Robust Underwater SLAM using Autonomous Relocalisation

Jonatan Scharff Willners * **Yaniel Carreno** * **Shida Xu** *
Tomasz Łuczyński * **Sean Katagiri** * **Joshua Roe** * **Èric Pairet** *
Yvan Petillot * **Sen Wang** *

* *Institute of Sensors, Signals and Systems, Heriot-Watt University, Edinburgh, United Kingdom*
{*j.scharff_willners, y.carreno, sx2000, t.luczynski, s.katagiri, joshua.roe, eric.pairet, y.r.petillot, s.wang*}*@hw.ac.uk*

Abstract: This paper presents a robust underwater simultaneous localisation and mapping (SLAM) framework using autonomous relocalisation. The proposed approach strives to maintain a single consistent map during operation and updates its current plan when the SLAM loses feature tracking. The updated plan transverses viewpoints that are likely to aid in merging the current map into the global map. We present the sub-systems of the framework: the SLAM, viewpoint generation, and high level planning. In-water experiments show the advantage of our approach used on an autonomous underwater vehicle (AUV) performing inspections.

Keywords: SLAM, Path Planning, Temporal Planning, AUV, Stereo Vision, Autonomy, Inspection

1. INTRODUCTION

Underwater robots play a central role in the offshore sector, and deepwater operations would not be possible without them. Currently, remotely operated vehicles are used for most tasks and are teleoperated from a support ship or platforms. However, with the rapid development of the offshore wind energy sector, and the inherent risk and cost associated with the deployment of human personnel offshore, a revolution is happening Birk et al. (2018). remotely operated vehicle (ROV) operators are moving onshore and many technologies developed for AUVs are increasingly being used to develop smart payloads for ROVs. They enable remote operators to plan missions and supervise operations at a higher level as confidence in the robustness of the autonomy modules grows. Central to this is the ability to embed on-board processing and decision making. One key functionality required for asset inspection and intervention is 3D mapping and navigation. Subsea, this is challenging as it is a GPS-denied environment, acoustic sensing is expensive and low-resolution and mainstream vision-based approaches are brittle due to refraction, poor illumination and low visibility. As such, visual SLAM can struggle to maintain a good view of features to perform tracking. To overcome this, we present a robust vision-based autonomous mapping and localisation framework which includes an active relocalisation procedure. This endows the system with autonomous capabilities which seek to maintain a single consistent map of the environment. [1]

⋆ This work was supported by the EPSRC funded ORCA HUB (EP/R026173/1).
⋆⋆J. Scharff Willners, Y. Carreno, S. Xu and T. Łuczyński have contributed equally.
[1] video: https://youtu.be/lPetVtFVe0M

1.1 Statement of Contributions

The novelty and main contributions of this paper can be summarised as follows:

- An algorithm for finding viewpoints (poses) that offer a high chance of relocalisation, based on the analysis of the map built so far.
- A framework for autonomous relocalisation when tracking of visual features fails, based on a high-level planner able to deviate from its current objective and create a temporal plan utilising suggested viewpoints.
- An experimental validation of the proposed approach. Results demonstrate that our approach clearly outperforms the competing approach.

2. RELATED WORK

In general, when tracking is lost, the best chance for a SLAM system to continue mapping is to relocalise itself. In SLAM, relocalisation is referred to as a state where the system tries to utilise place recognition algorithms (e.g., Cummins and Newman (2011); Lowry et al. (2016)) to detect a previously visited region in the accumulated map. Place recognition algorithms used in relocalisation are also commonly used in loop closing operations in case of the SLAM frameworks based on pose graphs. Despite being, algorithmically, very similar procedures, loop closing and relocalisation serve very different purposes: former allows for reduction of errors in the pose estimation accumulated over time, whereas the latter helps with recovering track of visual features or allows for joining disconnected maps (Campos et al., 2020). Stachniss et al. (2004) presented an active loop-closure during SLAM: by taking an opportunistic approach to revisit previously explored areas, the uncertainty could be reduced.

In Bourgault et al. (2002) a robot selects a trajectory according to a potential field based on the unknown areas that are acting as an attractive force and a utility function. This allows to find a balance between tracking known features and exploration while remaining localised. Similar approaches were presented in Lehner et al. (2017), for terrestrial robots, and Chaves et al. (2016); Palomeras et al. (2019b,a) for underwater robots. Other approaches to reduce uncertainty were proposed in Teniente et al. (2012) and Valencia et al. (2013), where the graph from pose-graph SLAM is used as a belief roadmap (BRM). This enables the robot to navigate between poses in the graph and selecting paths that yield the lowest uncertainty.

The SLAM solution presented in this paper builds on our previous work (Xu et al., 2021), an extension of ORB-SLAM3 (Campos et al., 2020) suitable for underwater environments. The SLAM system integrates dead reckoning (DR) from internal sensors of the AUV (e.g., Doppler Velocity Log (DVL), Inertial Measurement Unit (IMU), and depth sensor). By endowing ORB-SLAM3 with DR from internal sensors, it is able to estimate and track the robot's pose even when there are no visual features to track (Vargas et al., 2021). However, as DR is based on the integration of sensor data containing potential drift and noise, it will have an error growing without bound in pose estimation, and losing track of visual feature will result in multiple maps weakly connected with DR. The growing error will further cause the inconsistency of the map as shown in Fig. 1a.

