IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 9, 2021, accepted September 24, 2021, date of publication October 4, 2021, date of current version October 12, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3117343*

# Adaptive Visual Information Gathering for Autonomous Exploration of Underwater Environments

**ERIC GUERRERO, FRANCISCO BONIN-FONT, AND GABRIEL OLIVER**
Departament de Matemàtiques i Informàtica, Universitat de les Illes Balears, 07122 Palma, Spain

Corresponding author: Eric Guerrero (e.guerrero@uib.eu)

This work was supported in part by the Spanish Ministry of Economy and Competitiveness [Agencia Estatal de Investigación (AEI)/Fondo Europeo de Desarrollo Regional (FEDER), European Union (EU)] under Contract DPI2017-86372-C3-3-R, and in part by the Government of the Balearic Islands [Govern de les Illes Balears (CAIB)] under Grant FPI/2031/2017.

**ABSTRACT** This work presents the development and field testing of a novel *adaptive visual information gathering* (AVIG) framework for autonomous exploration of benthic environments using AUVs. The objective is to adapt dynamically the robot exploration using the visual information gathered *online*. This framework is based on a novel *decision-time adaptive replanning* (DAR) behavior that works together with a *sparse Gaussian process* (SGP) for environmental modeling and a *Convolutional Neural Network* (CNN) for semantic image segmentation. The framework is executed in mission time. The SGP uses semantic data obtained from stereo images to probabilistically model the spatial distribution of certain species of seagrass that colonize the sea bottom forming widespread meadows. The uncertainty of the probabilistic model provides a measure of sampling informativeness to the DAR behavior. The DAR behavior has been designed to execute successive informative paths, without stopping, considering the newest information obtained from the SGP. We solve the *information path planning* (IPP) problem by means of a novel *depth-first* (DF) version of the *Monte Carlo tree search* (MCTS). The DF-MCTS method has been designed to explore the state-space in a depth-first fashion, provide solution paths of a given length in an *anytime* manner, and reward smooth paths for field realization with non-holonomic robots. The complete framework has been integrated in a ROS environment as a high level layer of the AUV software architecture. A set of simulations and field testing show the effectiveness of the framework to gather data in *P. oceanica* environments.

**INDEX TERMS** Adaptive, exploration, information, marine, planning, posidonia, robotics, semantic, underwater.

## I. INTRODUCTION

### A. MOTIVATION

Autonomous exploration is being an important research area for the last decades. The main purpose of robotic exploration is to gather data in areas that humans cannot reach, such as, in deep waters, in the space, or in missions that involve unsafe or hazardous environments or actions. And, why autonomous? Such exploration can be performed by means of an operator controlling remotely a robot system. However this relies on having a proper bidirectional communication between the operator and the robot (for instance to transmit robot images and operator commands), and in many scenarios such communication is not possible, or is very limited, and the robot has to be equipped with some degrees of autonomy.

AUVs have shown a big improvement of their autonomous capabilities during last years. The localization problem has been tackled in many areas [1], different strategies of sensor fusion [2] and SLAM (Simultaneous localization and mapping) have shown to provide very good performance [3] and [4]. Additionally, CNNs (Convolutional neural networks) have shown to be very effective for online image segmentation [5], which provides robots with a semantic understanding of the environment. And the computing capabilities, energy efficiency and control of autonomous robots are also improving. However, data gathering missions with autonomous robots are normally limited to the use of pre-programmed paths, and their performance is supported only on a reliable localization and control.

The associate editor coordinating the review of this manuscript and approving it for publication was Varuna De Silva.

This paper is focused on performing autonomous underwater exploration on *Posidonia oceanica* seagrass meadows. *P. oceanica* is an endangered endemic seagrass species from the Mediterranean Sea, that has seen an estimated 35% aggregate reduction over the last 50 years [6]. The absence of definitive answers to why the *P. oceanica* is receding is broadly associated to the lack of precise data. Abadie *et al.* [7] highlight a lack of data on the spatial distribution, which can be obtained using last developments on AUVs.

### B. RELATED WORK

Many applications in robotics share the common objective of exploring an unknown environment for recording data to represent it. *Information Gathering* (IG) algorithms guide such exploration using an information metric which represents the informativeness of the environment variable under study in particular locations, and this metric is used to drive the data recording process towards the more informative spots whilst minimizing a cost, such as the number of measurements, the navigation distance or the mission time. IG algorithms have been used for different types of exploration, for instance; (a) goal-based, where the objective is traveling from an initial location to a goal location with a given cost budget [8]–[10], (b) front-based, for traversing a given threshold area [11], [12], (c) frontier-based, usually for indoor environment mapping [13]–[16], (d) multimodal, using different data sources [17], [18], (e) multirobot, using multiple robots [19]–[21], (f) hotspot-based, to find environmental variable hotspots [22], [23], and (g) coverage-based, for environmental variables dense estimation [24], [25], which is, in fact, the focus of this work.

The methods developed for such exploration are often differentiated by four components: (1) the technique for modeling the environment; (2) the information function; (3) the *Informative Path Planning* (IPP) strategy, and (4) the adaptive strategy for replanning.

One way to obtain environmental models is using a *Gaussian Process* (GP) fed with the data collected by an AUV [26]. GP are a powerful nonparametric set of techniques that can handle a large variety of problems, and have the ability to learn spatial correlation with stochastic noisy measured data [27]. The key feature of GP for IG algorithms is their ability to handle both, data uncertainty and data incompleteness, effectively, which allows to perform dense estimation of environmental variables for coverage-based exploration purposes. The problem of GPs is that they do not scale well, having a time complexity of $\mathcal{O}(n^3)$. Some works overcome this issue by proposing the use of a sparse version of the GP [24], [28] or local maps fusion using Bayesian Committee Machine (BCM) [16] to decrease time complexity and improve the online execution.

The so called *information function* is used to point out the more relevant spots to visit (or revisit), and is directly related to the environmental variable under study (for instance salinity [24], cyanobacteria [25] or plankton [29]). It is usually computed considering the modelling uncertainty. Whilst the authors in [30] propose the use of an interpolated version of the GP model predicted variance, most of the methods use either *Differential Entropy* (DE) or *Mutual Information* (MI) as information function. Using DE results in lower computation times and higher uncertainty reduction than using MI, when computed from a GP prediction in a stationary setup [31]. Despite of the submodularity property of MI [32], the computational time of such information metric results prohibitive in many applications.

