*, while the key difference lies in the expansion method. When a state s is expanded, its successors $s' \in Succ(s)$ are simultaneously updated in the $n + 1$ OPEN queues, if they have not been expanded before in any search. As exception if s' has been expanded in any of the n searches but not in the anchor, it gets inserted in its $OPEN_0$ only. SMHA* has been shown to be more adept at overcoming local minima as it combines partial solutions found by each search to reach the goal.

B. Multi-Representation MHA* Background

Multi-Representation, Multi-Heuristic A* (or *MR-MHA**) is particularly useful when planning for mobile manipulators and humanoids where the system has a decoupling nature (e.g. base, arms similar to the work in [13]). *MR-MHA** runs multiple searches, each search explores part of the problem by running on a partial-graph corresponding to a particular representation of the system, for example a mobile base, a leg or an arm. As a consequence, each expansion step is partial, and the successors generated belongs only to one group, thus leading to smaller graph size. Those limited graphs lead to faster search partially solving the high dimensionality problem, especially combined with domain-specific heuristics to ensure minimum state expansions.

Fig. 3 shows an illustrative example of the full-dimensional representation versus the MR approach. We assume a system of 15 DoF overall and composed of a mobile-base and two arms with 5, 3, and 7 DoF each. In the full-D representation (left of the figure), each expansion step generates n successors each a configuration of length 15 as well, with the n being the number of possible states from the current one. On the other hand, the multi-representation (shown on the right) consists of three sub-graphs (one per each entity: base, arm-1, arm-2). The dimension of the states in each sub-graph belongs to the number of DoF of each corresponding entity in the system. Thus both the red and green graphs belong to arm-1 & arm-2, while the blue represents the base. In order to combine the power of the sub-graphs to solve the search problem, a state sharing between the different searches is allowed in two fashions: *uni & bi-directional*. If a search "i" is bi-directional, its generated successors are inserted in every other bi-directional searches only, and allows any other search regardless of its type (bi/uni-directional) to access its OPEN queue by inserting their successors. On the other hand, a uni-directional search "j" inserts its generated successors in every other bi-directional search queues, but does not allow any other search to insert states into its own queue.

This distinction is also illustrated in Fig. 3, with single and double-headed arrows for uni & bi-directional sharing. The uni-directional version can be of great use to speed-up escaping local minima occurring due to state sharing.

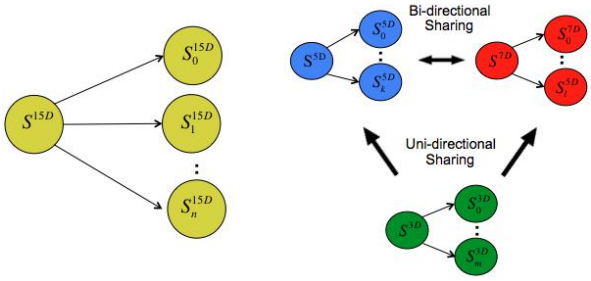


Fig. 3: Full-Dimensional Representation (left) vs. Multi-Representation (right) Schema

C. Planning for I-AUV using MR-MHA*

An I-AUV is a system composed of 2 heterogeneous components: 1) an AUV with a motion accuracy in the order of magnitude of several centimeters (in the best case) but having a wide workspace and 2) a manipulator with a motion accuracy below the millimeter but having a limited workspace. The motion planning algorithm has to decide which component has to move, the AUV (e.g. if the target is outside the arm work-space) or the arm (e.g. when the target is reachable) in order to reach goal. Therefore, optimizing the choice of the motion while producing high-quality trajectories is a requirement in our case. To achieve this purpose, our algorithm performs the desired motion decoupling through running multiple searches using multiple representations with different heuristics to exploit the power of both the base and the arm to get to the goal.

In the following, we first explain the algorithm details, and later the heuristics used by the proposed method.

