IEEE Robotics and Automation Magazine (RAM) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

An Autonomous Underwater Architecture for Long-term Deep-Ocean Inspection with Opportunistic (Re)planning*

Elisa Tosello[1], Paolo Bonel[2], Alberto Buranello[2], Marco Carraro[2], Alessandro Cimatti[1], Lorenzo Granelli[2], Stefan Panjkovic[1] and Andrea Micheli[1]

***Abstract*— Robots are increasingly used in subsea environments due to their positive impact on human safety and operational capabilities in the deep ocean. However, achieving full autonomy remains challenging due to the extreme conditions they encounter. In this context, we propose an Autonomous Underwater Architecture for long-term deep-ocean inspection that robustly plans activities and efficiently deliberates with no human help. It combines the innovative Saipem's Hydrone-R subsea vehicle with an advanced planning architecture, resulting in a robot that autonomously perceives its surroundings, plans a mission, and adapts in real-time to contingencies and opportunities. After describing the robot hardware, we present the technological advancements achieved in building its software, along with several compelling use cases. We explore scenarios where the robot conducts long-term underwater missions operating under resource constraints while remaining responsive to opportunities, such as new inspection points. Our solution gained significant reliability and acceptance within the Oil & Gas community, as evidenced by its current deployment on a real field in Norway.**

I. Introduction

Subsea pipelines are crucial for oil and gas transport, with damage posing severe environmental and financial risks. Regular inspections, using internal pegs and crawling robots or external sensors like electromagnetic and acoustic, are essential to ensure integrity. This paper focuses on external inspections, aiming to optimize procedures and provide adaptable technologies for asset owners and operators.

Unmanned Underwater Vehicles (UUVs) are crucial in this regard as they can operate for an extended duration in deep waters, surpassing the depths reachable by human divers. Their usage is independent of the field, and they can promptly react to changes in the setup or assignment. We distinguish between Remotely Operated Vehicles (ROVs), which are controlled remotely and need continuous power from supporting vessels, and Autonomous Underwater Vehicles (AUVs), which operate autonomously. AUVs are particularly compelling for our application as they enhance efficiency and safety by minimizing human involvement in demanding and high-risk tasks. Additionally, they help energy companies in reducing their ecological footprint.

For optimal performance, AUVs must exhibit deliberate behaviors, taking actions driven by specific objectives backed by rational reasoning aligned with these objectives [1]. For instance, they should monitor resources like battery life, optimize consumption, adapt to environmental changes, and make informed decisions in response to unforeseen events.

We propose a hardware and software architecture that enables deliberative deep-sea inspection for extended periods. The robot is the Saipem's Hydrone-R: a drone equipped with innovative hardware designed to withstand the extreme conditions of the deep ocean and make the system a subsea *resident*. Hydrone-R is currently operating autonomously from June 2023 on a real offshore Oil & Gas field in Norway[1], with the goal of achieving ten years of service (see Figure 1). It is equipped with a range of specialized sensors, computing power, advanced control systems, and AI-driven navigation and inspection modules that leverage feature recognition. Such components empower the robot with the ability to autonomously perceive its surroundings, detect pipelines, precisely localize itself and accurately estimate its internal state. We put on board a task planner that exploits this information to plan safe inspection missions while adhering to resource constraints. An Orchestrator manages the mission and triggers real-time replanning for contingencies, like battery issues, or emerging opportunities, like inspecting new locations. We perform planning via the Unified Planning (UP) library[2], an open-source Python3 library developed by the AIPlan4EU project[3]. It supports modeling and solving classical, numerical, and temporal assignments [2], as well as complex problems such as multi-agent and task and motion planning ones. We use the library to enable Hydrone-R to plan and optimize numerical and temporal missions, such as managing resource constraints, prioritizing targets, and defining ordering rules among the goals. These features promote the vehicle's long-term stay in the deep ocean.

The paper is organized as follows. The next section summarizes related work. Section III covers the robot hardware, and Section IV defines the planning problem for autonomous deep-sea inspection. Sections V, VI, and VII detail the UP library, planning architecture, and methods for handling contingencies and new opportunities. Experimental results are in Section VIII, followed by conclusions in Section IX.

II. State-of-the-art

The number of available AUVs is growing. For instance, the KM's HUGIN AUV [3] is effective for subsea pipeline in-

[1]Fondazione Bruno Kessler, Trento, Italy
[2]Saipem s.p.a. - Sonsub Robotics, Venice, Italy

*This paper is a reduced version of E. Tosello et al., "Opportunistic (Re)planning for Long-Term Deep-Ocean Inspection: An Autonomous Underwater Architecture," in IEEE Robotics & Automation Magazine, vol. 31, no. 1, pp. 72-83, March 2024, doi: 10.1109/MRA.2024.3352810.