*Informative path planners* define the most informative trajectories for IG, applying constraints such as the planning time or the travelling cost. The IPP strategies used for IG can be classified in two groups; myopic or nonmyopic. Myopic strategies are short sighted, they only plan one step forward, and usually work following greedy heuristics [33], [34]. Thompson *et al.* [35] proposed a complete pipeline for autonomous planetary exploration, they propose a *science on the fly* to adapt the robots exploration to the collected instrument data, using a GP for environmental modeling and a greedy method for IPP. In contrast, nonmyopic strategies look several steps ahead, providing paths that may result better on the long run; such strategies are usually graph-based, sampling-based or evolutionary-based. Whilst graph-based solutions are usually time expensive, they have been used for small state-space scenarios [8], [24] and in non-adaptive frameworks [18] where the mission path is processed offline. In contrast, sampling and evolutionary-based strategies, have been used for adaptive frameworks, and have yield high performance. For instance, [23], [31], [36]–[38] base their strategies in modified versions of standard sampling-based strategies [39], [40], considering the information gain. In particular, Hollinger and Sukhatme [23] presented a rapidly-exploring IG tree (RIG-tree), that was the base of a further developed two-step planning process presented by Viseras *et al.* [31]. They proposed a solution based on *Rapidly-exploring Random Trees* (RRT) and RRT* [41], using a GP for environment modeling and a DE as information function, all to find a goal position providing a high information path under a budget constraint. Moreover, evolutionary-based strategies solve the IPP problem by parametrization and optimization of a path. The authors in [25], [42] propose the use of a GP to model the environmental variable and a CMA-ES optimizer to optimize the path to be followed by the robot, with a fixed length and predefined initial and ending locations.

The literature is scarce in terms of adaptive replanning methods. Whilst the usual [14], [31] is to provide a linear execution pipeline where the vehicle is stopped for mapping and planning, some authors propose to perform the process of environmental modeling in parallel to the planning process [43], and even start planning in a further location of the current mission path to avoid stopping the vehicle [25]. Furthermore, most relevant strategies, either sampling-based [31] or evolutionary-based [25] do not transfer planning knowledge between consecutive planning iterations.

### C. CONTRIBUTIONS

This work is based on the GP modelling described in [28], yet it extends the latter by developing a novel: (a) *adaptive visual information gathering* (AVIG) framework, (b) *decision-time adaptive replanning* (DAR) behavior, and (c) *depth-first Monte Carlo tree search* (DF-MCTS) strategy for IPP. In order to deploy such adaptive information gathering in the field we propose a method that joins the advantages of graph-based and sampling-based methods. Devoting some time at the initialization to produce a network of neighbor nodes (a graph), that is used for sampling paths and grow a decision tree. The consecutive planning iterations update such decision tree and select a candidate path to execute in an anytime manner. Having a precomputed network of nodes boosts the sampling computations. This point, together with the fact of maintaining a decision tree alive between consecutive planning iterations are two key features to reduce the planning time and improve the overall performance. Moreover, the proposed planning method provides smooth paths, which are more adequate that rough paths, to be followed by the AUV.

The AVIG framework coordinates the parallel execution of four modules: navigation, data processing, map estimation and planning. Where, the planning module is designed using the novel DAR behavior of (b) for adaptive mission replanning, and integrates the novel IPP strategy of (c) that consists in a *depth-first* (DF) version of the *Monte Carlo tree search* (MCTS). The DF-MCTS is a reinforcement learning (RL) strategy guided by: a value function that considers the sampling informativeness and the recorded data density, and by a function that rewards smooth paths. The proposed framework is integrated in a ROS-based architecture to be executed in field test and prove the effectiveness of the method for autonomous exploration of underwater environments covered with *P. oceanica*.

The remainder of the paper is organized as follows: Section II introduces some background about GP models and RL algorithms. Sections III, IV and V describe the developed methods–i.e. AVIG framework, DAR behavior and DF-MCTS strategy, respectively–. Section VI describes the tests performed in simulation and in field and illustrate the results. Finally, Section VII summarizes the conclusions.

## II. PRELIMINARIES

### A. GAUSSIAN PROCESSES (GP)

This section introduces the basis of the Gaussian processes used in the map estimation processes described in Section III-C1. A GP is defined as a collection of random variables which have a joint Gaussian distribution [44]. The random variables represent the value of the non-observable function value $f(\mathbf{x})$ at location $\mathbf{x}$. Such function is specified by its mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$ values such that,

$$f(\mathbf{x}) \sim \mathcal{GP}\Big(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\Big) \tag{1}$$

Assuming a linear regression model, $f(\mathbf{x}) = \phi(\mathbf{x})^\top \mathbf{w}$, where $\phi(\mathbf{x})$ represents a basis function vector, or kernel, and the weights $\mathbf{w}$ follow a zero mean normal distribution $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_\mathbf{p})$. Since $\mathbb{E}[\mathbf{w}] = 0, f(\mathbf{x})$ results in a zero mean:

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] = \phi(\mathbf{x})^\top \mathbb{E}[\mathbf{w}] = 0 \tag{2}$$

In addition, the covariance of $f(\mathbf{x})$ is derived from using the zero mean weights assumption such that:

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \\ &= \mathbb{E}[f(\mathbf{x}) f(\mathbf{x}')] \\ &= \phi(\mathbf{x})^\top \mathbb{E}[\mathbf{w}\mathbf{w}^\top] \phi(\mathbf{x}') \\ &= \phi(\mathbf{x})^\top \Sigma_\mathbf{p} \phi(\mathbf{x}') \end{aligned} \tag{3}$$

It results in that the GP covariance directly depends on the basis function vector, the query locations and the variance in the weights.

Moreover, since there is no direct access to the function values themselves–i.e. $f(\mathbf{x})$–, and only noisy observations, we assume additive white noise $\epsilon$ with variance $\sigma_n^2$ on a latent variable modelled as $y = f(\mathbf{x}) + \epsilon$. Then according to [44] the joint distribution of the observations $\mathbf{y}$ and the latent values $\mathbf{f}_*$ according to the prior is defined as,

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right), \tag{4}$$

being $X = \{\mathbf{x_i} \mid i = 1, \ldots, n\}$ the train locations, $X_* = \{\mathbf{x_i^*} \mid i = 1, \ldots, n_*\}$ the test locations and $K(X, X_*)$ the matrix of covariances evaluated at the referenced locations.

### B. REINFORCEMENT LEARNING (RL)

This section introduces the basis of the reinforcement learning formalization used to solve the IPP problem with the DF-MCTS strategy described in Section V. The basic idea behind RL is to learn from interaction. The learning is based on trial and error; in real or simulated environments. In this work we will use RL in order to explore the space of possible paths to be executed by the AUV during its exploration mission by interacting with the environment model. The selection of the best path to follow is considered a sequential decision process. Next sections provide the basis of the problem formalization in terms of a *Markov decision process* (II-B1) and solutions for such problem (II-B2 and II-B3).

#### 1) MARKOV DECISION PROCESSES (MDP)

MDPs are a classical representation of sequential decision processes [45], and are characterized by the Markov assumption: the decisions taken only depend on the current state. Moreover, such processes are called Finite MDP (FMDP) when the states and actions are finite and Partially Observable Finite MDP (POFMDP) when the state cannot be completely observed.

MDPs are suited for RL problems, being useful to map situations to actions by maximizing a numerical reward signal; an agent interacts with the environment and gets a reward signal and a change of state. They are characterized by four main

subelements; a policy, the reward signal, the value function and optionally, a model of the environment:

- The *policy* determines the action to be taken from a particular state. It can be (1) stochastic, where a probability is specified for each action following, for instance, a Gaussian or an uniform distribution; (2) deterministic, for instance taking the action with the highest expected reward in a greedy fashion; or even (3) $\epsilon-$greedy, where a stochastic policy is used with a probability given by $\epsilon \in [0, 1]$ or greedy otherwise.
- The *reward signal* comprises the goal of the RL problem. The objective of the RL problem is to maximize the total expected reward along an executed path.
- The *state-value function* estimates how good is for the agent being in a given state. It considers the total amount of reward that the agent can expect to accumulate over the future. Whilst rewards are given directly from the environment, values are obtained from trial experience. This value function takes into account the future rewards of the states that are likely to follow the actual state.
- The *model* describes the behavior of the environment. It predicts the transitions between states given an action, which is, the next state and reward from the current state and action. Whilst *model-free* methods learn by trial and error, *model-based* methods are able to consider future possible situations before they are experienced.