3. ROBUST UNDERWATER SLAM

Opposed to using active-SLAM to seek future loop-closures, we look at improving the contingency of the map continuously during operation. The argument behind this is that features to track in the underwater domain are often sparse and require observations from a short distance due to limited visibility. Therefore, losing tracking of the object being inspected leads to unbounded growth of the error in the pose estimation until tracking is regained. Hence, if tracking is not regained in a short enough time the error can grow too large, such that finding the way back to the structure is no longer feasible. Likewise, if the SLAM system operates on multiple sub-maps, the cumulative error between these might grow too large to be reliable for global navigation (Pairet et al., 2018, 2020).

The problem we look at, to actively merge maps as soon as a sub-map has been created, shares similarities with active SLAM for loop closures. It does however differ in the sense that our system will focus on solving the map merge before continuing the exploration or inspection and therefore should have a more reliable localisation as opposed to using multiple sub-maps. While the problem we try to solve is different than many aforementioned active SLAM approaches, our approach has similarities to BRM within the pose-graph of SLAM, but instead of using the keyframes (poses) in the pose-graph as nodes able to be visited, we use these to define a sampling space. The sampling and evaluation of viewpoints for map merging around the keyframes extends on Palomeras et al. (2019b).

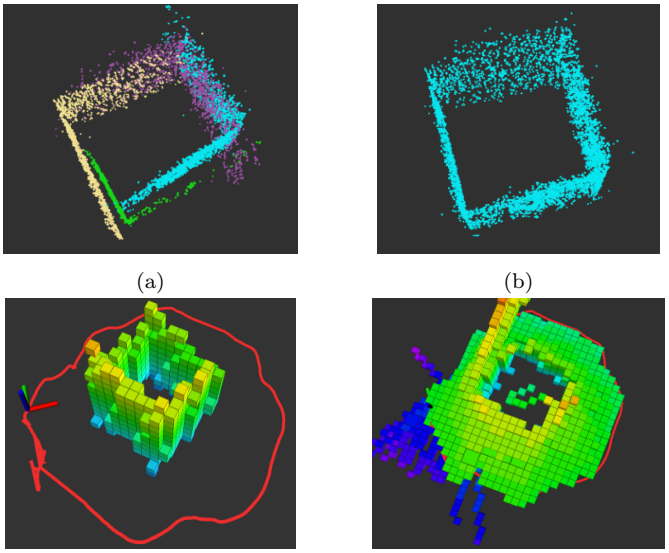


Fig. 1. **(a)** depicts a set of misaligned sub-maps caused by feature tracking loss and relying on DR. **(b)** shows the ground truth. **(c)** illustrates the occupied voxels and **(d)** the free voxels, in the OctoMap that is used for path planning and viewpoint generation.

3.1 Image Acquisition

To enable a vision-based algorithm developed to work in-air, like ORB-SLAM3, to be used underwater, a few issues have to be addressed. Underwater images not only often suffer from exceptionally challenging illumination (low light, high dynamic range), but also from refraction-based distortions and poor visibility due to backscatter. In this work, we utilise earlier studies on these problems. The hardware mounted on the ROV was designed especially for close-range 3D inspections, to satisfy the conditions of working in murky water (Łuczyński et al., 2019). The custom design also allowed for applying a Pinax camera calibration and image rectification model (Łuczyński et al., 2017), which speeds up the deployment of the whole system both in the research conditions and offshore. Finally, the colour balance is also addressed with a state-of-the-art algorithm (Bianco and Neumann, 2017).

3.2 Keyframe Selection

ORB-SLAM3, similarly to other visual SLAM solutions based on pose graph, relies on the keyframes stored in the graph and representing robot poses. ORB-SLAM3 maintains a set of keyframes and their connected map points as the local map. Each keyframe stores the information such as, the pose, feature points and its connection to map points and other keyframes, which would be used for optimisation and place recognition (loop closure). An efficient keyframe selection and culling scheme are adopted to provide good tracking robustness with limited computational resources.

To build the proposed relocalisation strategy, we rely on these keyframes, already available in the ORB-SLAM3 framework. In order to avoid the change of map origin, we only allow the merging of a sub-map with a prior map.

Once SLAM lose tracking of visual features, we evaluate all keyframes in the previous map via the visual feature quality, temporal and geometric relationship with the last keyframe obtained before losing tracking. To this end, a set of good candidates for relocalisation is chosen and sent to the planner. Let us define the given keyframe as $\mathcal{K}_i$ and a set of keypoints kp belonging to $\mathcal{K}_i$ and used in the map as $\mathcal{K}_i^{kp}$. The score $S(\mathcal{K}_i)$ given to each keyframe is calculated as the highest integer s for which it is true that at least $s * 100$ of $kp \in \mathcal{K}_i^{kp}$ can be observed by s other keyframes $\mathcal{K}_{1-s}$. E.g. $S(\mathcal{K}_i) = 6$ means, that there are at least 600 keypoints in $\mathcal{K}_i$ that are simultaneously observed by 6 keyframes. In practice, $S(\mathcal{K}_i)$ is found by iteratively checking the value of s. However, the condition required to increase the value of s gets increasingly harder with each step, therefore the score can be usually computed with just a few iterations. That allows for easy separation of useful candidates in a wide range of conditions, without overwhelming the system with unnecessarily grainy metric.

