Situation-Aware Task Planning for Robust AUV Exploration in Extreme Environments

Yaniel Carreno[1,2] [*], Jonatan Scharff Willners[2] [*], Yvan Petillot[1,2], Ronald P. A. Petrick[1]
[1]Edinburgh Centre for Robotics, Edinburgh, United Kingdom
[2]Ocean Systems Laboratory, Heriot-Watt University, Edinburgh, United Kingdom
{y.carreno, j.scharff_willners, y.r.petillot, r.petrick}@hw.ac.uk

Abstract

Achieving an accurate model of the underwater domain that captures all its complex dynamics represents a challenge. Therefore, planning solutions often involve replanning. High-level abstraction of the domain definition required for tractable planning means that the system often ignores specific properties, giving rise to states where survivability, reliability, and mission quality could be compromised. This work proposes an approach that combines planning, knowledge representation and decision making to achieve high-level mission goals that maintain mission survivability and improve robustness. We present the Situational Evaluation and Awareness (SEA) framework to bridge the high-level planning and low-level mission execution systems. SEA maintains a dynamic evaluation of the state to provide a goal completion assessment for local recovery and global mission execution. The approach shows good performance during evaluation in different problems involving real scenarios.

1 Introduction and Motivation

Autonomous Underwater Vehicles (AUVs) are increasingly used to implement complex missions that require robots with a high-level of operational autonomy [Thompson and Guihen, 2019]. AI planning is well situated to solve many real-world problems associated with the deployment of intelligent agents in complex scenarios that usually require long-term operability [Kunze *et al.*, 2018]. In particular, AI planners that reason about temporal and numeric constraints have previously been integrated into real AUV platforms [Maurelli *et al.*, 2016; Buksz *et al.*, 2018] to solve missions involving seabed exploration, inspection and maintenance of offshore structures, emergency response, and military intervention. These solvers generate a *plan*—a structured sequence of actions that guides the initial world state to a goal state—which is model-based and suitable for execution in the world. However, there are drawbacks in using solvers that generate deterministic plans to deal with uncertain domain dynamics since they increase the risk of plan execution failures. A long-standing solution for dealing with plan failures relies on the system's capacity to provide a new plan (after replanning) that maintains the mission implementation.Potential reasons for replanning might include: variations in the mission goal requirements, the failure of specific actions, and the acquisition of new sensing data, all of which directly affect the system's operability.

Figure 1: **Dora**, BlueROV2 operated as an AUV in real applications. The exploration scenario counts with multiple points of interest that require inspection. Dora uses an automated planning system to perform tasks that depend on the AUV sensors and actuators.

In the maritime domain, the maintenance of the AUV's operability is fundamental for achieving mission effectiveness. Operability is conditioned by the system's ability to provide solutions to unexpected situations at planning time, avoiding *dead-ends*—total failure during mission implementation. Robot operability is usually linked to three main components: survivability, reliability, and quality [Thompson and Guihen, 2019]. Due to the complexity of the domain, the maintenance operations of the AUV must consider (i) planning for a *current state*—the actual state that encapsulates the conditions and effects (properties) of currently executing actions—and (ii) failure specifications when they occur. The current state uses the sensing information to update the problem properties. Consider the following example.

Motivation. The AUV (Dora) in Figure 1 navigates around a structure to perform image reconstruction. The initial plan contains multiple instances of the `MAP(WPI,WPF)` action, defining AUV navigation from an initial point `WPI` to a final point `WPF` while mapping the trajectory. Consider the case in which one of these actions fails in the middle of the path. One possible reason for the failure could be poor fea-

[*]Authors have contributed equally in this work.

tures tracking[1] that leads the AUV to build multiple maps of the same structure, which is unwanted. This problem does not alter the execution of the navigation. However, it strikes one of the operability elements (quality) as Dora does not obtain a good outcome from the mapping process. After the failure, the AUV implements replanning to solve the unfinished goals. In this setting, the planning system does not know the robot's actual position. The action failed when the AUV was not at any of the points (`WPI` and `WPF`) included in the abstraction of the real-world, represented by the points, modelling this problem. This makes the planner incapable of generating a plan as knowing the AUV's location is a necessary precondition to implementing the `MAP` action. Assuming the AUV actual location is known, mission completion can also be affected by the bad judgement around the tasks the AUV should complete recovering from the failure. Recovery from features tracking issues requires a plan to complete new goals (local goals) for AUV relocalisation before mapping. Then Dora can keep implementing the mission goals (global goals). Standard replanning finds a new plan from the current state to achieve just global goals. In this case, reasoning about the failure types supports implementing a more effective replanning that might include planning for mission goals and other goals linked to the AUV failure recovery.

This paper proposes a solution to improve mission operability that combines a knowledge-based framework and goal-based mission planning to achieve plans for dynamic problems. This framework is adaptable to changes in the initial model, allowing a less domain-dependent description while keeping the potential of planning with temporal and numeric notions. We present a new approach, called Situational Evaluation and Awareness (SEA), that combines knowledge of mission failure types with planning for current states enhancing robustness over long-term periods while maintaining information gathering and goal execution quality. SEA is a failure solver which acts when failures occur, driving the robot from the failure state by proposing alternative behaviours or updating its knowledge. SEA evaluates the characteristics of failures to complete the global goals while introducing local goals that enable recovery from failure states.

We evaluate our approach using a BlueROV2[2] robot that acts as an AUV in several inspection tasks in a scenario[3] where different failure conditions can occur during mission implementation. Our results show that the combination of SEA and Online Planning outperforms a benchmark approach that plans for static states and does not consider the characteristics of the failures to support effective replanning.

2 Background and Related Work

The planning and robotics communities have a long history of exploring the performance of online mission monitoring [Pettersson, 2005; Levine, 2012] by analysing the characteristics of action failures, the variations in the initial mission goals, and the system's capacity to maintain operation at any cost. One solution to these issues are approaches based on probabilistic models that compute a policy considering the uncertainty in the domain to provide robustness to temporal planning solutions. An example in marine applications is [Duckworth *et al.*, 2021], which handle time-varying problems with unknown dynamics on single missions. Alternative strategies consider run-time contingencies in deterministic models. [Cashmore *et al.*, 2019a] propose the computation of alternative solutions in a Simple Temporal Network [Dechter *et al.*, 1991] plan to maintain a valid solution. The STPUD [Cimatti *et al.*, 2018] framework focuses on solving problems with uncertainty around action's duration. [Valentini *et al.*, 2020] present a technique that allows reasoning regarding time, conditions, and effect changes in the action duration. There is a limitation using these approaches when dealing with failures imply alternative plans with different goal sets. For these situations replanning is advantageous.