### 2) ITERATIVE POLICY EVALUATION

In this work the IPP strategy described in Section V is approached as a POFMDP, in which the set of states, actions and rewards ($\mathcal{S}$, $\mathcal{A}$ and $\mathcal{R}$ respectively) have a finite number of elements, and where the state cannot be fully observed. The POFMDP is solved using a *value function* method based on *iterative policy evaluation*. In particular, we are using *Monte-Carlo (MC)* for the expansion and *Temporal difference (TD)* for the value backpropagation, methods detailed below, in equations 15 and 16. In this case the Markov assumption can be translated into a probability of resulting a particular state $s'$ and reward $r$, at a particular time $t$, given a preceding state $s$ and action $a$:

$$p(s', r|s, a) \doteq \Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}, \tag{5}$$

for all $s, s' \in \mathcal{S}$ and $r \in \mathcal{R}$, $a \in \mathcal{A}(s)$. Such probability determines the dynamics of the MDP and is characterized by $\mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A}(s) \to [0, 1]$, where

$$\sum_{s'} \sum_{r} p(s', r|s, a) = 1, \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}(s). \tag{6}$$

The *state-value function* $v_\pi(s)$ for an arbitrary policy function $\pi(s)$ is defined as,

$$v_\pi(s) \doteq \mathbb{E}_\pi\{G_t|S_t = S\} \tag{7}$$

This is the expected return on the long run (several time steps ahead). Where $G_t$ is defined as the discounted sum of expected rewards such as,

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots \tag{8}$$
$$= R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \dots\right) \tag{9}$$
$$= R_{t+1} + \gamma G_{t+1} \tag{10}$$

where $\gamma \in [0, 1]$ is the *discount rate*. With a $\gamma = 0$ the agent would be myopic, considering only immediate rewards. Whereas with a higher $\gamma$ value the agent would have a stronger consideration on further rewards. From the last expression we can start revealing the recurrent nature of RL problems.

The *state-value function* can be further expanded using (7) and (10), and it can be deduced that,

$$v_\pi(s) \doteq \mathbb{E}_\pi\{R_{t+1} + \gamma G_{t+1}|S_t = S\} \tag{11}$$
$$= \mathbb{E}_\pi\{R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = S\} \tag{12}$$
$$= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\Big[r + \gamma v_\pi(s')\Big], \tag{13}$$

where the policy $\pi(a|s)$ provides the probability of choosing a particular action $a$ from a state $s$. The Bellman equation (14) provides an iterative expression to find an optimal solution $v_*(s) \doteq \max_\pi (v_\pi(s))$ to the RL problem as $k \to \infty$. *Dynamic programming (DP)*, *Monte-Carlo (MC)* and *Temporal difference (TD)* methods can be used to perform such *iterative policy evaluation*.

$$v_{k+1}(s) \doteq \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\Big[r + \gamma v_k(s')\Big] \tag{14}$$

DP can be used to incrementally compute optimal policies when a perfect model of the environment is available. However, they require a fully observable environment, they cannot be used to solve partially observable MDP. The value update for MC and TD methods is performed as follows:

$$MC : v_{k+1}(s) \doteq v_k(s) + \lambda\Big(G_t - v_k(s)\Big) \tag{15}$$
$$TD : v_{k+1}(s) \doteq v_k(s) + \lambda\Big(r + \gamma v_k(s') - v_k(s)\Big) \tag{16}$$

where the $\lambda$ parameter defines the *learning rate*.

The basic difference between them is that whilst MC learns after trial, TD guesses from guesses, that is MC uses the expected reward value $G_t$ and *TD* uses the discounted value of the next step $v_k(s')$, it *bootstraps*. Using either MC or TD, the value of one state can be updated from the rewards obtained on successive steps following a default policy.

### 3) DECISION-TIME PLANNING

Decision-time planning methods are used to plan a series of decisions from a root state [45]. These methods use simulated experience from a model to improve a policy or a value function. Instead of solving the whole MDP, they focus on solving a sub-MDP. They plan using the MDP model to look ahead from the current state. Here, the approximate value function is obtained on leaf nodes of the current state and then propagated backwards to the current state.

Rollout algorithms [45] are a type of decision-time planning methods, based on Monte Carlo control. They sample multiple trajectories in a depth-first fashion from a root state. Each trajectory (or rollout) consists in taking successive decisions according to a given default policy until a terminal state is reached. They are used to obtain near-optimal decisions by taking random samples in a decision space and building a search tree according to the results. They are useful for AI applications with large states and actions spaces. Moreover, this kind of methods are a good fit for our applications since they are *anytime*, always provide a solution, where more computing power leads to better performance.

Monte Carlo tree search (MCTS) is a particular rollout method improved to bias the growing of a decision tree towards highest valued regions. A tree is started from a root node, and grows iteratively following the next iteration steps: (1) *Selection* of the highest valued non-exhausted and non-ending node according to some tree policy. (2) *Expansion* of the tree. Randomly select a non-taken action to connect a new node to the selected one. (3) *Simulation* of a play-out. Propagate the new node by selecting sequential actions according to a default policy until a given budget is exhausted, and compute the new node value. (4) *Backpropagation* of the node value upwards through the tree. Once the search is interrupted an action of the root node is selected according to some predefined criteria. For instance select the action that conduces to the highest valued child, the most visited child, or a weighted function between value and visits. The key parts of the MCTS algorithm are the tree policy, the default policy and the value function; which can be shaped based on the nature of the problem domain.

The limitation of MCTS for our application lies in the fact that, whereas MCTS explores the environment in a depth first manner, the decision tree tends to grow exhaustively, resulting in a shallow tree if a short planning time is given or a high discount factor is used. For this reason we propose a novel variation, the DF-MCTS, described in section V.

## III. ADAPTIVE VISUAL INFORMATION GATHERING (AVIG)

This section introduces the framework built for AVIG in benthic underwater environments, which is illustrated in Fig. 1. A classic execution of a data gathering mission following a preprogrammed path would involve the execution only of the *navigation* module, which establishes a feedback loop; it inputs different sensor measurements to compute a pose estimation, compares it with a desired pose and outputs thruster setpoints to actuate. This is the basic autonomous behavior of an AUV, follow a mission path given several sensor measurements. With the AVIG framework we propose to close another feedback loop by adding two high level modules: one for *data processing* and another for *replanning*. We propose to synthesize the information about the spatial distribution of *P. oceanica* contained in the stereo images and replan the mission path to be followed by the robot using such information. The framework has been integrated in a ROS based software architecture [46].