[1]See https://www.saipem.com/en/projects/hydrone-njord-field-development
[2]See https://github.com/aiplan4eu/unified-planning
[3]See https://www.aiplan4eu-project.eu/

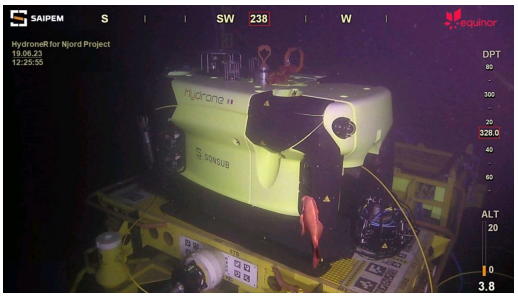

Fig. 1. Hydrone-R been deployed in the Norwegian waters in June 2023.

spection but requires constant forward motion due to its torpedo shape. Saab's Seaeye Sabertooth [4] and Girona 500 [5] are designed for asset inspection, with the latter tailormade and used in academia. None offer *true* autonomy, being unable to adjust missions in real-time, or long-term subsea deployment. In this context, the Saipem's Hydrone® family currently consists of Hydrone-W, FlatFish, and Hydrone-R, all using the same proprietary control system and algorithms but tailored to different offshore energy needs. Hydrone-W is a fully electric ROV with autonomous capabilities designed only to assist human pilots in demanding conditions. FlatFish performs long fully autonomous missions, and Hydrone-R, functioning as both ROV and AUV, can be teleoperated or autonomously inspect pipelines, risers, and perform surveys. It has been a subsea resident in Norwegian waters since June 2023, with a planned 10-year mission.

Focusing on decision-making, Albiez et al. Albiez et al. use behavior-based methods and a plan manager to deploy and manage AUV tasks, enabling effective mission execution and handling uncertain situations [6]. Mahmoudzadeh et al. integrate deliberative and reactive layers to manage scheduled tasks and real-time reactions [7]. Cashmore et al. treat AUV missions as temporal planning problems, dynamically addressing high-utility opportunities while achieving core goals [8]. Finally, [9] presents a fully automated task allocation algorithm for AUVs, modeled as a Traveling Salesman Problem, optimized using an Ant Colony Optimization algorithm with fuzzy logic and dynamic pheromone volatilization. In contrast, our planning architecture, using the UP library, gives generality to the problem addressed, allowing the robot to solve numerical and temporal planning problems. Our goal definition allows to assign resource constraints to each mission, like restrictions on the battery, time, and disk space. We can sort the inspection targets in sequences, temporally concatenate them, or arrange them in a partial order fashion, facilitating the incorporation of optional goals and the definition of a custom priority hierarchy among them. We can replan for new opportunities and contingencies via a close interaction between the UP library and the robot's Orchestrator. This system is compatible with the entire Hydrone® family, showing its highest utility on Hydrone-R.

III. The Underwater Vehicle

This section introduces Hydrone-R and the tools supporting its planning module and deliberate actions. As both an ROV and an AUV, it can operate autonomously or via teleoperation, handling pipeline inspections, riser tracking, and complex surveys. Details of its components follow.

Optical communication. An optical modem enables wireless control of the vehicle from its docking station or locations where operators need to receive real-time feedback.

Acoustic communication. A low-bandwidth, long-range acoustic modem enables operators to receive diagnostic data and send task updates to the robot during missions. It works also as an Ultra Short Base Line positioning system.

Advanced underwater vision system. Hydrone-R mounts two front cameras, a bottom camera, and a laser projector to create 3D point clouds. Lights enhance visibility, enabling pipeline tracking, obstacle avoidance, anomaly detection, and structure recognition for position data.

Sonars. In subsea environments, limited light and water particles restrict vision to a few meters, making long-range sonars essential. Hydrone-R is equipped with a frontal multibeam imaging sonar and a top-mounted scanning sonar, providing 360° situational awareness around the vehicle.

Advanced navigation system. Hydrone-R has an IMU sensor that combines optical fiber gyroscopes and solid state accelerometers aided, through data fusion algorithms, by a Doppler Velocity Logger, an Ultra-Short BaseLine, and the aforementioned vision system. The system limits the navigation error to less than 0.05% of the traveled distance.