The approach of [Cashmore *et al.*, 2019b] shares certain similarities with our work by considering planning techniques for situated replanning. In particular, they focus on searching (in parallel) for a better plan by replanning with a dynamic initial state, modelled as a temporal planning problem with timed initial literals (TILs) [Cresswell and Coddington, 2003]. TILs are also considered in [Cashmore *et al.*, 2018] where they model the solution by reasoning about external events and time. These approaches mainly focus on problems with time constraints. We consider the agent's operability instead of mission time limitations. In our work, we also represent the current state by a temporal planning problem and replan for failures that are not necessarily linked to the action in execution. This allows replanning in advance for situations that affect the plan's execution or its quality when the agent reaches future states, leading to robust solutions.

IxTeTeXEC [Lemai and Ingrand, 2004] is another strategy that considers the advantages of online planning. This framework executes missions with rigorous temporal constraints in dynamic environments by integrating planning, plan repair and execution control. The main differences between plan repair and replanning are presented in [Fox *et al.*, 2006]. Our system integrates replanning and plan repair as valid options depending on the characteristics of the failure, which can induce new requirements (problem goals).

3 System Overview

This section provides an overview of the main elements of our system (see Figure 2) that integrates the SEA framework (see Section 4) and Online Planning (see Section 5). The system contains four modules that are interconnected at different levels with the World: (i) Mission Interface, (ii) Planning Interface, (iii) Execution Interface, and (iv) Robot Interface.

Mission Interface includes the domain and problem that describe the requirements from the operator and the properties of the world we can encapsulate in the model.

Planning Interface includes all the available knowledge at the planning time to generate the problem and plan solution,

[1]Underwater image reconstruction algorithms struggle to maintain a good view of features to perform tracking while mapping. Therefore, approaches often choose new features, making the system build a new map taking as reference the new feature set.

[2]https://bluerobotics.com/store/rov/bluerov2/

[3]An underwater scenario was built in the facilities of the Ocean Systems Laboratory at Heriot-Watt University.

which is parsed and dispatched to the Execution Interface. SEA introduces the Intermediate Mission Requirements component ($\mathcal{IMR}$) extending the initial Mission Requirements ($\mathcal{MR}$) when notifications of failures or substantial changes occur during the plan implementation. SEA receives feedback for the Online Planning and Plan Execution to define the propositions and goals to change (add or remove) to avoid failures. Besides, SEA is connected to the World Model through a Failures Ontology ($\mathcal{FO}$) that defines the possible type of failures associated with the environment. The ontology is used for decision making to deal with unexpected situations. SEA can command the Online Planning to cancel the actual plan execution, acknowledging failures for processing the data arriving from the Execution Interface. This connection is fundamental because failures can occur during action implementation. However, they might not directly affect the action in execution but can affect the mission outcomes when the agent attempts to reach other states later in the plan.

Execution Interface takes the dispatched action by the Planning Interface and translates it to action commands understandable for the AUV. The Execution Interface acts as a bridge providing the sensing information used to determine the quality of plan implementation and helps SEA (in combination with $\mathcal{FO}$) identify the failure source. This interface embeds the low-level algorithms such as those mentioned in Section 4 that allow the performance analysis of individual actions and informs the system about the execution process.

Robot Interface includes the robotic platforms we can use in the mission. This interface provides all necessary perceptual information to evaluate the mission's execution and supports the definition of the application module in the $\mathcal{FO}$.

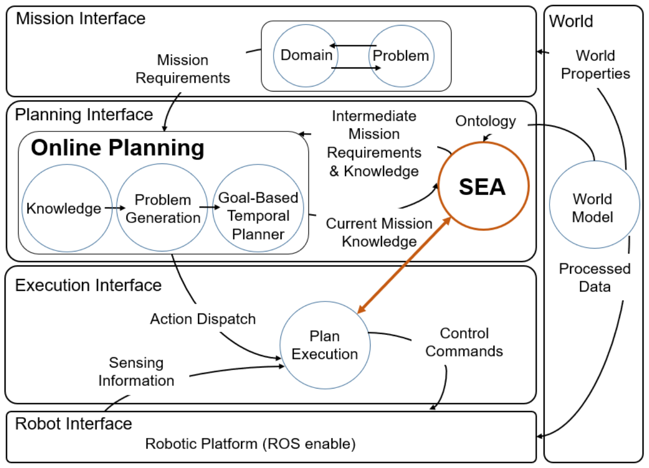

Figure 2: High-level task planning architecture. The SEA provides robustness to the automated planning system dealing with unexpected situations during plan execution.

4 Knowledge-Based Framework

Situation awareness (SA) can be defined as the capacity of the autonomous agent to reason about the changes occurring in the environment during mission execution. This understanding considers the system observations acquired during the mission by processing the sensing data and past experiences that help to create intelligent behaviour. The knowledge framework builds on *concept*, *definitions* and *statements*, and consistently connects to the elements of experience and data acquisition that define the SA. This connection is fundamental for the autonomous agent to decide its response to a particular mission state. Mission knowledge encloses a large set of possible elements which are used by AI planning algorithms to generate rational behaviours. We focus on the elements of knowledge that can support the SA over the mission: robot capabilities and possible failure types by considering the characteristics of the environment.