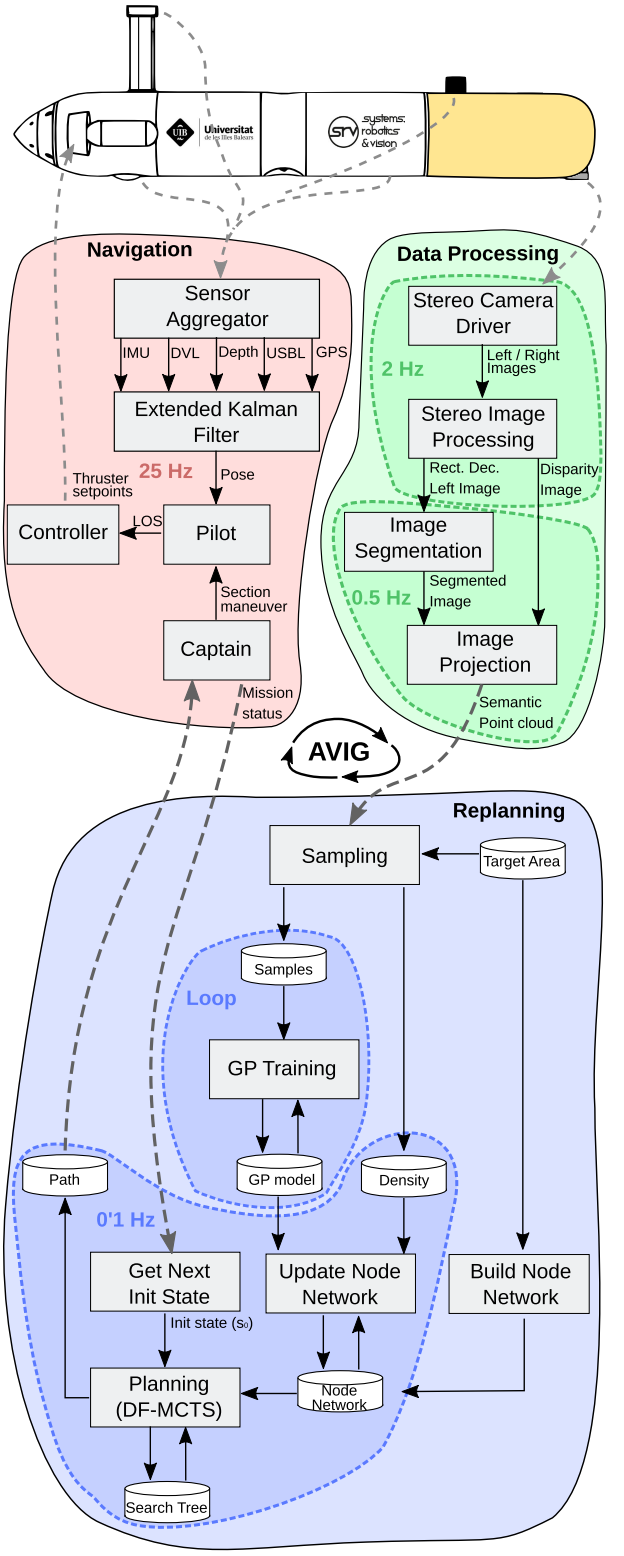



**FIGURE 1. AVIG framework scheme. The addition of the *data processing* and *replanning* modules to the AUV software architecture based on a classic *navigation* module establishes a feedback loop to guide the robot to execute informative paths considering the latest recorded data.**

### A. NAVIGATION

The navigation module governs the thruster setpoints in order to execute given mission paths, considering the navigation

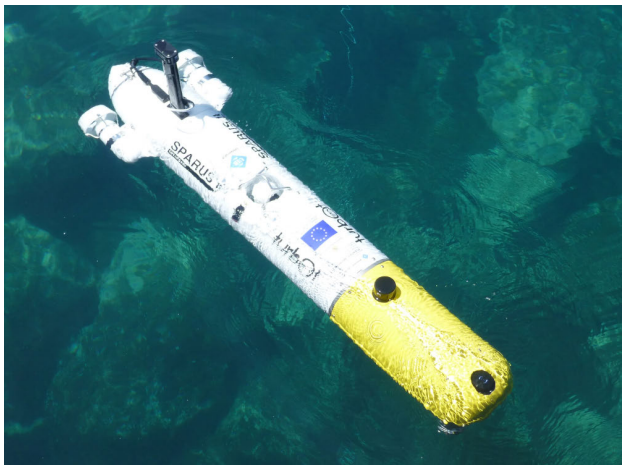


**FIGURE 2.** Turbot AUV.

sensor measurements. The processes involved in this module can be separated in two groups; localization and mission control.

#### 1) LOCALISATION

Is in charge of filtering pose measurements and generating accurate positioning. The main processes here are (1) a *sensor aggregator* node used to coordinate the publication of measurements coming from the navigation sensors, and (2) an *Extended Kalman Filter* (EKF) implementation to filter such measurements a provide a pose estimation. This localization setup is discussed in [2]. In particular, we tested the execution of the proposed framework on the Turbot AUV (see Fig. 2). Turbot is a 1.6*m* long torpedo-shaped AUV based on a Sparus II AUV [47], with three degrees of manoeuvrability; surge, heave and yaw. The EKF implementation integrates data provided by an *inertial measurement unit* (IMU), a *Doppler velocity log* (DVL), an *ultra-short baseline* (USBL), *depth* measurements and GPS (if the AUV is in the surface), and has shown to provide accurate and precise positioning.

The AVIG framework has been integrated with the ROS based COLA2 architecture [48], and can be executed in any AUV working with this open source software. In this work, we tested the execution of the proposed framework on the Turbot AUV (see Fig. 2). Turbot is a 1.6*m* long torpedo-shaped AUV based on a Sparus II AUV [47], with three degrees of manoeuvrability; surge, heave and yaw. The navigation module includes the software architecture that rules the vehicle localization, the mission control, and the data gathering.

#### 2) MISSION CONTROL

Governs the thruster power in order to follow a desired mission path. It is performed by the COLA2 control architecture described in [48], where (1) a *captain* node parses the incoming mission paths to lower level manoeuvers, that are sent to (2) a *pilot* node that provides pose and velocity reference commands to the controller using the estimated AUV pose and low level manoeuver to be followed. Then, (3) a *controller* node transforms the reference commands to thruster setpoints by following a double PID implementation.

Using COLA2 control the mission paths can be composed by waypoints or section maneuvers. Our proposed method builds the mission paths using section manoeuvres; defined by (i) an initial and final position (x,y,z), (ii) speed and (iii) tolerance. This manoeuvre follows a *Line of Sight with Cross Tracking Error* (LOSCTE) strategy [49] to produce reference velocity commands.

### B. DATA PROCESSING

The data processing module continuously generates semantic point clouds from the incoming stereo cameras data flow. The Turbot AUV carries a bottom-looking stereo camera in the front of the payload that uses a (1) a *stereo camera driver* node that is configured to provide 1936 × 1458 pixels resolution images with a framerate of 2*Hz*. Then, (2) a *stereo image processing* pipeline based on the standard ROS package found in [50] is used to rectify and downsample the images to 480 × 360 pixels, and compute the disparity between successive stereo pairs. Afterwards, (3) an *image segmentation* node performs a semantic segmentation of the downsampled images using a CNN, and (4) an *image projection* node projects the pixels of the semantic images into 3D coordinates using the disparity image.