Internal computational power and storage resources. Computing systems host sufficient power and storage resources for the robot to safely carry out assigned missions.

Manipulators. Hydrone-R is equipped with a light manipulator and a grabber. At the time of writing, manipulation operations are performed in a teleoperated fashion.

Integrated Tether Management System. To work in ROV mode, a tether and a tether management system provide real-time data exchange and battery charging power when connected to the host base of the subsea system. The robot can connect to multiple bases, according to mission needs.

Skid integration and vehicle capabilities extension. A skid is a set of sensors and electronics that you can attach to the bottom of the vehicle to extend its capabilities. Data and power flows occur thanks to inductive connectors.

Inductive connectors. An inductive connector enables tether and skid connection, offering up to 100MBps data transfer and 2kW power for charging. This contactless design allows more mating cycles than wet-mateable connectors.

IV. Problem formulation

To autonomously inspect pipelines, Hydrone-R must compute necessary actions, effectively manage resources, and minimize makespan. If new opportunities or contingencies occur during execution, e.g., a new cluster needs to be examined or the battery level drops unexpectedly, the robot replans to maximize the final outcome while preserving its safety, e.g., it may decide to return to the dock station. This is an *Automated Planning* problem, where *Automated planning* is a branch of AI that, given a model of a system (the robot and its capabilities) and a goal to reach (inspection of a set of locations), finds a course of actions to drive the system from

its current state to the target. When state variables include only rational numerical values, like the battery level, the problem to be solved is a *numeric planning* problem.

Def. 1: A *numeric* problem Ψ is a tuple $\langle V, I, A, G\rangle$, with:

- $V = \{f_1, .., f_n\}$ a finite set of variables (or fluents) $f \in V$, each with a domain $Dom(f)$.
- I the initial state, which assigns a value $I(f) \in Dom(f)$ to each variable $f \in V$.
- A a set of actions $a = \langle P, E\rangle \in A$ such that:
  - P is a set of preconditions, each been a combination of atoms of the form $f \bowtie v$, with $\bowtie = \{=, <, \leq, >, \geq\}$, $f \in V$, and $v \in Dom(f)$;
  - E is a set of instantaneous effects of the form $f := v$, with $f \in V$ and $v \in Dom(f)$.
- G the set of goal conditions, each been a combination of atoms of the form $f = v$, with $f \in V$ and $v \in Dom(f)$.

A plan π that solves Ψ is a sequence of actions $\{a_0, \ldots, a_m\}$ that brings the system from I to G by linking the effects of a_i with the preconditions of a_{i+1}.

When including timed goals, time constraints or durative actions, we enter the field of *temporal planning*.

Def. 2: A *temporal* problem Φ is $\langle V, T, I, A, G\rangle$, with:

- V, I, and G defined as before.
- T a set of timed-initial-literals, each of the form $\langle [t] f := v\rangle$ with $f \in V$, $v \in Dom(f)$, and $t \in \mathbb{R}_{>0}$ the time at which f assumes the value v.
- A the set of actions $a = \langle [l, u], C, E\rangle$, where $[l, u]$ are the duration bounds, C is the set of conditions, each one with start and end points, and E is the set of effects, each becoming true at a certain time instant [2].

π that solves Φ is a sequence $\{\langle t, a, d\rangle\}$, where a is paired with a start time instant $t \in \mathbb{R}_{\geq 0}$ and a duration $d \in \mathbb{R}_{>0}$.

We aim to solve both Ψ and Φ in the context of subsea pipeline inspection. Moreover, the robot must autonomously decide which location to inspect based on its significance and available resources. The goal becomes $G = G_h \cup G_s$, where G_h is the set of hard goals that must be achieved. G_s, instead, is the set of soft goals $\langle g_s, c(g_s)\rangle$ that the system may achieve: for each $g_s \in G_s$, there exists a cost $c(g_s) \in \mathbb{R}_{>0}$ to be paid if g_s is not achieved by π. The problem becomes a *planning problem with oversubscription* [10]. Finally, we allow to define an ordering rule $o = g_i < .. < g_j$ among the goals in G so as to prioritize inspection points. Our goal becomes to find π that complies with o, achieves all of the hard goals, and pays the minimum cost $c(\pi) = \min \sum c(g_s)$.