We have created a library of ontologies called Failures Ontology ($\mathcal{FO}$) that combines the autonomous agent's capabilities and the type of failures associated with different worlds (for the scope of this paper: underwater domain) that provide knowledge to the system which is used to deal with the environment changes. This library contains three modules: foundation, domain, and application. The first defines generic concepts and properties. The second provides space to map and integrate data from different sources (e.g., robot types, hardware properties, specific software, failure types and risk levels, mission requirements, world properties, etc.). The third integrates the source data (e.g., mission actions, agent resources, action execution properties, etc.). $\mathcal{FO}$ enables the system to identify actions that can be used to overcome different failure types. The knowledge framework structure makes the approach scalable to multiple failure types targeting other realistic and uncertain real-world scenarios e.g., robotic manipulation and self driving cars. The library is extended considering previous works [Pettersson, 2005; Levine, 2012] regarding tolerance analysis, anomaly and fault detection to define the set of possible failure reasons. This tool provides updated advice to the mission-goal system (see Section 5) about possible behaviours to deal with failures while maintaining mission survivability and utility. SEA creates a bridge between plan reasoning and execution to support system robustness when running for long periods of time.

Algorithm 1 shows the reasoning used by SEA. The system takes the initial set of mission goals as a reference (line 2) and performs global goals execution checks until all mission goals are completed (line 3). SEA acts independently from mission planning providing knowledge updates (γ) and intermediate mission requirements ($\mathcal{IMR}$). SEA keeps checking the processed sensing data $\mathcal{S_I}$ to ensure that action execution is working well (line 4). If a notification is received, SEA classifies the information (line 6) using $\mathcal{FO}$ and $\mathcal{N}$. Then, it generates recovery goals depending on the type of notification (line 7). The system acts as a finite state machine (FSM) to define the set of goals depending on the failure characteristics. For some particular cases, the generation of recovery goals might require communication with low-level algorithms to support the goal's generation. This is the case we describe in the motivation of this paper: when visual feature tracking for localisation is lost, the SEA needs to deal with this situation by connecting with a low-level algorithm that provides new points that enable the system to merge with the original map to improve image reconstruction. When new goals are obtained, γ and $\mathcal{IMR}$ (line 8-9) are updated and returned. Notice $\mathcal{IMR}$ contains, in this case, a new set of goals (recovery goals) that are not the original ones; therefore, the set of mission goals to achieve (i) does not change. SEA also checks if there are failures associated with the implementation of the action that is independent of external factors (e.g., the action was not completed in the predefined time causing the failure) (line 11). In this case, γ and $\mathcal{IMR}$ are updated and returned (line 13-14). However, no additional actions are introduced to the plan (a plan repair is introduced to deal with the problem). If none of the above situations occur, SEA updates the knowledge (line 16-17). As a result, SEA attempts to maintain mission survivability and adaptability over long-term missions: the evaluator forces the generation of recovery plans which are introduced in the execution of the original plan to solve particular failures when they occur.

Low-level Algorithm: Robust Localisation. To aid in robust localisation, a viewpoint planning module is used to connect simultaneous localisation and mapping (SLAM) system with the high-level planner. Here, we describe an example of the low-level algorithms interacting with SEA that are embedded in the Execution Interface. This planner generates viewpoints based on prior poses (keyframes) in the SLAM's pose-graph, and the current knowledge about the environment. In this paper, we use the SLAM system described in [Xu *et al.*, 2021] which integrates sensor data from dead reckoning based on a doppler velocity log (DVL), an inertial measurement unit (IMU) and a depth sensor. When visual feature tracking is lost, SLAM notifies the viewpoint planner along with a set of keyframes. When a lost tracking notification is received, the current pose estimation is evaluated to determine if it is supposed to have features to track in the previously mapped environment or not. If it is considered to have too few previously seen features to perform reliable pose estimation (e.g., traveling through an open featureless space or turning away from a structure) the mission can progress without interruption. However, if it is determined that the robot should have enough features in view to perform a reliable pose estimation, the planner will enter a relocalisation mode. Based on the keyframes, a sample-based approach is applied to find viewpoints deemed to have a high probabilistic likelihood of reobserving features in the map to relocalise. For each keyframe *n* viewpoints are sampled within a specified distance from the keyframe. The sampled viewpoints are evaluated through a utility function, and for each keyframe, the highest scoring viewpoint is selected as a new waypoint for a temporal plan for relocalisation. The utility function is a modified version of the one presented by [Palomeras *et al.*, 2019], and is described in detail in [Scharff Willners *et al.*, 2021]. The utility function consists of four weighted reward functions: i) The amount of observed voxels based on a simulated sensor in the mapped environment, represented as an OctoMap [Hornung *et al.*, 2013]; ii) The distance from the keyframe; iii) The distance to the closest observed voxel in the simulated sensor. If a voxel is too close it can increase the risk of collision and if too far away it might not be visible due to inadequate water conditions. iv) If the viewpoint is in a configuration classified as free in the OctoMap.

The generated set of viewpoints is sent to SEA, who uses them to advise the system to build a (new) plan for AUV relocalisation using the viewpoints as local goals. This is an attempt to force map-merging (loop closure). Merging sub-maps to the original map has multiple benefits, including more reliable pose estimation and a better reconstruction as all the merged maps are connected through visual features.

Algorithm 1: Situational Evaluation and Awareness

```
Input: $\mathcal{S_I}$: Sensing Information.
Input: $\mathcal{FO}$: Failures Ontology.
Input: $\mathcal{MR}$: Global Mission Requirements.
Input: $\mathcal{MK}$: Mission Knowledge.
Input: $\mathcal{AU}$: Action Update.
Output: $\mathcal{IMR}$: Intermediate Mission Requirements.
Output: $\gamma$: Current Mission Knowledge.
1  begin
2      $i \leftarrow$ ExtractGlobalMGoals($\mathcal{MR}$)
3      while not $i \leftarrow \emptyset$ do
4          $\mathcal{N} \leftarrow$ CheckFNotifications($\mathcal{S_I}$)
5          if $\mathcal{N}$ then
6              $\delta \leftarrow$ ClassifyNotification($\mathcal{FO}, \mathcal{N}$)
7              $\psi \leftarrow$ GenerateRecoveryG($\delta, \mathcal{MK}$)
8              $\gamma$.UpdateKnowledge($\mathcal{S_I}, \mathcal{MK}, \psi$)
9              $\mathcal{IMR}$.NotifyPlanning($\psi, \gamma$)
10         else
11             $u \leftarrow$ CheckActionFailure($\mathcal{AU}$)
12             if $u$ then
13                 $\gamma$.UpdateKnowledge($\mathcal{S_I}$, $\mathcal{MK}$, $u$)
14                 $\mathcal{IMR}$.NotifyPlanning($\gamma$, $u$)
15             else
16                 $\gamma$.UpdateKnowledge($\mathcal{S_I}$, $\mathcal{MK}$)
17                 $\mathcal{IMR}$.NotifyPlanning(null)
18         return ($\mathcal{IMR}, \gamma$)
```

5 Planning for System Autonomy

Embedded decision-making systems that reason about mission knowledge, including failures, and planning algorithms can optimise the long-term management of robotic platforms and provide fast, dynamic responses to events by autonomously coupling global mission requirements and local recovery mechanisms. This section describes the methods for mission plan generation.