Fig. 3 shows a sample of the downsampled images, together with the computed stereo disparity image. Even though the disparity in the seagrass areas is not dense, since we reduce the data resolution to allow online processing, the impact in the data used for map estimation is minimum.

The semantic segmentation uses the encoder-decoder based CNN architecture VGG16-FCN8 represented in Fig. 4. More specifically, it uses the model described in [51], pretrained to discriminate the *P. oceanica* from the background in underwater images. This model inputs a 480 × 360 pixels resolution RGB image and, as proposed by [28], instead of using the high resolution output of the network (output $0_1$ of Fig. 4) the last two transposed convolutional layers can be pruned, reducing the execution time at expenses of a lower output resolution of 30 × 23 pixels (output $0_3$). In any case, the network outputs a resolution grey scale image depicting the *P. oceanica* presence probability. Fig. 5 shows some examples of gathered images together with their corresponding segmented images and semantic point clouds.

The semantic point clouds are generated using the disparity image, and the reference of such point clouds to global coordinates is obtained using the robot pose, as described in [28].

### C. REPLANNING

The replanning module continuously generates informative mission paths by modeling the spatial distribution of *P. oceanica*. This module is composed by two main groups of processes, that are executed in two different threads: (1) a *map estimation* group that uses the segmented point clouds from the data processing module to train a GP model, executed in

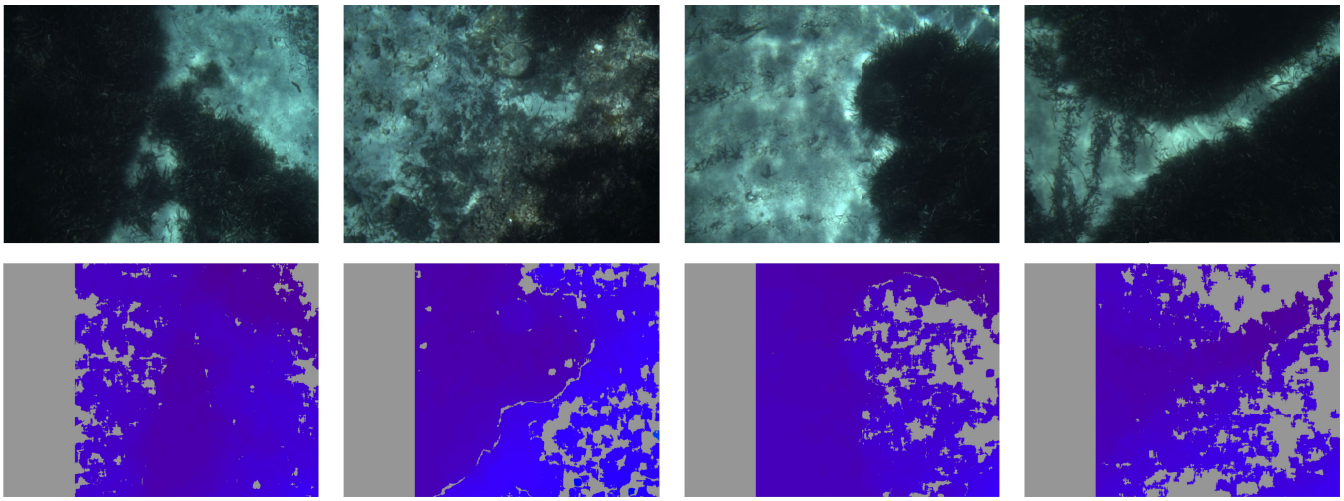


**FIGURE 3.** Example of 4 images gathered during field tests together with the corresponding disparity images.

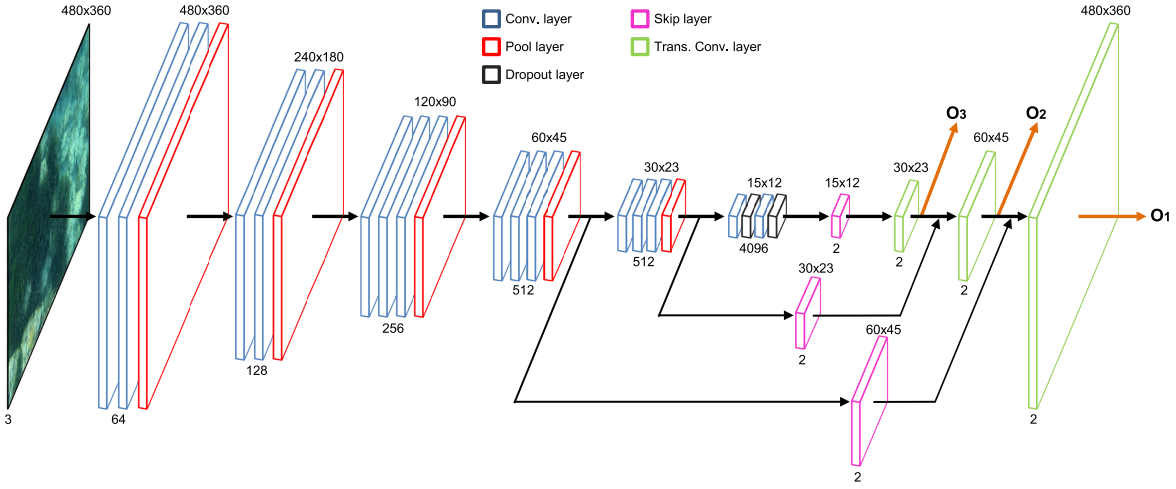



**FIGURE 4.** Scheme of the encoder-decoder based CNN architecture VGG16-FCN8 used for online image segmentation. Encoder: convolutional (blue), pooling (red) and dropout (black) layers. Decoder: score (purple) and transposed convolutional (green) layers form the decoder. The numbers under and above the layers indicate the number of feature maps and its size, respectively. The orange arrows represent the featured output points $O_1$,$O_2$ and $O_3$.

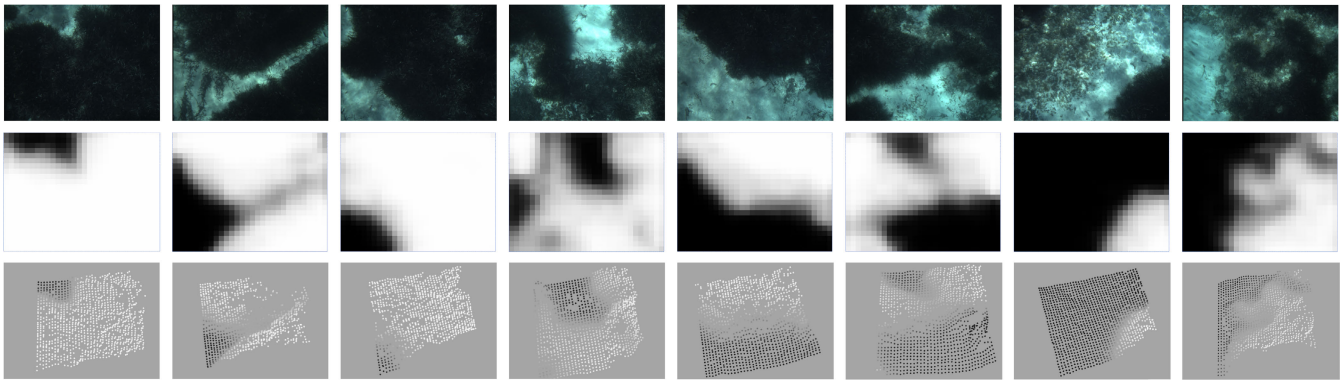


**FIGURE 5.** Example of 8 images gathered during field tests together with the corresponding segmented images and semantic point clouds.

loop, and (2) a *planning* group that uses the latest GP model to generate successive informative paths at $0.1Hz$.