1) Algorithm: The pseudo-code is detailed in Alg. 1, which is an instantiation of MR-MHA* explained previously. It operates on three queues: 1) bi-directional queue for searching in the base x, y, z, ψ representation, 2) uni-directional base queue searching in x, y, z, ψ as well, and 3) bi-directional queue for searching in the arm $Q = [q_5\ q_6\ q_7\ q_8]$ representation.

The anchor search, present in the MHA* algorithm, is omitted from the search process, since all heuristics used in this work are admissible. The algorithm main procedure *MR-MHA** starts with initializing search variables (lines 2-6) for all the searches. It then starts the expansion process in a Round Robin fashion, by alternating between the three $OPEN_i$ $i = 1..3$ queues as long as they are not empty (line 7). During the expansion, only the successors belonging to the current search representation are generated (line 13). Then, for each successor, its heuristic is computed given the current search representation j (line 18). If a lower g cost is found, the state gets updated in the current $OPEN_i$ queue, and in each other bi-directional queue (line 21). A state is inserted/updated in the j^{th} $OPEN$ depending whether it has been expanded before in this same search (line 23-26). The *GetSuccessorsByGroup* procedure (line 34-38) generates the successors depending on the requested search representation i, so either the base or the arm motions are considered (i.e decoupling the base and the arm search graphs and the resulting motion). The search terminates once the *f-value* of the goal state is less than the minimum f-value in the OPEN list (line 10).

Algorithm 1 Multi-Graph Multi-Heuristic A*

```
1:  procedure MR-MHA*(s_start, sgoal)
2:      g(s_goal) = ∞
3:      g(s_start) = 0
4:      for each j = 1...3 do
5:          OPEN_i = ∅
6:          insert s_start in OPEN_i with key(s_start, i)
7:      while all OPEN_i not empty do
8:          for each j = 1...3 do
9:              s_min = OPEN_i.MinKey()
10:             if f_i(s_min) ≤ f_i(s_goal) then
11:                 return solution
12:             expand(s_min,i)
13: procedure expand(s,i)
14:     succ(s) = GetSuccessorsByGroup(s, i)
15:     for each s' ∈ succ(s) do
16:         g(s') = ∞
17:         for each j = 1...3 do
18:             f_j(s') = ∞
19:             h_j(s') = GetGoalHeuristic(s', j)
20:         if g(s') > g(s) + cost(s, s') then
21:             for each j = 1...3 do
22:                 if i == j or j is bidirectional then
23:                     f_j(s') = key(s', j)
24:                     if s' has not been expanded in j then
25:                         Insert (s') in OPEN_j
26:                     else
27:                         Update (s') in OPEN_j
28: procedure key(s,i)
29:     return g(s) + w * h_i(s)
30: procedure GetGoalHeuristics(s,i)
31:     if search i corresponds to base representation then
32:         return GetBaseHeuristic(s)
33:     else
34:         return GetArmHeuristic(s)
35: procedure GetSuccessorsByGroup(s,i)
36:     if search i corresponds to base representation then
37:         return ComputeBaseMotionPrimitives(s)
38:     else
39:         return ComputeArmMotionPrimitives(s)
```

2) Heuristics Computation: A key to the success of the search-based motion planners, is the design of an informative, fast-to-compute heuristic capable of capturing the domain at hand. In the work of [14], a *breadth-first-search (BFS)* heuristic has been introduced in the context of plan-

ning for manipulation and shown to be more informative compared to the commonly used Euclidean one. The idea is to keep track of the obstacles information in the environment through a 3-D grid with cells marked free/occupied. This grid has the x, y, z end-effector goal as the start cell for the BFS. An inflation radius is specified for the end-effector - given its dimensions - and it is used to inflate the grid cells. Those cells lying in the area within this radius are marked as occupied, helping to guide the end-effector in cluttered regions. As a result, BFS computes distances for all the cells in the constructed 3-D grid that correspond to the lengths of paths for the end-effector to reach its goal while avoiding obstacles.