High-Level Planning. According to the robot's capabilities, AI temporal planning solves a temporal planning problem (see Definition 1), guiding the autonomous agent to implement a sequence of actions called a temporal plan (see Definition 2), leading the system from an initial state to a final state that achieves the mission goals. The plan actions can be instantaneous (see Definition 3) or durative (see Definition 4). We adopt the full Planning Domain Definition Language (PDDL) version 2.1 [Fox and Long, 2003] with continuous change considering propositional temporal planning problems with TILs.

Definition 1. *A temporal planning problem is a tuple* $\mathcal{P}_t := \langle \mathcal{P}, \mathcal{A}, \mathcal{I}, \mathcal{G}, \mathcal{T} \rangle$*, where* $\mathcal{P}$ *is set of propositions;* $\mathcal{A}$ *is* *a set of instantaneous and durative actions, where the duration of the actions is controllable and known; $\mathcal{I}$ is the complete function defining the initial state of propositions, $\mathcal{I} : \mathcal{P} \rightarrow \{\top, \bot\} \cup \mathbb{R}$, where $\top$ and $\bot$ denote the defined and undefined values, respectively; $\mathcal{G}$ is a set of goals, where $\mathcal{G} : \mathcal{P} \rightarrow \{\top, \bot\} \cup \mathbb{R}$ is a (possibly partial) function that describes the goal conditions; and $\mathcal{T}$ is a set of time windows defined using TILs.*

Definition 2. *A temporal plan Π_t for a temporal planning problem $\mathcal{P}_t$ is a set of tuples $\pi_t := \langle a, t, d \rangle$, where $a \in \mathcal{A}$, t is the action starting time and d and the action's duration. A durative action a hold $t \in \mathbb{R}_{\geq 0}$ and $d \in \mathbb{R}_{>0}$, where $\mathbb{R}_{\geq 0} = \{x \in \mathbb{R} \,|\, x \geq 0\}$ and $\mathbb{R}_{>0} = \{x \in \mathbb{R} \,|\, x > 0\}$.*

Definition 3. *An instantaneous action a_i, where $a_i \in \mathcal{A}$, is a tuple $\langle a_{pre}, a_{eff} \rangle$, a_{pre} is a set of pre-conditions that must hold for action's application, a_{eff} is the set of action effects.*

Definition 4. *A durative action a_d, where $a_d \in \mathcal{A}$, is a tuple $\langle a_{pre}, a_{eff}, a_{dur} \rangle$; a_{pre} is a set of conditions that must hold for the action to be applicable; a_{eff} is the set of action effects; and a_{dur} is a set of duration constraints.*

Several approaches exist in the AI literature for solving temporal planning problems [Coles *et al.*, 2010; Benton *et al.*, 2012]. Our planning system models the problem by defining types, fluents, functions, durative actions and different search metrics and heuristics. For instance, our domain[4] defines the types such as `robot`, `sensor`, and `waypoint`. Typing allows the resources to be classified in different sorts. These sorts are linked to the system knowledge. Action parameters are associated to these sorts, e.g., for `navigation`, `?r - robot ?wpi ?wpf - waypoint`. Fluents allow the modification of numerical or data resources (e.g., `(assign (battery_level ?r) 100)`). Functions are linked to the observed knowledge and provide numerical values to certain properties of the available resources (e.g., `(distance ?wpi ?wpf - waypoint)`, `(speed ?r - robot)`, etc.). Actions have a duration cost function d, which can be represented by fixed values or to the function value of their resource parameters.

The planner generates a sequence of actions to meet a set of propositional preconditions and then applies scheduling techniques to arrange this action sequence, considering the numeric and temporal constraints. For the durative action `navigation` the preconditions for its implementation requires knowing the robot's actual position `(at ?r - robot ?wpi - waypoint)`, its availability `(available ?r - robot)` and the true connectivity between the initial and the final point where the AUV needs to navigate `(connected ?wpi ?wpf - waypoint)`. The effects of implementing this action are the robot's position changes to `?wpf` and the new waypoint inspected (`inspected ?wpf - waypoint`). In addition, we can define a set of properties related to the robot capabilities such as proposition `(camera_equipped ?r - robot ?s - sensor)` which

[4]In https://github.com/YanielCarreno/domain-ijcai-r2aw.git, we present the domain and problems. Our domain includes actions for navigation, communication, control of the robot's light system, robot recovery, battery recharge, and hardware checks.

```
  Time:      (Action Name)                                [Duration]
 0.000:      (navigation dora wp0 wp10)                    [10.000]
10.001:      (turnon-light dora  wp10)                      [2.000]
12.002:      (navigation dora wp10 wp11)                   [10.000]
(...)
72.007:      (navigation dora wp40 wp41)                   [10.000]
82.008:      (turnoff-light dora  wp41)                     [2.000]
84.009:      (recover dora  wp41)                           [1.000]
```

Figure 3: A fragment of an original temporal plan solution for the image reconstruction of a structure.

is a precondition in actions that require a camera. The mission plan is generated following a cost function f defined by the operator. For temporal planning the default f attempts to minimise the *makespan*—the time required to implement a plan. Therefore, the new state k from a set of possible candidates $\mathcal{X}$ (see Equation 1) is obtained by evaluating $\mathcal{X}$ (see Equation 2), where γ is the progression state transition function and s_i is the state.

$$\mathcal{X} = \bigcup_{a \in exec^{-1}(s_i)} \langle \gamma^{-1}(s_i, a) \,|\, a \in \mathcal{A} \rangle, \tag{1}$$

$$s(\mathcal{X}) = arg_{min}\{f(k) \,|\, k \in \mathcal{X}\}, \tag{2}$$

Figure 3 shows an example of a plan solution for the image reconstruction of a structure using the AUV (BlueROV2). The mission goals are to inspect a set of points[5] around a structure to obtain its image reconstruction. The problem goals are `(inspected wp10) ... (inspected wp41)`. One of the advantages of temporal task planners is that we obtain plans that reach goals associated with different types of actions while controlling the mission duration. For instance, the actions to turn the AUV lights and recovery support the implementation of goals (`light_on wp10`), (`light_off wp41`) and (`recovered wp41`), which are not related to the reconstruction; however they represent another set of mission requirements defined by the operator.