A *new opportunity* (e.g., inspection of a new cluster) is a soft goal $\langle g_s, c(g_s)\rangle$ created at runtime by an event $E = \langle [t] f := v\rangle$ (i.e., action effect or external occurrence), which instantiates a fluent $f \in V$ via $v \in Dom(f)$ [11]. The problem becomes an *opportunistic planning problem with oversubscription* $\langle \Psi, A', G'\rangle$ (or $\langle \Phi, A', G'\rangle$), where $A' \in A$ is the subset of actions that are preemptable and G' is the set of new opportunities. If an opportunity is discovered, the initial state I is updated and replanning is triggered. Being in a nondeterministic world, contingencies result in the same procedure and A' collects the set of actions that are preemptible to handle both opportunities and contingencies.

In the following sections, we present the planning library used to solve the problem and the planning architecture implemented to find π and monitor its execution.

V. The Unified Planning Library

To enable autonomous decision-making for our robot, we adopted the Unified Planning (UP) library[2]. This library is open-source, reusable, and planner-agnostic, facilitating the modeling, manipulation, and solving of various planning problems, including classical, numerical, temporal, multi-agent, and task and motion planning. It is part of AI-Plan4EU[3], an H2020 project that aims to enhance accessibility, reuse, and integration of planning algorithms and data.

The library's APIs provide users with the capability to model planning problems manually, parsing a formal language (e.g., PDDL [12] or ANML [13]), exploiting the interoperability interfaces with other frameworks (e.g., Tarski [14]), or mixing such approaches. It allows guiding plan search with *Custom Heuristics* to evaluate state quality. Users can specify *Quality Metrics* to optimize criteria, like minimizing steps or action costs. It allows to define optional goals with associated costs if they are unmet (*Oversubscription*). Once all specifications have been outlined, the UP library passes them to the connected planning engines via *Operation Modes*. Each engine solves a specific planning problem (e.g., numerical, temporal, etc.), and each operation mode offers an abstract interface to such engines and gives access to their functionalities. The simplest mode is *OneshotPlanner*, used for one-time plan generation. We also provide ways to *validate* a plan, *simulate* it, or trigger *replanning*.

These capabilities make the UP library ideal for planning in diverse domains, including autonomous ocean inspection.

VI. Our Advanced Planning Architecture

To enable autonomous deliberation, we designed a new goal-specification grammar and developed a planning architecture that takes these goals, retrieves resources, plans the mission, and commands the robot to execute it.

The grammar represents a set of goals to be achieved. Each <*single_goal*> specifies an <*expression*> to **reach**, such as a system state (e.g., "*switch from ROV to AUV*") or an environmental condition (e.g., "*valve 1 must be closed*"). Targets can have resource (**while** <*expression*>) or time (**within** <*interval*>) constraints, like limiting battery usage or makespan. Goals can be structured in sequences (each goal has a unique <*id*>), ordered (<*ord_constrs*> *::=* <*id*> **<** <*id*>), and concatenated (<*ord_constrs*> **and** <*ord_constrs*>). This allows optional goals (**optionally**) and custom priority hierarchies (**optionally with priority** <*number*>). For example, "*reach valve1-closed within [0, 30] while battery* $\geq$ *50*" instructs the robot to close *valve1* within 30 minutes while keeping the battery above 50%. The *optionally* keyword marks goals as desirable but not mandatory, e.g., "*optionally reach valve2-photographed*" indicates that photographing *valve2* is preferred but not required.

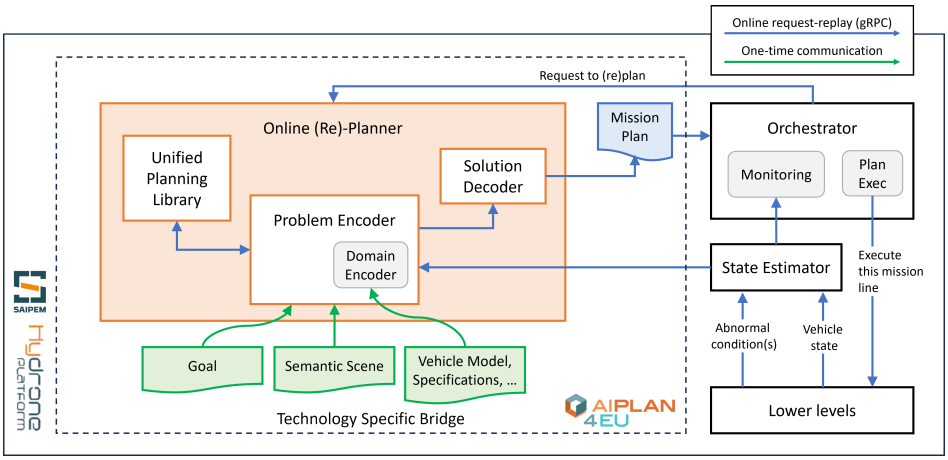

Fig. 2. Our software architecture. The Online (Re)-Planner takes in the robot model, current state, scene, and goal. It combines this data and models the planning problem via Problem Encoder, finds a plan through the UP library, and sends it to the Orchestrator via Solution Decoder. The Orchestrator sends the plan to the lower levels, manages its execution, monitors the robot state via State Estimator, and requests real-time replanning when needed.