The base heuristic uses the same BFS concept. We run 3-D BFS on a separate base grid where all obstacles are inflated by the inner radius of the AUV. In this case, the goal is not a single cell, but a region computed based on an estimation of the extended arm length and transformed to the origin of the base (or B in Fig. 1). Fig. 4 shows an example of the base goal region (red grid), given an end-effector goal (blue sphere), whereas the end-effector reaches the goal, the base frame B is intersecting some of the grid cells.

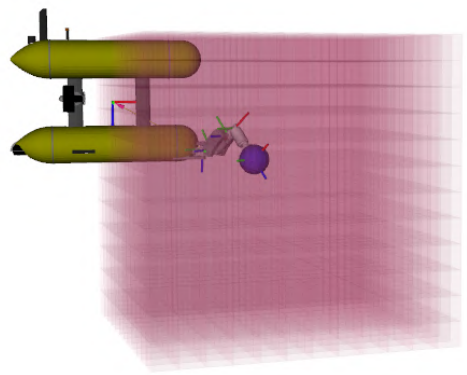


Fig. 4: Base-Goals Cells (red grid) for a Given EE Goal (blue sphere) with an example configuration lying on the grid

IV. Experimental Results

Our experimental analysis consists of two parts: In the first one, we did an analytical comparison of our technique against five other common planners, three of which are sampling-based (RRT [15],$RRTConnect$ [16],RRT^* [17]), one optimization-based planner ($STOMP$ [18]), and a common search-based planner (ARA^*). The evaluation consists of comparing four metrics based on 20 runs of each planner in each of the five benchmarks: planning time, path length, success rate, and consistency. Each planner was allowed at most 30 sec for planning. In addition, the reported solutions by each planner have been post-processed (e.g. shorcutting), and the reported planning time includes the post-processing step. It is worth noting that for both RRT^* & ARA^* results are reported for the first solution found without further improvements to ensure fair comparison with other planners.

The second part of the analysis consists of a simulation of a real underwater manipulation mission in a typical sub-sea structure like those used by the OilGas industry. There, pipe inspection and connector plug/unplug missions are simulated. The implementation is based on *MoveIt!* [19] benchmark plugin, *OMPL* [20] and search-based motion planning library *SMPL* [21].

A. Benchmarks

In order to evaluate the performance of our algorithm, we developed five benchmarks (shown in Fig. 5), similar to previous work demonstrated on other manipulation domains, and adapted to the underwater environment.

Both BM#1 and BM#2 are simulation of a common underwater intervention, consisting of turning a valve in an offshore sub-sea panel, with and without obstacles. BM#3 and BM#4 simulate a task of moving an elongated object (stick) through a window-like opening. BM#5 performs the same operation but going through a space cluttered with obstacles. The stick length in BM#3 and BM#5 is 50 cm, which is smaller then the window width. For BM#4 the stick is 1m length ($>$ window width) and the initial I-AUV configuration is close to the window, preventing the possibility of passing the stick through the window.

The comparison results are shown in Tables I,II, and III. From the first table, our approach results in shorter path length (measured as a combination of the L2 norm of both translational and rotational joints) compared to all other planners, with the exception of the first benchmark. A reason for this behavior, is the simplicity of the scenario with wide obstacle-free areas around, where all planners after post-processing result in similar solutions. On the other hand, with more challenging benchmarks, the success rate of some sampling-based planners decreases, whereas the ARA^* showed consistent failure in BM#4 & BM#5. This proves the improvement of our proposed approach in the scenarios where the direct path to the goal is blocked by obstacles or not feasible, and moving the base alone is necessary to explore other areas. A drawback is seen in BM#3 where ARA^* is able to find the direct path through the window faster than the proposed approach, and expands much less nodes (as shown in Table III). Fig. 6 shows solutions of BM#4 & BM#5, where the states in red have been found by search operating on the base representation and states in blue have been found by the search on the arm representation. It can be appreciated the base support getting around the blocked areas and advancing the search forward finding its way to the goal.