Online Planning. Temporal planning solutions have proved their effectiveness while implementing AUV missions. However, such planners focus on deterministic planning models with predictable outcomes and completely known initial states. These characteristics limit the applicability of temporal planners to solve missions in the underwater domain that require algorithms reasoning about uncertainty considering the environment is highly dynamic. In this work, we acknowledge the state's dynamic by combining temporal planning and the knowledge provided in real-time by the SEA framework, which allows the implementation of an online planning strategy. This procedure provides robustness to the approach by dealing with the state changes associated (in some cases) with failures of different nature (e.g., actuators, sensors, exogenous events, etc.). Algorithm 2 shows the Online Planning approach, which extracts the mission goals (line 2) and introduces the initial state with the requirements $\mathcal{MR}$ (line 3). The online planning algorithm is active until all mission goals are completed (line 4). It takes the current knowledge to generate a planning problem (line 5), which is used to generate a plan (line 6). The algorithm cancels any plan

[5]We have the notion of the initial robot and structure coordinates.

Algorithm 2: Online Planning

```
Input: $\mathcal{MR}$: Global Mission Requirements.
Input: $\mathcal{IMR}$: Intermediate Mission Requirements.
Input: $\gamma$: Current Knowledge.
Output: $\mathcal{MK}$: Mission Knowledge.
1  begin
2      $i \leftarrow$ ExtractGlobalMGoals($\mathcal{MR}$)
3      $\mathcal{IMR} \leftarrow \mathcal{MR}$
4      while not $i \leftarrow \emptyset$ do
5          $\mathcal{P} \leftarrow$ GenerateProblem($\mathcal{IMR}$)
6          $\Pi_t \leftarrow$ GeneratePlan($\mathcal{P}$)
7          $\mathcal{F}_{i-1}$.CancelCurrentPlan()
8          $\mathcal{F}_i$.DispatchPlan($\Pi_t$)
9          p.CheckSEA($\mathcal{IMR}, \gamma$)
10         if $p$ Failed then
11             $i$.UpdateGoals($\mathcal{IMR}, \gamma$)
12             $\mathcal{MK}$.UpdateKnowledge($\mathcal{IMR}, \gamma$)
13             return $\mathcal{MK}$
14         else if $\mathcal{F} \leftarrow$ isAchieved then
15             $i \leftarrow \emptyset$
```

Run-Time Plan (1)	**Description**
`(navigation dora wp0 wp10)`	Navigate to `wp10`
`(turnon-light dora wp10)`	Turn-On light at `wp10`
`(map dora slam wp10 wp11)`	Mapping `wp10` to `wp11`
SEA Notification	Dora Localisation Issue
Run-Time Plan (2)	**Description**
`(map dora slam wpra0 wpr0)`	Localise `dora`
`(map dora slam wpr0 wpr1)`	Localise `dora`
`(map dora slam wpr1 wpr2)`	Localise `dora`
SEA Notification	Dora Localised
Run-Time Plan (3)	**Description**
`(map dora slam wpra1 wp11)`	Mapping `wpra` to `wp11`
`(map dora slam wp11 wp12)`	Mapping `wp11` to `wp12`
`(map dora slam wp12 wp13)`	Mapping `wp12` to `wp13`
`(map dora slam wp13 wp14)`	Mapping `wp13` to `wp14`
SEA Notification	Dora Battery Issue
Run-Time Plan (4)	**Description**
`(navigation dora wpra2 wps0)`	Navigate to surface
`(communication dora wps0)`	Communicate with base
`(recharge dora wps0)`	Recharge battery
`(map dora slam wps0 wp14)`	Mapping `wps0` to `wp14`
`(...)`	
`(turnoff-light dora wp41)`	Turn-Off light at `wp41`
`(recover dora wp41)`	Recover `dora` at `wp41`

Table 1: A run-time plan implementation. The original temporal plan and the recovery (and repair) plans introduced.

in execution (line 7) before dispatching the new plan (line 8). The method keeps checking the updates from SEA (line 9) during the whole mission. If SEA notifies of substantial changes and advises replanning (line 10), $\mathcal{MK}$ and i is updated (line 11-13), returning the actual $\mathcal{MK}$. If SEA does not notifies failures during the plan execution, the system maintains the execution of the plan until the goal state is achieved (line 14). Then the set of mission goals is empty and Online Planing stops (line 15). Notice that the strategy generates a substitute plan in parallel to execution to avoid the issues associated with planning delays in non-quiescent environments. The recovery strategy considers the number of times the agent is visiting the same state. Then after a set of attempts (or recovery plans that includes the same state) if the goal is not achieved it is cancelled from $\mathcal{MR}$ and $\mathcal{IMR}$. Using this policy the system avoids *cyclic redundancy*—the system replans for the same actual state to achieve the same global or recovery goal—during plan execution. An example of cyclic redundancy can occur when strong sea currents impede the AUV to reach a point `WPF`. In that case, after multiple attempts the system removes the goal(s) associated with `WPF` from the list of unfinished goals.