3.3 Mapping for Planning

The SLAM generates a sparse point cloud which, after minor processing, can be used for navigation. We represent the environment obtained via our SLAM as an OctoMap (Hornung et al., 2013). An OctoMap is constituted of voxels representing a 3-dimensional occupancy grid map at variable resolution. The voxels in the OctoMap are initially categorised as unknown. The point cloud generated by the SLAM is used to find the occupied voxels in the octomap, and the ray between the camera origin and associated points in the point cloud marks voxels as free. The OctoMap efficiently stores the voxels as an octree structure, which optimises memory usage and provides fast access. The map used for planning and viewpoint generation is periodically updated from SLAM. An example of the occupied voxels generated through the SLAM's point cloud can be seen in Fig. 2.

4. AUTONOMY AND PLANNING

When a robot is performing a task, such as an inspection of an offshore asset using SLAM, it is operating in an hostile environment. For example, currents and waves generate external forces that directly disturb the robot, and water turbidity limits the robot's exteroceptive visibility. In this section we present an end-to-end planning framework to counter the loss of feature tracking. The planning framework is based on two modules: a path planner able to generate viewpoints for relocalisation, and a mission and high level planner that manage the tasks and actions to perform.

4.1 Path Planner - Relocalisation Viewpoint Generation

As discussed in Section 3.2, for each keyframe in the map a score can be obtained, describing how useful this frame is in the context of relocalisation. Whenever the the SLAM is lost, a set of the highest scoring keyframes $\mathcal{K}$, belonging to the map that just stopped being tracked, is passed to the planner. For each keyframe $\mathcal{K}_i \in \mathcal{K}$, m collision-free viewpoints are sampled. The m viewpoints' sampling space

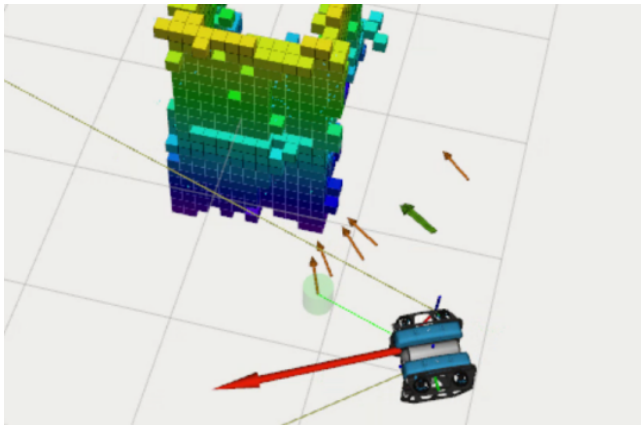


Fig. 2. The AUV has lost tracking. SLAM sends a set of 5 keyframes (green arrows). These keyframes specify the sampling space, and a new set of 5 viewpoint deemed good for relocalisation is generated. The keyframes are very dense, while the new sampled viewpoints (orange arrows) are more sparsely distributed, giving the set a probabilistically higher chance to relocalise than the dense keyframes from SLAM.

is centred around the corresponding keyframe with a specified maximum distance in x,y,z position and difference in yaw angle. This gives a set of viewpoints around keyframe $\mathcal{K}_i$ as $\mathcal{V}_i$. The j:th viewpoint in $\mathcal{V}_i$ is noted as $\mathcal{V}_{i,j}$. The viewpoints in $\mathcal{V}_i$ are then evaluated using a utility function to find a single viewpoint for each keyframe. The utility function (1) is composed of 4 weighted reward functions. Each reward function is in the range $[0, 1]$. The weights are normalized such that the sum of the weights is equal to 1.0. Hence, the sum of the 4 weighted utility functions is in the range $[0, 1]$. A similar approach for sampling and evaluation of a viewpoint was used in Palomeras et al. (2019b) during next-best-view planning for frontier exploration. However, utility function presented in this paper considers additional safety features and the optimal sensing range. The 4 reward functions are described below and can be seen in (1):

- *Distance*, see (1a) – the closer to the keypoint the higher the reward. The Euclidean distance as between a keyframe and a viewpoint as $|\mathcal{K}_i - \mathcal{V}_{i,j}|$. The maximum distance a viewpoint can be sampled from a keyframe as $\mathcal{D}_{max}$. If $|\mathcal{K}_i - \mathcal{V}_{i,j}| = \mathcal{D}_{max}$ the reward is 0 and a if $|\mathcal{K}_i - \mathcal{V}_{i,j}| = 0$ the reward is 1.
- *Occupied*, see (1b) – The number of occupied voxels seen from $\mathcal{V}_{i,j}$ is estimated by a simulated sensor in the OctoMap is noted as $\mathcal{V}_{i,j}^{occupied}$. The maximum voxels seen by a viewpoint in $\mathcal{V}_i$ is $max(\mathcal{V}_i^{occupied})$. The reward function for the number of observed voxels is the ratio of the maximum viewed voxels of a viewpoint in the set. This favours viewpoints that view many previously observed voxels.
- *Optimal Viewing Distance*, see (1c) – the optimal distance to view the closest cell at. This should improve the quality of the collected data as it rewards viewpoints closer to the optimal distance from the structure. The reward for the function is 1 if the distance is the same as the optimal sensing distance.

The reward decreases from the optimal towards 0 at the minimum observation distance as well as towards the maximum sensing distance.