The objective of the sampling process is to (i) efficiently collect the processed data in a *raw data* dataset, (ii) generate a downsampled *samples* dataset when queried by the *GP training* process, and (iii) compute the *raw data* density when queried by the *update node network* process. The first objective is attained by a thread that continuously filters the spatial distribution data by means of downsampling each segmented point cloud using a voxel filter with a $0.1m$ resolution and appended to a *raw data* dataset. The second is attained by downsampling the raw data dataset using a grid filter with a given *samples resolution* (by configuration) and returns the *samples*; such samples dataset is used to train the GP model. The third is attained by building a *k-d tree* [52] using the *raw data* dataset and a minimum leaf size of $1.0m$. Such raw data density quantifies the data recorded in a particular location and is used to compute the utility of sampling that particular spot.

#### 1) MAP ESTIMATION

Every iteration of the *map estimation* group launches a query to the sampling process to provide a samples dataset, and executes an instance of the *GP training* process to build a model of the environment. The objective of building such model is to be able to obtain predictions about the seagrass spatial distribution in punctual locations. Such predictions will be further used by the IPP planner to compute the information richness of visiting particular locations. The GP, introduced in Section II-A, are useful to build such environment models and provide predictions in non-visited locations, thanks to their uncertainty representation. Previous work presented in [28] provided an extensive study for selecting the type of GP and its configuration for modeling *P. oceanica* spatial distribution. This aforementioned work pointed out the high performance of using a *Sparse Variational GP using MCMC* (SGPMC) [53] model with a Matérn function with scaling factor $\nu = 3/2$, and a beta function to represent the GP likelihood distribution. In this work, we use the same GP model and configuration. However, in this case, a fixed *kernel lenghtscale* $\lambda = 30$ is used to train the GP. This parameter controls the correlation strength between pairs of samples depending on their distance. Fixing this hyperparameter permits to control the distance foresight of the GP model. This hyperparameter balances the exploration-exploitation trade-off; increasing $\lambda$ will result in an increased exploitation of predicted areas, while decreasing $\lambda$ will result in an increased exploration, or increased data coverage.

*Remark 1 (Limited Samples): A key aspect of the strategy designed for mapping is to use a fixed sample resolution R and induction points density IPD. This allows to get a high mapping frequency at the beginning of the exploration, reducing the number of samples for training, and a bound on the time expend for mapping as the data coverage increases. For this purpose we set a maximum number of samples $N_{samples}$ and of induction points $N_{IP}$ and compute R and IPD using the area A of the target region as follows*:

$$R = \sqrt{\frac{A}{N_{samples}}} \tag{17}$$

$$IPD = \frac{N_{IP}}{A} \tag{18}$$

#### 2) PLANNING

The planning module uses the developed DAR method described in Section IV. It iteratively executes informative paths using the novel DF-MCTS described in Section V, considering the current mission status and the most recent GP-based environment to maximize the information gain during the exploration process.

This method is executed with a fixed frequency. It first builds a network of possible sampling locations inside a target area (*build node network*), establishing connections between neighbor nodes. Then it iteratively executes (1) an *update node network* process to query the sampling process for the recorded data density at the sampling locations and to query the GP model for predictions at the same locations and update the network of sampling locations. Then, considering the actual mission status, (2) a *get next in it state* provides the starting state for the next IPP execution. Finally (3) a *planning* process based on the novel DF-MCTS described in Section V sequentially updates a search tree to retrieve an informative path that is stored to be launched as the robot finishes the current section manoeuver.

## IV. DECISION-TIME ADAPTIVE REPLANNING (DAR)

This section describes the developed strategy for decision-time adaptive replanning (DAR). This DAR behavior has been designed to be executed online while the AUV is moving; the planner iteratively command new mission paths considering the newest data available as the robot navigates and the GP-based environment model is updated. It consists in four key ideas:

- Building and updating a network of nodes $\mathcal{N}$ that results in a pre-initialization of part of the content included in the states and actions considered during decision-time planning.
- Having the AUV in constant motion, while being flexible to execute updated missions. The robot is neither stopped for planning nor forced to complete the commanded mission paths.
- Growing a search tree in a depth-first fashion, following a novel DF-MCTS strategy for decision-time planning.
- Recycling part of the last search tree for successive planning executions.

The main structure of the method is illustrated in Algorithm 1. It inputs: a target area $\mathcal{A}$ and an obstacle region $\mathcal{O}$ defined as polygon shapes in global coordinates, a model of the environment $\mathcal{M}$ containing the GP model and the *k-d tree* used for computing the raw data density, an initial state $s_0$ defined by the initial AUV pose, the sampling nodes density $\rho$ configured, and the nearest neighbor distances $d_1$ and $d_2$ set. The path to execute $\mathcal{P}$ and the remaining path $\mathcal{P}_{s_0}$ are

initialized empty. The algorithm starts generating a set of sampling nodes $\mathcal{N}$, and then iteratively: (1) updates the nodes utility using the newest environment model $\mathcal{M}$, (2) gets the next initial state $s_0$ if a path $\mathcal{P}$ has been previously generated, (3) gets a path $\mathcal{P}_{s_0,s_n}$ using the DF-MCTS strategy, and (4) saves the path.

**Algorithm 1** DAR()

**Input:** $\mathcal{A}, \mathcal{O}, \mathcal{M}, s_0, \rho, d_1, d_2$
- $\mathcal{P} \leftarrow \varnothing$; $\mathcal{P}_{s_0} \leftarrow \varnothing$;
- $\mathcal{N} \leftarrow buildNodes(\mathcal{A}, \mathcal{O}, d, d_1, d_2)$
- **while** $\neg stopCondition()$ **do**
  - $\mathcal{N} \leftarrow updateNodes(\mathcal{N}, \mathcal{M})$
  - **if** $\mathcal{P} \neq \varnothing$ **then**
    - $s_0 \leftarrow getNextInitState(\mathcal{P})$
  - $\mathcal{P}_{s_0,s_n} \leftarrow getPath(s_0)$
  - $\mathcal{P}_{s_0} \leftarrow getRemainingPath(\mathcal{P}, s_0)$
  - $\mathcal{P} \leftarrow \mathcal{P}_{s_0} + \mathcal{P}_{s_0,s_n}$
  - $savePath(\mathcal{P})$

### A. NODE NETWORK

The Algorithm 2 describes the generation of the node network $\mathcal{N}$ (of Algorithm 1). It randomly generates a set of nodes $\mathcal{N}$ inside of a given target area $\mathcal{A}$ and outside of given obstacle areas $\mathcal{O}$. The number of nodes $n$ to build is determined by a desired node density $\rho$ and the area of $\mathcal{A}$. Then, a *k-d tree* is build using the locations of the nodes contained in $\mathcal{N}$ for quick nearest-neighbor lookup. The node network is build by querying the *k-d tree* for two sets of neighbors for each node; a first set of nodes $\mathcal{N}_1$ located at a distance $d < d_1$, and a second set of nodes $\mathcal{N}_2$ located at a distance $d > d_1$ and $d < d_2$. Being the distance thresholds $d_1 < d_2$.