Unexpected Failures. This work presents a model-based approach where SEA attempts to meet the model inaccuracies when dealing with failures in a dynamic environment. Therefore, the system behaviour when failures occur depends on the failures listed in the $\mathcal{FO}$ and the system capacity to find a recovery solution for every possible failure (including those not listed in the $\mathcal{FO}$). In cases the failure is listed the system advises the Online Planner considering the information in the knowledge-based framework. When multiple (known) failures coinciding, the approach considers the level of risk of each failure and decides on the replanning strategy based on the failure ranking, e.g., if there is a high-risk failure, SEA advises a replanning solution for high-risk situations. Finally, when not listed failures occur SEA advises a backup plan involving communication with human operators to evaluate the situation.

6 System Integration Example

Table 1 shows an example of a solution where the execution of the original plan (see Figure 3) is interrupted, as a result, SEA receives a notification that the AUV has lost its localisation when executing an action (action in red). In this case, SEA identifies the type of failure and reasons for the alternative plan that makes the robot recover. The system takes the best possible course of recovery provided by the low-level algorithm for viewpoint generation, and sends a set of new goals and knowledge updates to the Planning System that allows the generation of a new (substitute) plan (recovery-plan). The updates in the $\mathcal{IMR}$ ensure the information used for planning, such as the waypoints connected, meet with the robot and environment kinematics and dynamics, making the new plan achievable. The new plan forces the robot to navigate these points to relocalise. Suppose during the execution of the recovery plan the AUV achieves relocalisation. In this case, the SEA determines that the recovery plan is completed, even when all actions are not achieved. The system will then replicate the uncompleted mission goals and send these new requirements to the system to obtain a plan that solves the remaining goals. Suppose the AUV completes the recovery plan (intermediate), and relocalisation is not achieved. In this case, the system will ask for new relocalisation points to maintain the robot exploring the area looking for the map merging before executing the incomplete goals in the $\mathcal{MR}$. However, the system is designed to avoid situations where an infinite loop can occur as a result of experiencing the same failure multiple times (see Section 5).

Our approach can deal with other types of failures such as battery level notifications (see Table 1). In this case, the system advises the online planning module to implement a plan repair that forces the robot to recharge. The current knowledge updates the proposition associated with the battery level. The new value is under the best (safe) energy range for the robot. Therefore, the actual plan under execution is not valid, and it is replaced with a new plan obtained by updating the initial state with the real battery level. This can happen when the initial model has a low fidelity. SEA helps to avoid situations where model inaccuracy leads to poorly built plans.

7 Evaluation

We implement multiple missions associated to our robot capabilities, including the autonomous inspection of a structure (see Figure 1). Our BlueROV2 is a full ROS-enabled [Quigley *et al.*, 2009] platform with a stereo-vision system designed for close-range 3D inspections. The vision system was developed to satisfy working in murky water [Łuczyński *et al.*, 2019]. This section evaluates the system robustness and the quality of the plan implementation. Our system uses OPTIC [Benton *et al.*, 2012] for plan generation and the architecture builds on ROSPlan [Cashmore *et al.*, 2015]. However, we introduce a set of new elements to support integrating the dynamic replanning and SEA components. The task planning approach finds a plan that leads the robot to explore multiple points of interest.

Experiment 1 (Robustness). This experiment focuses on the inspection of the structure and proposes to assess the capability of our approach to maintain a single consistent map during autonomous operations. We compare the autonomous relocalisation framework presented in this paper with two independent runs of an autonomous waypoint controller. Both approaches have the same set of waypoints to explore. The waypoints construct a vertical lawnmower pattern consisting of 24 waypoints around the structure. In all inspections, we forced the SLAM to lose feature tracking by replacing the images from the cameras with featureless black images.

Experiment 2 (Plan Execution). This experiment evaluates mission implementation quality in a real scenario when the AUV builds the map of a structure moving around a set of defined points (36 waypoints). Evaluation considers three scenarios: (S1) static state, (S2) current state with (partial) SEA, and (S3) current state with SEA. For S1 the initial state is the one reached when action on execution finishes. S2 considers current state updates without $\mathcal{FO}$ (partial SEA). The system considers the planning problem where the initial state is the current state, and there is no need to wait until the action finishes to replan if this is required. Here, there are no notifications associated with the ontology. However, the system can receive updates from the current knowledge and action execution. For S3 the system considers a current state and receives failure notifications from SEA. The experiment consists of 10 problems, each containing failure notifications of different types: (i) high-risk (e.g., water leaking and battery level) and (ii) standard (e.g., localisation). The time for failure notification is the same for all scenarios and are randomised for each problem. We evaluate the mission survivability (MS), mission data quality (MQ), and the execution time (ET) for these problems. The AUV needs to navigate to the surface and communicate with the mainland to implement the recharge.

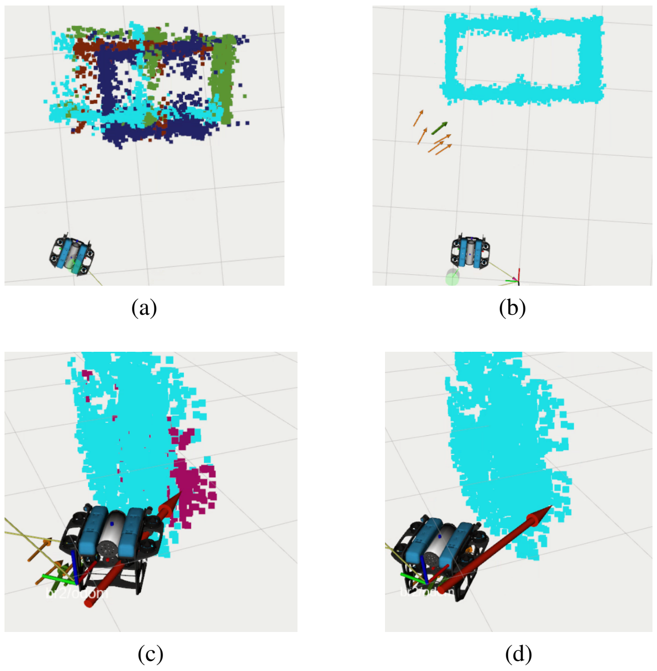

Figure 4: In **(a)**, there is no active relocalisation procedure leading to multiple sub-maps. In **(b)**, the approach (SEA-Online Planning) uses the visual feature tracking information to relocalise, resulting in a single map. In **(c)**, the vehicle lost track around a corner, creating a new sub-map using the new features. **(d)** shows map merging considering the approach generates a recovery temporal plan to deal with multiple map generation. After solving the issue, the AUV keeps the implementation of the originally defined mission goals.