- *Safe*, see (1d) – If the sampled state is in voxels marked as free it is considered safe and the function returns 1, otherwise 0. This is not the same as being collision-free. But this is within a region that has been sensed as free, instead of occupied (a sample that does not pass the collision check is discarded) or unknown (neither free nor occupied).

$$u = dw_d + ow_o + cw_c + sw_s \tag{1}$$
$$d = \frac{\mathcal{D}_{max} - |\mathcal{K}_i - \mathcal{V}_{i,j}|}{\mathcal{D}_{max}} \tag{1a}$$
$$o = \frac{\mathcal{V}_{i,j}^{occupied}}{max(\mathcal{V}_i^{occupied})} \tag{1b}$$
$$c = optDist(\mathcal{V}_{i,j}) \tag{1c}$$
$$s = safe(\mathcal{V}_{i,j}) \tag{1d}$$

By selecting the sampled viewpoint with the highest utility for each of the n keyframes we obtain a set of viewpoints that have a high probabilistic likelihood to succeed in relocalising the robot. The set of relocalisation viewpoints is passed to the high level planner to update the current plan to focus on relocalising before to continuing its current objective. An example of how a set of keyframes are used to generate relocalisation viewpoints when the robot has lost tracking can be seen in Fig. 2.

4.2 Autonomy Framework - High Level Planner

Automated planning technologies have been successfully introduced in a growing number of underwater applications for solo (Maurelli et al., 2016) and multi-AUV (Carreno et al., 2020) missions to provide the high-level reasoning of the task. AI planner solutions guide the robot to implement a set of actions (plan) that solve mission goals according to its capabilities, which the problem's model defines. In this work, we use the temporal planner OPTIC (Benton et al., 2012), which obtains more realistic plans supporting concurrent actions, synchronising multiple tasks, and handling deadlines. The task planning approach finds a plan that leads the robot to explore a set of points in the environment to inspect a structure. The problems we solve follow the temporal planning problem representation described by the Planning Domain Definition Language (PDDL) version 2.1 [2] (Fox and Long, 2003) with continuous effects. A temporal planning problem is defined as a tuple:

$$\mathcal{P}_T := \langle \mathcal{P}, \mathcal{V}, \mathcal{A}, \mathcal{I}, \mathcal{G}, \mathcal{T} \rangle, \tag{2}$$

where $\mathcal{P}$ is a set of propositions defining the available resources in the platform and the environment's properties; $\mathcal{V}$ is a set of fluents that describes the platform and world conditions that can change over time; $\mathcal{A}$ is a set of instantaneous and durative actions; $\mathcal{I}$ is the initial state defined by the propositions and fluents ($\mathcal{P} \cup \mathcal{V}$), and $\mathcal{G}$ is a set of goals, where each goal $g \in \mathcal{G}$ is defined as $p \cup v$, where $p \subseteq \mathcal{P}$ and $v \subseteq \mathcal{V}$; and $\mathcal{T}$ is a set of time windows. Each time window is defined using Timed Initial Literals (TILs) which define the time $t \in \mathcal{T}$ at which particular propositions of $p \in \mathcal{P}$ become true/false. Each $a \in \mathcal{A}$ 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. For instance, our domain defines the durative action `navigation` which specifies as preconditions for its implementation to know 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 (`at ?r - robot ?wpf - waypoint`) and the new waypoint is 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 is a precondition required to implement actions that depend on a camera

[2] PDDL2.1 introduces the notion of time in the action definitions.

```
  Time:    (Action Name)                                  [Duration]
 0.000:    (navigation bluerov2 wp0 wp10)                   [10.000]
10.001:    (turnon-light bluerov2  wp10)                     [2.000]
12.002:    (map bluerov2 wp10 wp11)                         [10.000]
22.003:    (map bluerov2 wp11 wp12)                         [10.000]
(...)
72.007:    (navigation bluerov2 wp40 wp41)                  [10.000]
82.008:    (turnoff-light bluerov2  wp41)                    [2.000]
84.009:    (recover bluerov2  wp41)                          [1.000]
```

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

The mission plan Π_T generated by OPTIC to solve the $\mathcal{P}_T$ can be seen as 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 $\mathbb{R}_{>0} = \{x \in \mathbb{R} \,|\, x > 0\}$. Fig. 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 [3] around a structure to obtain its image reconstruction. Therefore, problem's goals can be `(inspected wp10) ... (inspected wp41)`. One of the advantages of task planners is that we can obtain plans that reach goals associated with different types of action and requirements, assuming we have domain actions that support the goal execution. For instance, the actions to turn the AUV lights and recovery support the implementation of goals (`light_on wp10`) and (`recovered wp41`), which are not related to the structure's reconstruction goals. However, they represent another set of mission requirements defined by the operator at the initial state.

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 many missions in the underwater domain that require algorithms reasoning about uncertainty considering the environment is highly dynamic. In this paper, the problem we solve where the system can lose

[3] The set of points are generated considering we have a notion of the distance between the robot and the structure centre. The algorithm defines a set of coordinates to explore based on the number of points the AUV needs to reach for structure mapping.