Algorithm 1 Online (Re)-Planner

```
1  connect StateEstimator
2  connect Orchestrator
3  procedure REPLAN(V, T, A, G, G', mode)
4      p, R, G̅ ← StateEstimator.getCurrentState()
5      I ← < p, R>
6      Φ ← < V, I, T, A, G >
7      π ← Planner.solve(Φ, mode)
8      if π ≠ None then
9          E' ← []
10         E ← getEvents(π)
11         for each e ∈ E do
12             E'.push(e)
13             I ← Simulator.apply(I,e)
14             for each r ∈ R do
15                 v ← I.getValue( r)
16                 E'.push(check(r, v))
17         π' ← getPlan(E')
18         ρ ← decode(π')
19         out ← Orchestrator.startMission(ρ)
20         if out == replan then
21             if replan == contingency then
22                 return REPLAN(V, T, A, G - G̅, G', replanner)
23             else if replan == opportunity then
24                 return REPLAN(V, T, A, G - G̅ + G', ∅, replanner)
25         else if out == success then
26             return True
27         else
28             return False
29     else
30         return False
```

The Online (Re)-Planner models the problem via Problem Encoder, finds a plan through the UP library, and sends it to the robot via Solution Decoder (see Algorithm 1). An Orchestrator, instead, manages the execution, monitors the robot state, and hands unforeseen situations by triggering safety operations or requesting real-time replanning (see Figure 2). The following section details the Orchestrator, while each component of the Online (Re)-Planner is described below.

State Estimator. It retrieves the current state of the robot, that is its current pose p, the available resources R (e.g., battery charge), and the goals $\overline{G}$ already visited (line 4). <p, R> forms the initial state I of the system (line 5). Together with $\overline{G}$, they will be updated at each step of the mission.

Problem Encoder. The Encoder receives the initial state I, current goal G, and robot specifications, i.e., the logical and temporal expressions representing the set of actions A that the AUV can perform, like moving from a start location s to a goal g. It combines this information with the set of fluents V (e.g., resource variables) and any timed-initial literals T. Then, it models the problem Φ (or Ψ) using the UP library API (line 6). It sends Φ to the UP library and waits for a plan π, if one exists (line 7). The Encoder decorates π with *check* events to verify if actual resource values match those estimated by the UP *Simulator* at corresponding time instants. This helps identify deviations and promptly triggers replanning in case of unforeseen events. In case of numeric planning, π is an ordered sequence of actions and we can add checkpoints at the end of each of them. When considering time, instead, actions may overlap: we must first transform π into an ordered list of events (the effects of the actions), wherein the order reflects the time instants at which these events are scheduled. We enrich this list with the checkpoints and transform it back into a plan (lines 9-17). The Encoder sends the decorated plan π' to the Solution Decoder for its translation into a robot-compliant mission ρ. If the Orchestrator triggers a replan request while executing ρ, the Encoder retrieves the new initial state I, the available opportunities G', and the goals $\overline{G}$ already achieved, and sends this data to the UP library asking for replanning.

Unified Planning Library. When the UP library receives a problem, it uses *OneshotPlanner* to solve it - with *Oversubscription* for optional goals (line 7 with *mode* = *OneshotPlanner*). In case of unforeseen events, instead, UP triggers *Replanner*, which updates the initial state of the robot and the goals to be reached according to the current situation, and tries to solve the new problem (lines 20-24). In case of numeric planning, *OneshotPlanner* solves the problem via Expressive Numeric Heuristic Search Planner (ENHSP) [15]. When including time, we use Tamer [2].

Solution Decoder. It translates π into a robot-compliant mission ρ (line 18) and sends it to the Orchestrator (line 19).

Orchestrator. A Plan Executor runs the mission while

a Monitor tracks the vehicle state. If contingencies or new opportunities arise, we the task, inform the Problem Encoder, and enable online replanning (lines 22-32).

Our planning architecture is hosted onboard to ensure autonomy, real-time responsiveness, and maintain the robot's condition in the unpredictable deep-ocean environment.