**Algorithm 2** buildNodes()

**Input:** $\mathcal{A}, \mathcal{O}, \mathcal{M}, \rho, d_1, d_2$
- // Get sampling locations:
- $a \leftarrow Area(\mathcal{A})$
- $n \leftarrow a \cdot \rho$
- **while** $k < n$ **do**
  - $p \leftarrow getRandomLocation(\mathcal{A})$
  - **if** $(p \notin \mathcal{O})$ **then**
    - $\mathcal{N} \leftarrow append(\mathcal{N}, p)$
    - $k \leftarrow k + 1$
- // Set neighbors:
- $kdt \leftarrow KDTree(\mathcal{N})$
- **for** $N \in \mathcal{N}$ **do**
  - $\mathcal{N}_1 \leftarrow getNearNodes(kdt, N, d_1)$
  - $\mathcal{N}_2 \leftarrow getNearNodes(kdt, N, d_2) - \mathcal{N}_1$
  - $setNodeNeighbors(N, \mathcal{N}_1, \mathcal{N}_2)$

The Algorithm 3 describes the update of the node network $\mathcal{N}$ (of Algorithm 1). The objective is updating the sampling utility at the $\mathcal{N}$ locations. The model $\mathcal{M}$ is queried to get the data density and GP prediction in the $\mathcal{N}$ locations. Then, the information gain $I$ and the utility value $U$ of the nodes in $\mathcal{N}$ are computed, that will be used to guide the IPP.

**Algorithm 3** updateNodes()

**Input:** $\mathcal{N}, \mathcal{M}$
- $D \leftarrow getDensity(\mathcal{M}, \mathcal{N})$
- $\mu, \sigma^2 \leftarrow getPrediction(\mathcal{M}, \mathcal{N})$
- $I \leftarrow computeInformation(\mu, \sigma^2)$
- $U \leftarrow computeUtility(D, I)$
- $\mathcal{N} \leftarrow D, \mu, \sigma^2, I, U$

### B. INFORMATION GAIN AND NODE UTILITY

The information gain $I$ is computed from the prediction obtained with the GP model at the node location. We have considered two options to compute such information, either using the differential entropy (DE) $I_{DE}$ or the upper confidence bound (UCB) $I_{UCB}$ functions, Equations (19) and (20), respectively.

$$I_{DE} = \frac{1}{2}\ln(2\pi e\sigma^2) \tag{19}$$

$$I_{UCB} = \mu + 1.96\sqrt{\sigma^2} \tag{20}$$

Whilst $I_{DE}$ uses the variance $\sigma^2$ provided by the GP in the query locations, $I_{UCB}$ also uses the mean semantic label $\mu$ provided by the GP. $I_{UCB}$ provides higher values to higher mean label locations, which increases exploitation (coverage) in such locations at the expenses of reducing exploration in lower mean label locations. Moreover, instead of using the $I$ directly for planning we build an utility value $U$ to leverage the $I$ with the neighboring data density $D$ such as:

$$U = I'(1 - D')^{\alpha} \tag{21}$$

The objective of using such $U$ is to attenuate the interest of visiting areas that do not present a reduction on the predicted information $I$, even though they have been repeatedly recorded (high density). This situation may happen in areas with heterogeneous data, such as meadow boundaries. Moreover, since the resolution of the GP model is bounded by $R$ value and *IPD* of Remark 1, recording more data on a high information location does not implies improving the fitness of the GP model. $I'$ and $D'$ represent the normalized information and density values for all the nodes to the range [0, 1] using a *min-max* normalization. The $\alpha$ parameter works as a weight on the density factor, the higher the $\alpha$ the bigger is the impact of the density $D$.

### C. REPLANNING

In order to allow a continuous navigation and a flexible path execution, we take into account the time expend in planning in order to set the initial state used for next planning iteration. The distance between the actual position and the next initial state $s_0$ has to be larger than a minimum distance $d_{min} = v_{max} \cdot t_{plan}$, where $v_{max}$ is the maximum speed of the robot and $t_{plan}$ is the planning time. The next initial state $s_0$ used for next planning is selected taking into account the distance

covered by the robot while the planning process is executed. As the robot finishes a section maneuver, it executes the latest planned path.

## V. DEPTH-FIRST MONTE CARLO TREE SEARCH (DF-MCTS)

This section describes the IPP strategy developed to provide mission paths to the DAR behavior. Considering the high number of possible states and actions available in our field robotics application and the necessity of an online realization, a decision-time planning method has been developed to solve the IPP problem. DF-MCTS is different from MCTS in one key aspect: it keeps all the states traversed during the rollout in the search tree. This provides a faster growth of the tree, and guarantees a solution path of a given length. The limitation of MCTS for our application is that whereas MCTS explores the environment in a depth first manner, the decision tree tends to grow exhaustively resulting in a shallow tree if a short planning time is given or a high discount factor is used.

### A. PROBLEM FORMALIZATION

We propose to solve the IPP problem using a RL-based algorithm using a finite number of non-fully observable states. For that end, a POFMDP represents the sequential decision problem.

#### 1) STATES

A *state* $S_k$ is defined by an associated *node* $N$, a parent state $S_{k-1}$, a set of action candidates $\mathcal{A}(S_k)$, a state value $V$, a distance cost $c$ from the initial state, and an orientation $\theta$; $S_k \leftarrow \{N, S_{k-1}, \mathcal{A}(S_k), V,\ c, \theta\}$.

#### 2) NODES

The nodes are used to provide a discrete representation of the possible sampling locations in the target area. A *node* $N$ is defined by a *north-east* position $x$, a raw data density value $d$, an information gain $i$, an utility value $u$, and two sets of neighbor nodes $\mathcal{N}_1^N$ and $\mathcal{N}_2^N$; $N \leftarrow \{x, d, i, u, \mathcal{N}_1^N, \mathcal{N}_2^N\}$. The nodes are initialized at the initialization of the DAR behavior and updated before each IPP execution as described in section IV-A.

#### 3) ACTIONS

The actions are directly associated to the transition to a given neighbor node. So, the selection of an action directly means the selection of a neighbor node to be visited next. Taking a particular action from a given state $S_k$ will result on the transition to the neighbor node considered in such action, and in the creation of a new state $S_{k+1}$. The set of action candidates $\mathcal{A}(S_k)$ of a given state $S_k$ will be composed by a subset of the neighbor nodes $\mathcal{N}_2^N$ of its associated *node* $N$ such as $\mathcal{A}(S_k) \in \mathcal{N}_2^N$

### B. METHOD

The structure of the proposed method is represented in Fig. 6 and includes 3 sequential steps: selection, expansion and backpropagation. It starts building a tree $\mathcal{T}$ from a root state

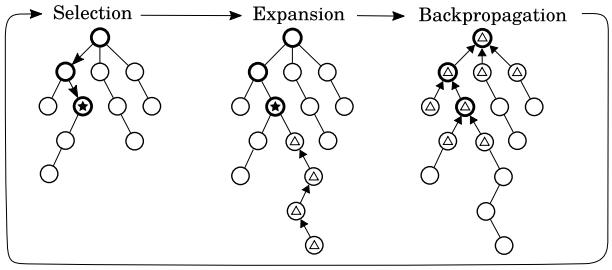



**FIGURE 6.** One iteration of the proposed tree search algorithm. The stared state represents the selected state for expansion, the states with a triangle correspond to the states used for update the tree values for the expansion and the backpropagation steps. The darker circles represent the states whose values are updated.