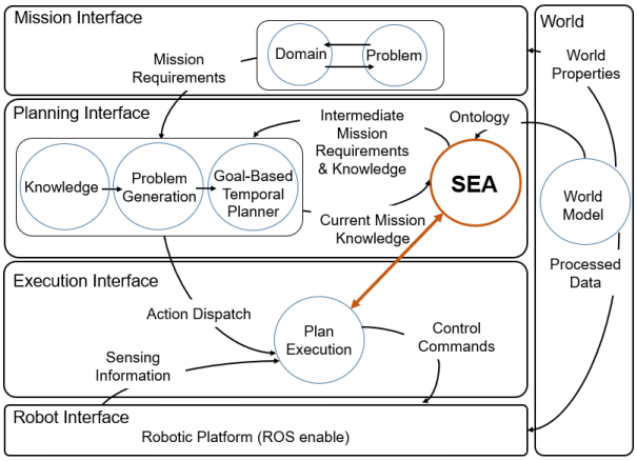


Fig. 4. High Level task planning architecture. The SEA provides robustness to the automated planning system dealing with unexpected situations during plan execution.

```
  Time:    (Action Name)                                  [Duration]
 0.000:    (navigation bluerov2 wp0 wp10)                   [10.000]
10.001:    (turnon-light bluerov2  wp10)                    [2.000]
12.002:    (map bluerov2 wp10 wp11)                         [10.000]
-------------------------------------------------------------------
0.000:     (map bluerov2 wpra_0 wpr0)                       [0.500]
0.501:     (map bluerov2 wpr0 wpr1)                         [3.000]
3.502:     (map bluerov2 wpr1 wpr2)                         [1.200]
-------------------------------------------------------------------
0.000:     (map bluerov2 wpra_1 wp11)                       [2.500]
2.501:     (map bluerov2 wp11 wp12)                        [10.000]
(...)
62.506:    (navigation bluerov2 wp40 wp41)                 [10.000]
72.507:    (turnoff-light bluerov2  wp41)                   [2.000]
74.507:    (recover bluerov2  wp41)                         [1.000]
```

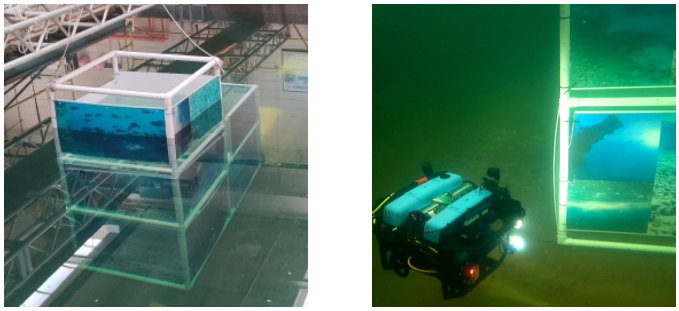
Fig. 5. A fragment of an original temporal plan and the recovery plan introduced as a result of failures.

track during the mission due to multiple external effects is a clear case where temporal planners are not, in principle, the optimal option. Our approach provides a solution to this issue while keeping the OPTIC planner in place as the problem solver. We introduce in our planning framework a new element called Situational Evaluation and Awareness (SEA) (Carreno et al., 2021), which creates a bridge between plan reasoning and execution to maintain the system running for long-term periods avoiding missing failures. SEA is a failure solver which acts when failures occur, driving the robot from the failure state by proposing alternative sub-plans. SEA makes a diagnostic of the dynamic environment to identify the type of failure and offers alternatives solutions to the planner that allows the system to recover before executing the remaining goals from the original mission. Fig. 4 shows the high level planning architecture which connects with ROSPlan (Cashmore et al., 2015) to execute the high-level plan:

- *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 [4] which is parsed and dispatch to the Execution Interface. Intermediate Mission Requirements extend the operator's Mission Requirements that SEA introduces when failures occur during the plan implementation. SEA receives feedback from the knowledge component in the planning process and the Plan Execution to define the propositions to change and goals to add to avoid failures. Besides, SEA is connected to the World Model thought and ontology that defines the possible type of failures associated with the environment. SEA can force the actual plan execution based on failure accounted using the direct connection with the Execution Interface.
- *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 identify the failure source. This interface embeds the low-level algorithms 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 data to evaluate the mission's execution.

SEA maintains the mission survivability and adaptability over long-term missions. The evaluator forced the generation of recovery plans which are introduced in the execution of the original plan to solve particular failures. Fig. 5 shows an example of a solution where the execution of the original plan (see Fig. 3) is interrupted, as a consequence, SEA receives a notification the AUV has lost its localisation when executing and 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 points to recover, provided by the path planner for viewpoint generation, and sends a set of new goals and knowledge to the Planning System that allows OPTIC to generate a new plan (recovery-plan). The Planning Interface generates a new plan that forces the robot to navigate these points to localise (actions in blue). Suppose during the execution of the recovery plan (blue); the AUV achieves the relocalisation. In that case, the SEA determines the recovery plan is completed, even when not all actions are achieved. The system will then replicate the original plan's incompleted mission goals to keep with the mission implementation. Suppose the AUV completes the recovery plan (intermediate), and relocalisation is not achieved. In that 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 mission goals in the Mission Requirements. Our approach can deal with different failure types, while considering their risk level. The system embeds a recovery mechanism to deal with unexpected mission failures that includes communication with an operator.

[4] Actual planning times are restricted to 5 sec. (maximum) considering the dynamics of the environment.



(a) (b)

Fig. 6. **(a)** The facilities used for testing at Heriot-Watt University. The artificial structure was used for inspection. **(b)** The AUV during an inspection.