VII. Handling Contingencies and Opportunities

Failures, such as instrument faults, require an immediate halt and surfacing of the vehicle. Contingencies or new opportunities, like unexpected battery levels, necessitate pausing the mission to address issues or optimize outcomes. Replanning is needed in both cases. To this aim, our Orchestrator exploits a Finite State Machine (FSM) where each node is a robot mode. For instance, the *auv* mode allows the vehicle to operate autonomously, *rov* enables its teleoperation, *standby* keeps the system ready for instructions, and *all_stop* halts the robot. Nodes can be preemptable, allowing interruptions by observations or commands, or non-preemptable, where an external program takes full control until it finishes, at which point the Orchestrator resumes its transitions. For example, *auv* is preemptable and *emergency_surfacing* can be triggered once this node is stopped. *emergency_surfacing* is non-preemptable and safely ascends the robot to the surface. The edges of the FSM control the robot's modality changes and are of three types: *command*, *trigger*, and *replan*. A *command* is an external input received from a human operator or from the Online (Re)-Planner. For example, *abort_mission* aborts any mission running while in *auv* mode and brings the robot to the *rov* modality. A *trigger* causes transitions based on certain conditions, like *imminent_collision*, which temporarily halts autonomy and moves the robot to a safe location. The *replan* edge cycles in *auv* mode, pausing current tasks, and awaits a new plan from the Online (Re)-Planner before resuming operations.

In our use case, the robot starts in *rov*, entering *standby* until receiving a plan execution request from the Online (Re)Planner, transitioning the FSM to *auv*. Plans may include *check* actions that require feedback from the State Estimator or replanning. In all cases, the FSM remains in the *auv* mode until mission completion or abortion. In low-battery situations, initial replanning attempts to maintain the *auv* mode. If it fails, to avoid the robot to shut down in an unreachable area, two thresholds, α (low battery) and β (critical battery), trigger mode changes: α prompts navigation to the charging station (*go home*), while β triggers *emergency surfacing* to avoid system loss. For imminent collision, the *auv* mode is suspended, switching to *get off my back* mode. If the collision risk persists, the robot enters *emergency surfacing*.

VIII. Experimental Experience

We are engaged in a comprehensive and ongoing testing program, proving our hardware and software contributions.

Hydrone-R has been operating uninterrupted since June 2023 (see Figure 1) at a depth of 325 meters in Norway, aiming for a 10-year subsea residency with occasional scheduled maintenance. The robot has started inspection activities and is closely monitored. Meanwhile, Saipem's commitment to subsea pipe inspection is evident with the Flatfish robot, currently operating at 1700 meters depth off the Brazilian coast. Figure 3 shows Flatfish inspecting a pipeline[4].

Saipem offers a simulated environment based on the Gazebo simulator [16], featuring a Hydrone-R robot (faithful to the real one), pipelines, and inspection clusters. We equipped the robot with the ability to perform the action a = *move(s, g)*, moving from s to g. The action requires the robot to be at s with a suffient battery according to the consumption model. As an effect, *move* brings the robot to g, marks this location as visited, and consumes the battery. Then, we asked the robot to visit a set of targets. In numeric planning, *move* is instantaneous and increases the plan duration d based on the elapsed timed from the comsumption model of the robot. In temporal planning, *move* is a durative action with fix duration d. In both cases, we tested the capability of our planning module to automatically generate valid plans, optimize resources, handle mandatory and optional goals, follow ordering rules, and replan. Details about performed tests and obtained results follow, highlighting the planning time t (s), consumed battery b (%), number n of targets reached, duration d of the plan (s), and cost c of the plan (see Table I). Results are shown for both numeric (left) and temporal (right) planning, with the UP library automatically using ENHSP for numeric and Tamer for temporal planning.

Automatic Plan Generation and Validation. The robot must visit $N = 6$ targets randomly sampled within a 100-meter cube around its initial pose, with enough battery to visit all. No optimization metrics or constraints are applied, and all goals are mandatory. The result is a sequence of *move* whose order depends on whether the problem is numeric or temporal, as seen in Test 0 of Table I (battery consumption: 92% vs. 89%, plan duration: 279.90 s vs. 274.20 s).

Plan and Optimize Resources. We repeat the test with battery optimization. The UP library still does not provide an optimal temporal planner; thus, results are available only for the numeric use case. ENHPS outputs a plan that consumes 86% of the battery instead of 92% (Test 1).