To perform a consistency test, the start configuration per run per benchmark has been randomly modified in the exact same way for all the planners, and within a range of (20cm,5°) for the translational and rotational joints respectively, and then the test was performed. Finally the reported numbers are the accumulation of the *Frechet Distance* between the combination of the N found motions (i.e. the joint-space plan), where N is the number of runs (i.e. 20). Furthermore, the success rate is reported as different failures were

encountered due to the start configuration change. In table II, "bold" is used to denote the highest consistency (lower Frechet distance) as long as success rate is higher than 90% . As expected, due to their deterministic nature, search based planners demonstrated higher consistency with the MR-MHA* outperforming.

B. Application Example

An additional validation has been performed in order to illustrate the capability of the proposed method to plan a representative underwater intervention mission. A model of a real sea structure typically involved in such missions, is used as an a-priori known octomap. We assume a mission consisting of two common intervention tasks: 1) Pipe Inspection & 2) Connector unplug/plug, with the corresponding goals labeled as $B(1..3)$ & $C(1..2)$ in Fig. 7. For the sake of simplicity same manipulation object was used for both, while in reality, a bar with a camera fixed on its tip is required to perform the inspection.

Fig. 7 shows some of the I-AUV configurations during the execution with the start configuration shown in sub-figure(1). Then, the inspection intervention consists of three tasks (i.e. three start to goal planning): first to pick the bar from its initial position (figures 2,3) with planning Time = 4.8 sec, second to inspect pipes on the upper part of the structure (figures 4,5) and it took 6.8 sec to plan this task. Finally, moving to the side of the structure holding the bar (figures 6,7) with a planning time of 14.3 sec. After terminating the inspection, the bar is placed on the structure to start the new intervention.

For the second intervention, the connector has to be picked and un-plugged (figures 8,9), and this phase took 1.6 sec to plan. At last, the connector is plugged in one of the destined holes for connectors (as shown in figures 10,11), with 1.7 sec planning time. We can conclude that the planner is able to cope with complex tasks, given that the overall mission (i.e. 5 tasks) planning time is almost 30 sec, which would enable real-time performance in real environments.

V. CONCLUSIONS

We presented an approach to planning for an underwater manipulator using Multi-Representation, Multi-Heuristic A*. The approach exploits loose coupling between the base and the arm when performing mobile manipulation tasks. This is particularly useful given the floating nature of the base (i.e. harder accuracy of execution) and the limited arm reachability. Through the benchmarks experiments, we have evaluated the quality of the produced trajectory in terms of path length while keeping relatively fast planning time. In addition we showed a successful simulation of a real mission conducted over an actual underwater structure.

In our future work, we plan to do water tank experiments. This step will require resolving some challenges due to the uncertainties involved (both sensing and actuating). For that the framework needs further development to be endowed with the proper reactions (e.g. reactive behavior at the control level to ensure safety, and fast re-planning capabilities). In a later stage, we would further develop the framework to cope with dynamic environments.

REFERENCES

[1] O. Khatib, "Ocean One." `https://cs.stanford.edu/groups/manips/ocean-one.html`.

[2] P. Ridao, M. Carreras, D. Ribas, P. J. Sanz, and G. Oliver, "Intervention auvs: The next challenge," *Annual Reviews in Control*, vol. 40, pp. 227 – 241, 2015.

[3] D. Youakim, P. Ridao, N. Palomeras, F. Spadafora, D. Ribas, and M. Muzzupappa, "Autonomous underwater free-floating manipulation using moveit!," *IEEE Robotics & Automation Magazine*, 2017.

[4] D. Ribas, N. Palomeras, P. Ridao, M. Carreras, and A. Mallios, "Girona 500 auv: From survey to intervention," *Mechatronics, IEEE/ASME Transactions on*, vol. 17, no. 1, pp. 46–53, 2012.

[5] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[6] I. Pohl, "First results on the effect of error in heuristic search," *Machine Intelligence*, vol. 5, pp. 219–236, 1970.

[7] B. Bonet and H. Geffner, "Planning as heuristic search," *Artificial Intelligence*, vol. 129, no. 1-2, pp. 5–33, 2001.