5. EXPERIMENTAL EVALUATION

We deploy the proposed framework on a BlueROV2 [5] to perform an autonomous inspection of an artificial structure (see Fig. 6). Our BlueROV2 is a fully ROS-enabled (Quigley et al., 2009) 6 degree of freedom (DoF) ROV with 4 vertical and 4 horizontal thrusters, with a DVL [6] integrated for DR, a stereo vision system, and a back seat driver for autonomous operations, enabling the ROV to operate as an AUV. The artificial structure to inspect is placed in the middle of a $12 \times 12 \times 2.5$ metres water tank at Heriot-Watt University. The purpose of the test is to assess the capability of our proposed 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. We set our viewpoint generation to get 100 samples around each keyframe with the following weights: $w_d = 0.1$, $w_o = 0.6$, $w_c = 0.2$ and $w_s = 0.1$. The sampling space is set to ± 0.3 metres for x and y, ± 0.1 metres in z (depth) and ± 0.4 radians for yaw around the keyframe. 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. This was done at the same waypoints in all trials to create a controlled comparison. Fig. 7 depicts the number of sub-maps generated during the inspection. It can be seen that the proposed approach is able to relocalise and merge the maps in a short amount of time (blue line), whereas both runs of the autonomous waypoint controller result in multiple sub-maps (red and green lines). Fig. 8 illustrates the resulting 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 Fig. 8c-8d), our approach ends with a single consistent map of the structure (see Fig. 8a-8b). These results demonstrate that our approach can autonomously cope with feature tracking loss by triggering relocalisation plans that allow merging sub-maps.

[5] https://bluerobotics.com/store/rov/bluerov2/
[6] https://waterlinked.com/product/dvl-a50/

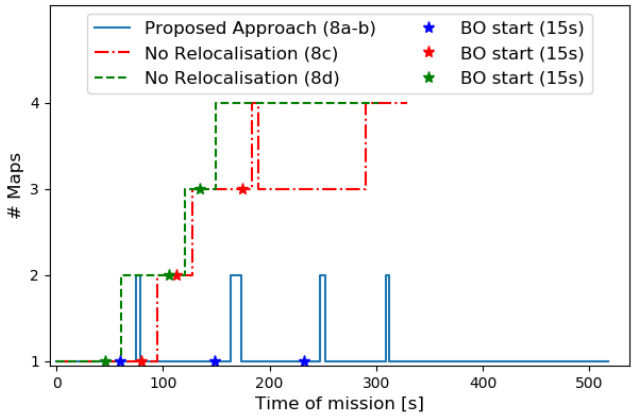


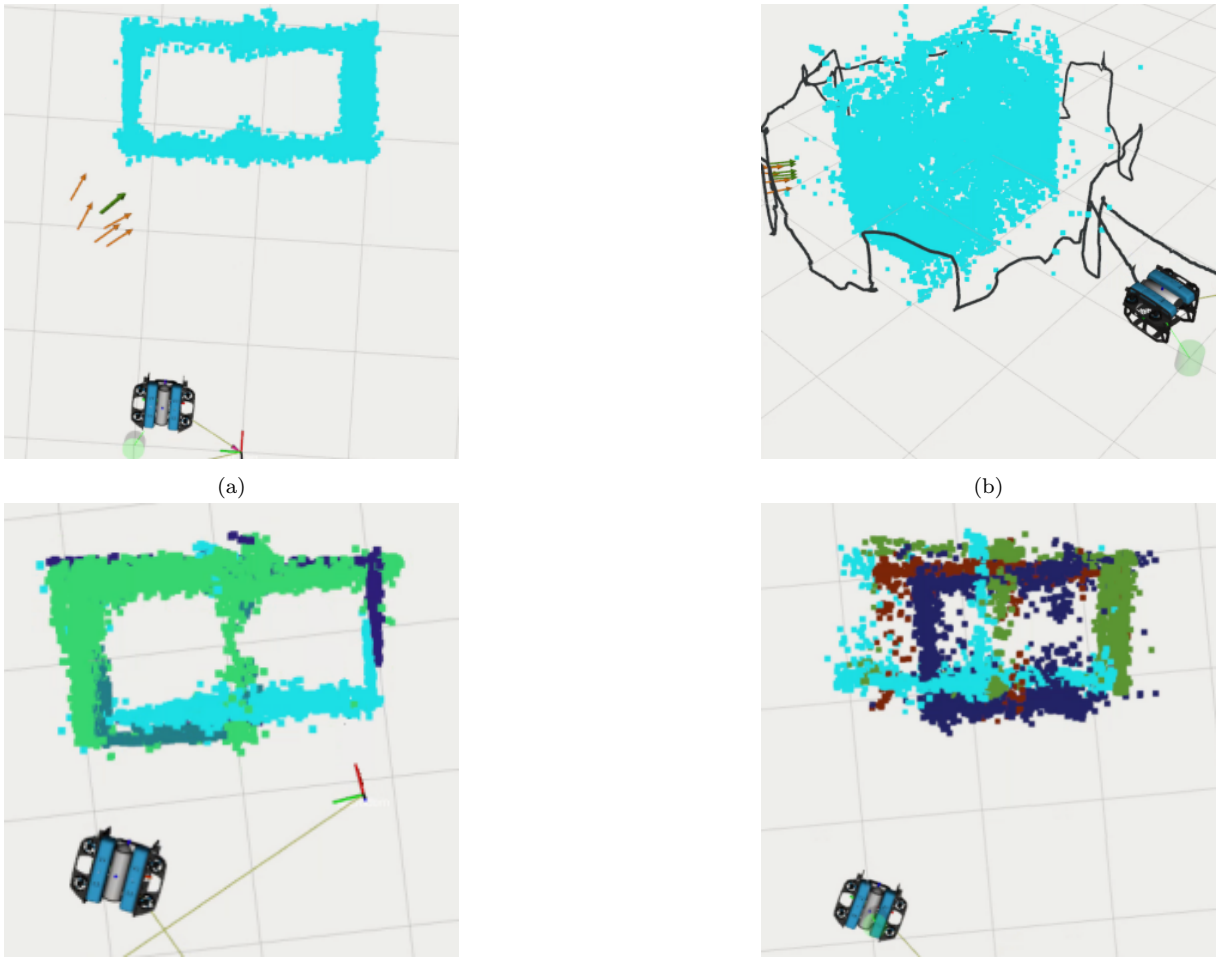
Fig. 7. The number of maps along the duration of the inspections. We introduced controlled 15 seconds of lost feature tracking (black outs (BO)) at the same 3 waypoints during the inspections. The graph shows the scenarios as in Fig. 8.