Accept mandatory and optional goals. The robot must visit $N = 8$ random targets, with associated costs $\{t_0: 7, t_1: 6, t_2: 3, t_3: 1, t_4: 5, t_5: 8, t_6: 2, t_7: 4\}$ - for a total cost of 36 if no target is reached. The battery cannot cover all targets, but they are optional and the robot will pay the cost of missed ones. We performed the test with no constraint, a "*while battery > 10 %*" constraint for each goal, and a "*within 180 seconds*" constraint for each goal. Without constraints, ENHSP and Tamer produced different plans but both skipped t_3 (with $c(t_3) = 1$) to minimize penalties (see b and d of Test 2). With the battery constraint, they skipped t_6 to balance cost and battery preservation, saving about 10% more battery compared to the unconstrained test (Test 3). Under the time constraint, both engines planned to visit 4 out of 8 goals (Test 4), with plan durations near the upper limit (175.70 s for numeric and 177.90 s for temporal planning) and the

[4]See `https://www.youtube.com/watch?v=3ni6kLWgvAk`

test	optimal	optional	ordered	$\sqrt[3]{A}$	N	t (s)	b (%)	n ($\leq N$)	d (s)	c
0	-	-	-	100	6	0.68 | 0.16	92 | 89	6 | 6	279.90 | 274.20	-
1	[]	-		100	6	0.74 | -	86 | -	6 | -	268.46 | -	-
2	-	[]	-	100	8	10.27 | 767.52	99 | 97	7 | 7	292.10 | 287.96	1 | 1 (out of 36)
3	-	[while]	-	100	8	2.96 | 221.22	86 | 86	7 | 7	266.70 | 266.70	2 | 2 (out of 36)
4	-	[within]	-	100	8	2.50 | 186.84	55 | 57	4 | 4	175.70 | 177.90	10 | 10 (out of 36)
5	-	-	[]	100	6	0.72 | 0.21	84 | 87	6 | 6	265.20 | 269.51	-
6	-	-	[while]	100	6	0.76 | 51.49	89 | 89	6 | 6	274.30 | 274.56	-
7	-	-	[within]	100	4	0.49 | 33.19	54 | 54	4 | 4	174.10 | 174.30	-

TABLE I

HYDRONE-R VISITS N TARGETS IN A $\sqrt[3]{A}$-SIDE CUBE. WE REQUESTS *optimal* PLANS AND PLANS WITH *optional* OR *ordered* TARGETS. CONSTRAINTS ARE NONE ([]), NUMERIC ([*while battery > 10 %*]), AND TEMPORAL ([*within 180 s*]). RESULTS SHOW PLANNING TIME t (S), CONSUMED BATTERY b (%), NUMBER n OF TARGETS REACHED, PLAN DURATION d (S), AND COST c FOR NUMERIC (LEFT) AND TEMPORAL (RIGHT) PLANNING.

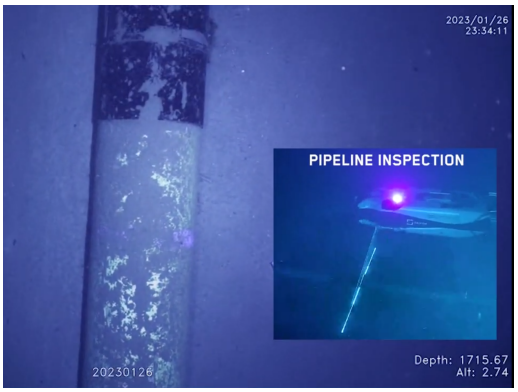

Fig. 3. Flatfish inspecting a pipeline in the Brazilian coast.

lowest possible cost for skipping 4 goals (c = 1+2+3+4).

Accept ordering rules. The robot must visit $N = 6$ ordered and mandatory targets (t_0<t_4, t_1<t_5, t_2<t_3, t_3<t_5, t_4<t_5), reduced to $N = 4$ (t_0<t_3, t_1<t_2, t_2<t_3) under time constraints. ENHSP and Tamer generate valid plans in all cases (Tests 5-7); e.g., with no constraints, ENHSP plans $s \rightarrow t_1 \rightarrow t_2 \rightarrow t_0 \rightarrow t_4 \rightarrow t_3 \rightarrow t_5$ using 84% of the battery, while Tamer plans $s \rightarrow t_1 \rightarrow t_0 \rightarrow t_4 \rightarrow t_2 \rightarrow t_3 \rightarrow t_5$ using 87%.