[8] M. Likhachev, G. J. Gordon, and S. Thrun, "Ara*: Anytime a* with provable bounds on sub-optimality," in *Advances in neural information processing systems*, pp. 767–774, 2004.

[9] C. M. Wilt and W. Ruml, "When does weighted a* fail?," in *SOCS*, pp. 137–144, 2012.

[10] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic a*," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 224–243, 2016.

[11] M. Phillips, V. Narayanan, S. Aine, and M. Likhachev, "Efficient search with an ensemble of heuristics.," in *IJCAI*, pp. 784–791, 2015.

[12] V. Narayanan, S. Aine, and M. Likhachev, "Improved multi-heuristic a* for searching with uncalibrated heuristics," in *Eighth Annual Symposium on Combinatorial Search*, 2015.

[13] A. Dornbush, K. Vijayakumar, S. Bardapurkar, F. Islam, and M. Likhachev, "A single-planner approach to multi-modal humanoid mobility," *arXiv preprint arXiv:1801.10225*, 2018.

[14] B. Cohen, S. Chitta, and M. Likhachev, "Single- and Dual-Arm Motion Planning with Heuristic Search," *The International Journal of Robotics Research*, 2013.

[15] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[16] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, pp. 995–1001, 2000.

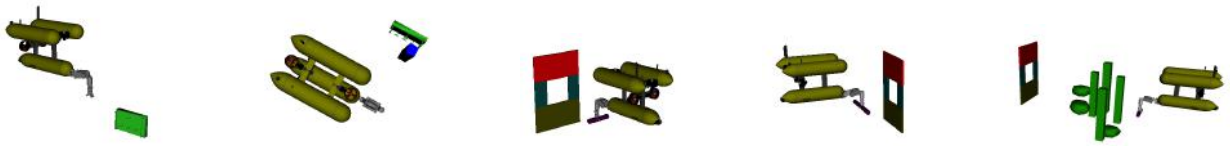
[17] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[18] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 4569–4574, IEEE, 2011.

[19] S. C. I. A. Sucan, "MoveIt!, Available Online:." `http://moveit.ros.org`, 2013.

[20] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, pp. 72–82, December 2012. `http://ompl.kavrakilab.org`.

[21] A. Dornbush, "SMPL Library." `https://github.com/aurone/smpl`, 2016.



(a) BM#1: Simple Valve Turn (b) BM#2:Valve Turn With Obstacle (c) BM#3: Stick 50 Environment (d) BM#4: Stick 1m Manipulation (e) BM#5: Stick 50cm with Scattered Obstacles

Fig. 5: Benchmarks representing various test environments used for simulation validation

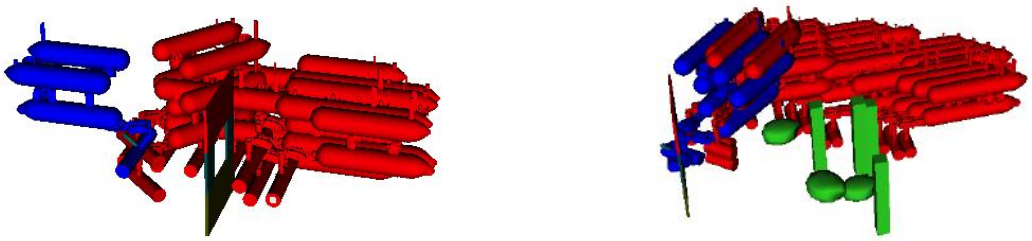


Fig. 6: Found Solutions for BM#4 & BM#5 (base motions in red & arm motions in blue)