Experiment 3 (Planning Process). This experiment compares the real planning times required to generate a new plan solution combining Online Planning and SEA (S3) with the replanning process using a static state (S1).

8 Results and Discussion

Our domain and problems are encoded in PDDL. All experiments in this section are run on Ubuntu 16.04, with an Intel Core i7-8700, limiting the planner to 5 sec of CPU@3.2GHz, 16GB of RAM. The planning time[6] reference value considers the dynamics of the maritime domain.

The results[7] of implementing Experiment 1 (see Figure 4) illustrate the map at the end of the inspection for each trial. While the autonomous inspection without our relocalisation generates 4 different sub-maps (indicated in different colours in Figure 4a), our approach ends with a single consistent

[6]Time required to generate a plan, times for problem generation and plan parsing are excluded.

[7]System demonstration: https://youtu.be/lPetVtFVe0M

		MS			MQ			ET		
Problem	Notifications	S1	S2	S3	S1	S2	S3	S1	S2	S3
1	(1, 1, –)	0.5	1	1	1	1	1	–	768.3	768.3
2	(1, 2, –)	–	1	1	–	1	1	–	792.3	792.3
3	(2, 2, –)	0.25	1	1	1	1	1	–	814.1	814.1
4	(5, –, –)	–	1	1	–	1	1	–	834.2	834.2
5	(2, 3, –)	0.2	1	1	1	1	1	–	866.3	866.3
6	(–, 3, 2)	–	1	1	–	0.6	1	–	878.3	1032.7
7	(–, –, 5)	1	1	1	–	–	1	867.2	867.2	1234.7
8	(3, 3, 3)	–	1	1	–	0.66	1	–	989.3	1512.2
9	(5, 2, 3)	–	1	1	–	0.7	1	–	1303.1	1684.3
10	(4, 5, 5)	–	1	1	–	0.64	1	–	1349.2	1960.0

Table 2: Experiment 2:The mission survivability (MS), mission data quality (MQ) normalised, and the execution time (ET) in sec for S1, S2, and S3 considering the Notifications are presented in the following order (battery, water leaking, localisation).

Problem	Notifications	S1	S3
1	(1, 1, –)	0.4	0.04
2	(1, 2, –)	0.43	0.04
3	(2, 2, –)	0.6	0.04
4	(5, –, –)	1.2	0.05
5	(2, 3, –)	2.1	0.08
6	(–, 3, 2)	2.5	0.08
7	(–, –, 5)	2.3	0.09
8	(3, 3, 3)	2	0.1
9	(5, 2, 3)	3.2	0.1
10	(4, 5, 5)	3.1	0.2

Table 3: Experiment 3: Times (sec) taken to synthesise the plan solution for S1 and S3 .

map of the structure (see Figure 4b). These results demonstrate the system can autonomously cope with feature tracking loss by triggering relocalisation plans that allow merging sub-maps. Experiment 1 demonstrates the capacity of the system to maintain the quality of the reconstruction by dealing with subsets of recovery goals while implementing the initial goal requirements. One of the system's advantages can be seen in Figure 4c-4d, while moving around the corner of the structure, the vehicle naturally loses track of the features. When this happens, our system generates a new map (magenta coloured map in Figure 4c), and the SLAM notifies the planner that it has been lost along with keyframes from the previous (original) map seen in blue. A set of viewpoints that should help the AUV to relocalise in the prior map is generated for the high-level planner to include in its temporal plan. The AUV executes the plan to relocalise, leading to a merge of the two maps and being able to proceed around the corner without losing track, as the map now contains features to track on both sides of the corner.

Results of Experiment 2 (see Table 2) demonstrate our approach (represented in S3) outperforms the other two options. The poorest results are achieved for Scenario 1, where replanning needs to wait until the action in execution finishes. The robot manages to survive if the replanning is forced in a location very close to the surface. Analysing survivability, the introduction of the current state in our approach enhances robot performance, which manages to recover and implement the rest of the goals. However, the quality of the plan implementation is compromised for S2 when the SEA notifications detect poor image reconstruction, leading to multiple maps at the end of the mission (see Experiment 1). The AUV manages to implement all the goals, but there are problems with plan quality since SEA does not provide feedback to the Online Planning to deal with this situation. In terms of execution time, S3 requires more time to complete the whole mission considering SEA's advice which makes the system implement a set of goals associated with relocalisation and, therefore, mission goals completion is delayed. However, we recognise the importance of acquiring good data even when that increases the execution time.

The results of Experiment 3 are presented in Table 3. Here, we attempt to analyse the advantages of using Online Planning when the system needs to deal with failures, and the scheme considers the current state. For the static state, we execute replanning manually, and we analyse the average planning time required to obtain a new plan that the system can use. For this experiment, we understand the planning time that considers problem generation, planning and plan parsing. Results show our approach manages to outperform standard solutions considering the new problem, and plan generation in the presence of failures is introduced dynamically in our system. The AUV does not need to wait for new plan construction, making the system more robust against changes.

9 Conclusions and Future Work

This work proposes an approach that combines planning, knowledge representation and decision making to achieve high-level mission goals that maintain mission survivability and improve robustness. We propose the SEA framework to bridge the high-level planning and low-level mission execution systems. SEA maintains a dynamic evaluation of the state to provide a goal completion assessment for local recovery and global mission completion by introducing an Online Planning System that generates plan solutions for the current state. Combining the Online Planning and SEA enhances the AUV performance, improving mission survivability and quality while keeping the advantages of using temporal planning to generate the mission plan. Furthermore, SEA provides a connection with intelligent low-level algorithms that support the SEA decision-making process when it evaluates possible failures and proposes alternative solutions to the planning system to deal with the environment's dynamics. Experimental results in real missions show that the system's capacity to deal successfully with a variety of dynamics changes.

In the future, we plan to extend the framework with an analytical procedure to receive new goals from the operator during the mission. Furthermore, we aim to extend the solution to more failure types and multiple robots, establishing different coordination levels that might help deal with the model inexactitudes more optimally. We also plan to evaluate the system in the open sea while executing long-term missions.