Replan. The robot must reach t_0 while tracking an ordered set of opportunities $G' = \{t_1, t_2, t_3\}$. *check* actions verify the battery level; if sufficient, the next target in G' is added. Upon reaching t_0, a *check* confirms the battery is 99.34%, above the required 93.63%, so t_1 is added. This continues until all targets in G' are visited; without replanning, only t_0 would have been visited, leaving G' for later.

Manual planning for medium to complex missions previously took 3 to 10 times the mission duration and was prone to errors as complexity increased. Humans struggled to create optimal plans or manage unexpected conditions beyond predefined scenarios. Now, contingencies can be defined and managed in real-time through replanning.

IX. CONCLUSION

In this paper, we introduced Hydrone-R, an AUV designed for autonomous deep-ocean inspection, optimizing resources and minimizing makespan while adapting to unforeseen situations and new opportunities. Our planning module utilizes the open-source Unified Planning library, enhancing decision-making and efficient (re)planning during underwater inspections. We detailed the hardware and software architecture and discussed its practical applications in the offshore energy industry. Our solution advances underwater robotics by enabling *true autonomy* and long-term ocean *residency*.

ACKNOWLEDGMENTS

This work is supported by the AIPlan4EU project (EU Horizon 2020, GA 101016442), the AI@TN project (Autonomous Province of Trento), and the iNEST project (EU Next-GenerationEU, PNRR, mission 4, comp. 2, inv. 1.5, D.D. 1058 23/06/2022, ECS00000043).

REFERENCES

[1] F. Ingrand and M. Ghallab, "Deliberation for autonomous robots: A survey," *Artificial Intelligence*, vol. 247, pp. 10–44, 2017.
[2] A. Valentini, A. Micheli, and A. Cimatti, "Temporal planning with intermediate conditions and effects," in *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 06, 2020, pp. 9975–9982.
[3] K. B. Ånonsen, O. K. Hagen, Ø. Hegrenæs, and P. E. Hagen, "The hugin auv terrain navigation module," in *2013 OCEANS-San Diego*. IEEE, 2013, pp. 1–8.
[4] B. Johansson, J. Siesjö, and M. Furuholmen, "Seaeye sabertooth, a hybrid auv/rov offshore system," in *SPE Offshore Europe Conference and Exhibition*. SPE, 2011, pp. SPE–146 121.
[5] D. Ribas, N. Palomeras, P. Ridao, M. Carreras, and A. Mallios, "Girona 500 auv: From survey to intervention," *IEEE/ASME Transactions on mechatronics*, vol. 17, no. 1, pp. 46–53, 2011.
[6] J. Albiez, S. Joyeux, and M. Hildebrandt, "Adaptive auv mission management in under-informed situations," in *OCEANS 2010 MTS/IEEE SEATTLE*, 2010, pp. 1–10.
[7] S. MahmoudZadeh, D. M.W Powers, K. Sammut, A. Atyabi, and A. Yazdani, "A hierarchal planning framework for auv mission management in a spatiotemporal varying ocean," *Computers & Electrical Engineering*, vol. 67, pp. 741–760, 2018.
[8] M. Cashmore, M. Fox, D. Long, D. Magazzeni, and B. Ridder, "Opportunistic planning in autonomous underwater missions," *IEEE TASE*, vol. 15, no. 2, pp. 519–530, 2018.
[9] Z. Yan, W. Liu, W. Xing, and E. Herrera-Viedma, "A multi-objective mission planning method for auv target search," *Journal of Marine Science and Engineering*, vol. 11, no. 1, 2023.
[10] D. E. Smith, "Choosing objectives in over-subscription planning," in *Int. Conference on Automated Planning and Scheduling*, 2004.
[11] D. M. Gaines, T. Estlin, F. Fisher, C. Chouinard, R. Castano, and R. C. Anderson, *Planning for rover opportunistic science*. Pasadena, CA: Jet Propulsion Laboratory, National Aeronautics and Space ..., 2004.
[12] M. Fox and D. Long, "Pddl2. 1: An extension to pddl for expressing temporal planning domains," *Journal of artificial intelligence research*, vol. 20, pp. 61–124, 2003.
[13] D. E. Smith, J. Frank, and W. Cushing, "The anml language," in *The ICAPS-08 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*, vol. 31, 2008.
[14] G. Francés, M. Ramirez, and Collaborators, "Tarski: An AI planning modeling framework," 2018.
[15] E. Scala, P. Haslum, and S. Thiébaux, "Heuristics for numeric planning via subgoaling," in *Int. Joint Conf. on Artificial Intelligence*, 2016.
[16] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, vol. 3, 2004, pp. 2149–2154.