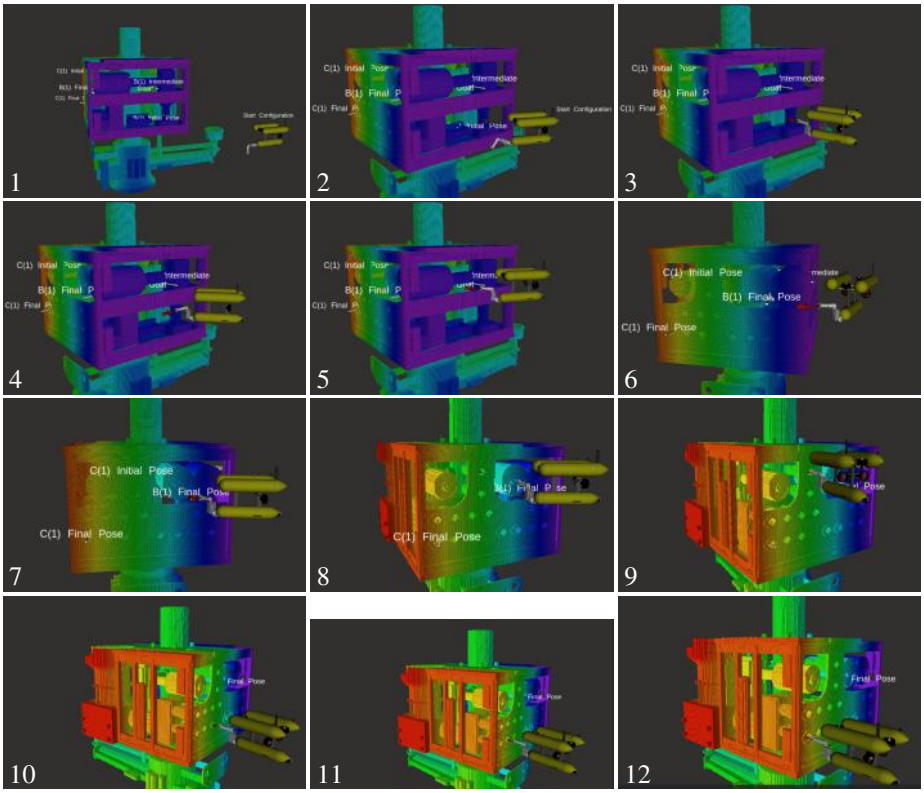


Fig. 7: Parts of the Different Trajectories during the 5 Phases of the Mission.

Planner	BM#1			BM#2			BM#3			BM#4			BM#5		
	Time	Length	%	Time	Length	%	Time	Length	%	Time	Length	%	Time	Length	%
RRT	0.087	5.5	100	0.84	8.2	100	0.068	24.12	100	**0.045**	18.74	95	5.6	11.15	65
RRTConnect	**0.02**	**5.43**	100	**0.036**	13.8	100	**0.03**	19.8	100	0.05	17.94	100	**0.035**	16.4	90
RRT*	0.06	5.68	100	1.8	15.1	100	0.23	18.48	100	0.05	22.7	100	3.34	14.3	100
STOMP	2.68	6	100	3.65	6.25	100	3.38	4.45	25	F	F	0	23.8	11.16	10
ARA*	0.087	7.1	100	0.12	**5.88**	100	0.25	6	100	F	F	0	F	F	0
MR-MHA*	0.12	5.9	100	0.15	5.9	100	2.43	**4.2**	100	0.94	**14.98**	100	0.39	**9.56**	100

TABLE I: Planning Time, Path Length & Success Rate

Planner	BM#1		BM#2		BM#3		BM#4		BM#5	
	C.I	Success%	C.I	Success%	C.I	Success%	C.I	Success%	C.I	Success%
RRT	319	100	475	100	802	100	1026	100	158	45
RRTConnect	149	100	438	100	706	100	1492	100	829	100
RRT*	219	100	311	100	672	100	1126	100	660	100
STOMP	228	100	135	95	0.0	5	18	15	3.98	20
ARA*	115	100	171	100	154	100	F	0	181	65
MR-MHA*	**96**	100	**125**	100	**69**	100	**804**	100	**249**	95

TABLE II: Consistency

Planner	# Expansion				
	BM#1	BM#2	BM#3	BM#4	BM#5
ARA*	7	9	9	F	F
MR-MHA*	12	12	340	80	50

TABLE III: Average Number of Expansions for search-based Planners