A clear advantage of the proposed system can be seen in Fig. 9. While moving around the corner of the structure (as seen in Fig. 6), the vehicle naturally loses track of the features. When this happens, our system generates a new map (magenta coloured map in Fig. 9a) 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.

6. CONCLUSION AND FUTURE WORK

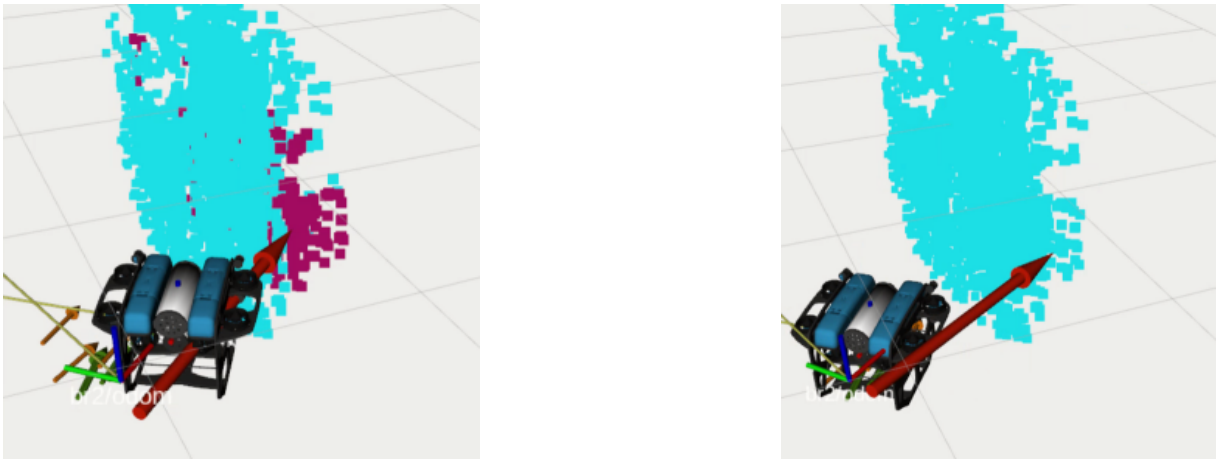
In this paper we have presented a framework for autonomously relocalisation when a SLAM system loses tracking. This enables the system to maintain a single consistent map instead of multiple sub-maps and hence increasing reliability. We present the sensor and vehicle set up along with the framework consisting of image-enhancing methods used to improve the quality of the captured images, the SLAM system, a viewpoint generator to increase the chance of relocalisation and a high-level planner. We show that our proposed approach is able to finish inspections with a single map by actively searching for map merges when tracking is lost in SLAM. We show that executing the same inspection mission, but without using the adaptive plan when features are lost, results in multiple sub-maps at the end of the mission.

The current system expects to continuously have features to track, as it will otherwise start a relocalisation procedure. For the future, we are extending the framework with an analytical procedure to determine if relocalisation should be used or not. This is as the robot might need to move through areas in which it will not be able to perform feature tracking, e.g., in open water, between structures.



(a) (b) (c) (d)

Fig. 8. In **(a)** the final map when using the proposed approach for robust SLAM using relocalisation can be seen. **(b)** shows the full trajectory of the robot based on the pose estimates from SLAM. **(c)** and **(d)** shows two scenarios where the robot was following the same waypoints as in **(a)**, but without the proposed autonomous relocalisation procedure. The proposed approach has a single consistent map at the end of the inspection, while not using our relocalisation approach results in 4 sub-maps.



(a) (b)

Fig. 9. **(a)** The AUV is moving around a corner, by doing so it loses track of the current map (blue) and creates a second map (magenta). **(b)** The relocalisation procedure is initiated, which moves the robot to a state in which it is able to merge the two maps before continuing the mission with a single consistent map.

REFERENCES

Benton, J., Coles, A.J., and Coles, A. (2012). Temporal planning with preferences and time-dependent continuous costs. In *ICAPS*.

Bianco, G. and Neumann, L. (2017). A fast enhancing method for non-uniformly illuminated underwater images. In *OCEANS 2017 - Anchorage*, 1–6.

Birk, A., Antonelli, G., Di Lillo, P., Simetti, E., Casalino, G., Indiveri, G., Ostuni, L., Turetta, A., Caffaz, A., Weiss, P., Gobert, T., Doernbach, T., Chemisky, B., Gancet, J., Siedel, T., Govindaraj, S., Martinez, X., Letier, P., Mueller, C., Luczynski, T., Gomez Chavez, A., Koehntopp, D., Kupcsik, A., Calinon, S., and Tanwani, A.K. (2018). Dexterous Underwater Manipulation from Onshore Locations: Streamlining Efficiencies for Remotely Operated Underwater Vehicles. *IEEE Robotics & Automation Magazine*, 25(4), 24–33. doi: 10.1109/MRA.2018.2869523.

Bourgault, F., Makarenko, A.A., Williams, S.B., Grocholsky, B., and Durrant-Whyte, H.F. (2002). Information based adaptive robotic exploration. *IEEE International Conference on Intelligent Robots and Systems*, 1(June 2014), 540–545. doi:10.1109/IRDS.2002.1041446.