Acknowledgements

This work was funded and supported by the ORCA Hub (`orcahub.org`), under EPSRC grant EP/R026173/1.

References

[Benton *et al.*, 2012] J Benton, Amanda Jane Coles, and Andrew Coles. Temporal planning with preferences and time-dependent continuous costs. In *ICAPS*, 2012.

[Buksz *et al.*, 2018] Dorian Buksz, Michael Cashmore, Benjamin Krarup, Daniele Magazzeni, and Bram Ridder. Strategic-tactical planning for autonomous underwater vehicles over long horizons. In *IEEE/RSJ IROS*, pages 3565–3572. IEEE, 2018.

[Cashmore *et al.*, 2015] Michael Cashmore, Maria Fox, Derek Long, Daniele Magazzeni, Bram Ridder, Arnau Carrera, Narcis Palomeras, Natalia Hurtos, and Marc Carreras. ROSPlan: Planning in the Robot Operating System. In *ICAPS*, 2015.

[Cashmore *et al.*, 2018] Michael Cashmore, Andrew Coles, Bence Cserna, Erez Karpas, Daniele Magazzeni, and Wheeler Ruml. Temporal planning while the clock ticks. In *ICAPS*, 2018.

[Cashmore *et al.*, 2019a] Michael Cashmore, Alessandro Cimatti, Daniele Magazzeni, Andrea Micheli, and Parisa Zehtabi. Robustness envelopes for temporal plans. In *AAAI*, pages 7538–7545, 2019.

[Cashmore *et al.*, 2019b] Michael Cashmore, Andrew Coles, Bence Cserna, Erez Karpas, Daniele Magazzeni, and Wheeler Ruml. Replanning for situated robots. In *ICAPS*, pages 665–673, 2019.

[Cimatti *et al.*, 2018] Alessandro Cimatti, Minh Do, Andrea Micheli, Marco Roveri, and David E Smith. Strong temporal planning with uncontrollable durations. *Artificial Intelligence*, 256:1–34, 2018.

[Coles *et al.*, 2010] Amanda Jane Coles, Andrew Coles, Maria Fox, and Derek Long. Forward-chaining partial-order planning. In *ICAPS*, pages 42–49, 2010.

[Cresswell and Coddington, 2003] Stephen Cresswell and Alexandra Coddington. Planning with timed literals and deadlines. In *Workshop of the UK PlanSIG*, pages 23–35, 2003.

[Dechter *et al.*, 1991] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial intelligence*, 49(1-3):61–95, 1991.

[Duckworth *et al.*, 2021] Paul Duckworth, Bruno Lacerda, and Nick Hawes. Time-bounded mission planning in time-varying domains with semi-mdps and gaussian processes. *Journal of Machine Learning Research*, 2021.

[Fox and Long, 2003] Maria Fox and Derek Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *JAIR*, 20:61–124, 2003.

[Fox *et al.*, 2006] Maria Fox, Alfonso Gerevini, Derek Long, and Ivan Serina. Plan stability: Replanning versus plan repair. In *ICAPS*, pages 212–221, 2006.

[Hornung *et al.*, 2013] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013.

[Kunze *et al.*, 2018] Lars Kunze, Nick Hawes, Tom Duckett, Marc Hanheide, and Tomáš Krajník. Artificial intelligence for long-term robot autonomy: A survey. *IEEE Robotics and Automation Letters*, 3(4):4023–4030, 2018.

[Lemai and Ingrand, 2004] Solange Lemai and Félix Ingrand. Interleaving temporal planning and execution in robotics domains. In *AAAI*, pages 617–622, 2004.

[Levine, 2012] Steven James Levine. *Monitoring the execution of temporal plans for robotic systems*. PhD thesis, Massachusetts Institute of Technology, 2012.

[Łuczyński *et al.*, 2019] Tomasz Łuczyński, Piotr Łuczyński, Lukas Pehle, Manfred Wirsum, and Andreas Birk. Model based design of a stereo vision system for intelligent deep-sea operations. *Measurement: Journal of the International Measurement Confederation*, 144:298–310, 2019.

[Maurelli *et al.*, 2016] Francesco Maurelli, Marc Carreras, Joaquim Salvi, David Lane, Kostas Kyriakopoulos, George Karras, Maria Fox, Derek Long, Petar Kormushev, and Darwin Caldwell. the pandora project: A success story in auv autonomy. In *IEEE OCEANS – Shanghai*, 2016.

[Palomeras *et al.*, 2019] Narcis Palomeras, Natalia Hurtos, Eduard Vidal, and Marc Carreras. Autonomous exploration of complex underwater environments using a probabilistic next-best-view planner. *IEEE Robotics and Automation Letters*, 4(2):1619–1625, 2019.

[Pettersson, 2005] Ola Pettersson. Execution monitoring in robotics: A survey. *Robotics and Autonomous Systems*, 53(2):73–88, 2005.

[Quigley *et al.*, 2009] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009.

[Scharff Willners *et al.*, 2021] Jonatan Scharff Willners, Yaniel Carreno, Shida Xu, Tomasz Łuczyński, Sean Katagiri, Joshua Roe, Èric Pairet, Yvan Petillot, and Sen Wang. Robust underwater slam using autonomous relocalisation. *IFAC Conference on Control Applications in Marine Systems*, 2021.

[Thompson and Guihen, 2019] Fletcher Thompson and Damien Guihen. Review of mission planning for autonomous marine vehicle fleets. *Journal of Field Robotics*, 36(2):333–354, 2019.

[Valentini *et al.*, 2020] Alessandro Valentini, Andrea Micheli, and Alessandro Cimatti. Temporal planning with intermediate conditions and effects. In *AAAI*, pages 9975–9982, 2020.

[Xu *et al.*, 2021] Shida Xu, Tomasz Luczynski, Jonatan Scharff Willners, Hong. Ziyang, Kaicheng Zhang, Yvan R. Petillot, and Sen Wang. Underwater Visual Acoustic SLAM with Online Extrinsic Calibration. In *IEEE/RSJ IROS*. IEEE, 2021.