Campos, C., Elvira, R., Rodríguez, J.J.G., Montiel, J.M., and Tardós, J.D. (2020). Orb-slam3: An accurate open-source library for visual, visual-inertial and multi-map slam. *arXiv preprint arXiv:2007.11898*.

Carreno, Y., Pairet, È., Petillot, Y., and Petrick, R.P. (2020). A decentralised strategy for heterogeneous auv missions via goal distribution and temporal planning. In *ICAPS*, volume 30, 431–439.

Carreno, Y., Scharff Willners, J., Petillot, Y.R., and Petrick, R.P.A. (2021). Situation-Aware Task Planning for Robust AUV Exploration in Extreme Environments. In *Proceedings of the IJCAI Workshop on Robust and Reliable Autonomy in the Wild*.

Cashmore, M., Fox, M., Long, D., Magazzeni, D., Ridder, B., Carrera, A., Palomeras, N., Hurtos, N., and Carreras, M. (2015). ROSPlan: Planning in the Robot Operating System. In *ICAPS*.

Chaves, S.M., Kim, A., Galceran, E., and Eustice, R.M. (2016). Opportunistic sampling-based active visual SLAM for underwater inspection. *Autonomous Robots*, 40(7), 1245–1265. doi:10.1007/s10514-016-9597-6.

Cummins, M. and Newman, P. (2011). Appearance-only SLAM at large scale with FAB-MAP 2.0. *The International Journal of Robotics Research*, 1100–1123. doi:10.1177/0278364910385483.

Fox, M. and Long, D. (2003). PDDL2.1: An extension to PDDL for expressing temporal planning domains. *JAIR*, 20, 61–124.

Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3), 189–206. doi:10.1007/s10514-012-9321-0.

Lehner, H., Schuster, M.J., Bodenmuller, T., and Kriegel, S. (2017). Exploration with active loop closing: A trade-off between exploration efficiency and map quality. *IEEE International Conference on Intelligent Robots and Systems*, 2017-Septe, 6191–6198. doi: 10.1109/IROS.2017.8206521.

Lowry, S., Sunderhauf, N., Newman, P., Leonard, J.J., Cox, D., Corke, P., and Milford, M.J. (2016). Visual Place Recognition: A Survey. *IEEE Transactions on Robotics*, 32(1), 1–19. doi:10.1109/TRO.2015.2496823.

Łuczyński, T., Łuczyński, P., Pehle, L., Wirsum, M., and Birk, A. (2019). Model based design of a stereo vision system for intelligent deep-sea operations. *Measurement: Journal of the International Measurement Confederation*, 144, 298–310. doi: 10.1016/j.measurement.2019.05.004.

Łuczyński, T., Pfingsthorn, M., and Birk, A. (2017). The Pinax-model for accurate and efficient refraction correction of underwater cameras in flat-pane housings. *Ocean Engineering*, 133(March), 9–22. doi: 10.1016/j.oceaneng.2017.01.029.

Maurelli, F., Carreras, M., Salvi, J., Lane, D., Kyriakopoulos, K., Karras, G., Fox, M., Long, D., Kormushev, P., and Caldwell, D. (2016). the pandora project: A success story in auv autonomy. In *IEEE OCEANS – Shanghai*.

Pairet, È., Hernández, J.D., Lahijanian, M., and Carreras, M. (2018). Uncertainty-based online mapping and motion planning for marine robotics guidance. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2367–2374. IEEE.

Pairet, È., Hernández, J.D., Petillot, Y., and Lahijanian, M. (2020). Online mapping and motion planning under uncertainty for safe navigation in unknown environments. *arXiv preprint arXiv:2004.12317*.

Palomeras, N., Carreras, M., and Andrade-Cetto, J. (2019a). Active SLAM for autonomous underwater exploration. *Remote Sensing*, 11(23), 1–19. doi: 10.3390/rs11232827.

Palomeras, N., Hurtos, N., Vidal, E., and Carreras, M. (2019b). Autonomous exploration of complex underwater environments using a probabilistic next-best-view planner. *IEEE Robotics and Automation Letters*, 4(2), 1619–1625. doi:10.1109/LRA.2019.2896759.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Ng, A. (2009). ROS : an open-source Robot Operating System. *ICRA Workshop on Open Source Software*.

Stachniss, C., Hähnel, D., and Burgard, W. (2004). Exploration with active loop-closing for FastSLAM. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2, 1505–1510. doi: 10.1109/iros.2004.1389609.

Teniente, E.H., Valencia, R., and Andrade-Cetto, J. (2012). Dense outdoor 3D mapping and navigation with Pose SLAM. *Workshop de Robótica Experimental*, 567 – 572.

Valencia, R., Morta, M., Andrade-Cetto, J., and Porta, J.M. (2013). Planning reliable paths with pose SLAM. *IEEE Transactions on Robotics*, 29(4), 1050–1059.

Vargas, E., Scona, R., Willners, J.S., Luczynski, T., Cao, Y., Wang, S., and Petillot, Y.R. (2021). Robust Underwater Visual SLAM Fusing Acoustic Sensing. *IEEE International Conference on Robotics and Automation 2021*.

Xu, S., Luczynski, T., Scharff Willners, J., Ziyang, H., Zhang, K., Petillot, Y.R., and Wang, S. (2021). Underwater Visual Acoustic SLAM with Online Extrinsic Calibration. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.