$s_0$ (initial state from DAR algorithm of Section IV). Then, it iteratively selects a high valued node, expands the tree with multiple rollouts, and backpropagates the state-values; selection, expansion and backpropagation.

*Remark 2 (Keep Search Tree): For successive plannings, the tree structure that hangs from the next initial state $s_0$, from now on called $\mathcal{T}'$, is conserved. And prior to the next IPP execution $\mathcal{T}'$ is traversed to update the distance cost, extend the leave states and update the state values to the newest map estimation.*

#### 1) SELECTION

Select a state from the tree $\mathcal{T}_{ne}$ to expand, where $\mathcal{T}_{ne}$ includes the non-exhausted states of $\mathcal{T}'$–i.e. state containing non-tested actions–. The state selection follows a *tree policy* based on an $\epsilon - greedy$ method (with a parametrizable epsilon), which consists in getting a random number $r$ and evaluating if $r$ is bigger or lower than epsilon. If $r > \epsilon$, the highest valued state $s^*$ from $\mathcal{T}_{ne}$ is selected. Otherwise get a random state. Being,

$$s^* = \underset{s \in \mathcal{T}}{\operatorname{argmax}}\, v(s) \tag{22}$$

#### 2) EXPANSION

Expand the tree following a given *default policy*. This step is different from the expansion step in MCTS algorithms in the following: DF-MCTS (a) performs multiple rollouts from the same selected state, and (b) the states transversed are kept in the tree and are considered for further tree expansions.

A MC rollout is build by iteratively (1) searching for candidate actions $\mathcal{A}(s)$ to the current state $s$, (2) selecting an action $a \in \mathcal{A}(s)$ according to a given *default policy* $\pi(a|s)$, (3) performing action $a$ and (4) getting next state $s'$ and reward $r$ until a distance budget $\mathcal{B}$ is exhausted.

We define the default policy $\pi(a|s)$ with an uniform distribution (23) and deterministic action-state transitions (24). We assume that the execution of action $a$ being in state $s$ results in a new state $s'$ located on a neighboring node position pointed by action $a$.

$$\pi(a|s) = \frac{1}{\|\mathcal{A}(s)\|}, \quad \text{for all } a \in \mathcal{A}(s) \tag{23}$$

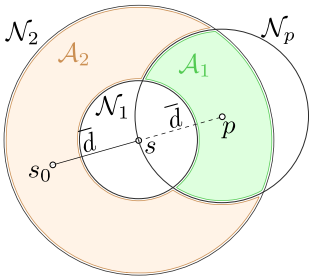



**FIGURE 7.** Two sets of action candidates for a given state $s$: $\mathcal{A}_1 = \mathcal{N}_2 \cap \mathcal{N}_p$ and $\mathcal{A}_2 = \mathcal{N}_2 - \mathcal{N}_p$. Where $\mathcal{N}_p$ includes the nodes located at a distance $d$ from $p$ and $\overline{s\,p} = \overline{s_{parent}\,s} = \overline{d}$.

$$p(s'|s,a) = 1, \quad \text{for all } s \in \mathcal{S}, a \in \mathcal{A}(s) \tag{24}$$

The set of action candidates $\mathcal{A}(s)$ of a given state $s$ is built when a state is visited for the first time. These actions include the nodes that can be reached one step ahead. The set of actions of a given state $\mathcal{A}(s)$ are build checking the node network built in the DAR initialization, see Section IV-A. In order to get smoother rollout trajectories, two sets of action candidates are used: $\mathcal{A}_1$ and $\mathcal{A}_2$, represented in Fig. 7.

The set $\mathcal{A}_1(s)$ includes the priority actions. The default policy $\pi(a|s)$ will use the set $\mathcal{A}_1(s)$ if it is not empty, otherwise it will use the set $\mathcal{A}_2(s)$. Besides, afterwards an action candidate $a$ is used, it is removed from its action candidate set. As a result from this, as the tree grows, some states get exhausted of action candidates. Such states are considered exhausted and are removed from the $\mathcal{T}_{ne}$ used for selection.

The values $v(s)$ of the states transversed during rollout are updated using the state-value function (7). That, considering (5) and (23), can be rewrite as:

$$v(s) = r(s) + \gamma v(s'), \tag{25}$$

where $r(s)$ represents the reward generated in state $s$.

In order to generate smooth trajectories the relative turns between consecutive states are considered in the reward function $r(s)$. Defined as

$$r(s) = u(s) \cdot \cos^{\omega} \frac{\theta_r(s)}{2}, \tag{26}$$

where $u(s) \in U$ is the utility value $u$ of the node belonging to state $s$ obtained from the update of the node network using equation 21, and $\theta_r(s)$ is the relative angle between the orientations of $s$ and its parent state. The second term has been designed to penalize sharp turns, rewarding small variations of $\theta_r(s)$, and is contained in the interval $[0, 1]$; the $\omega$ acts as a parameterizable weight. Since $u \in [0, 1]$ (Section IV-A) the reward will be unitary $r \in [0, 1]$. Computing the value of the rollouts is a matter of sequentially computing the discounted sum of rewards. Furthermore, as the exploration advances and more data is recorded, the map estimation gets more accurate, which impacts in a more accurate utility value and reward signal. Hence, the more accurate the GP regression becomes the more accurate the reward signal becomes.

#### 3) BACKPROPAGATION

Back-propagate the rollout values to the preceding states. We use for this a value function based on the TD expression (16) presented in Section II-B2. However instead of updating the state value using only a child state value, we use the mean value of all child states $s_c \in \mathcal{S}_c(s)$ for this value update.

$$v_{k+1}(s) = v_k(s) + \lambda \left[ r + \gamma \left\| \sum_{s_c \in \mathcal{S}_c} v_k(s_c) \right\| - v_k(s) \right] \tag{27}$$

Assuming that there will not be child outlier values is necessary and strongly supported in the fact that the child's state value accuracy is directly determined by the rewards signal accuracy, which directly depends on informativeness accuracy computed from the GP estimation, elements already discussed previously. Although the GP estimation might be very inaccurate at the beginning of the exploration due to the limited data, as they progress, GPs with the configurations used have shown to provide very smooth and valued predictions that inhibit the possibility of outliers in the child state values.

## VI. EXPERIMENTS AND RESULTS

### A. EVALUATION METRICS

The results are presented using the following evaluation metrics, which are obtained after each iteration of the map estimation module.

#### 1) MEAN OF THE DIFFERENTIAL ENTROPY (MDE)

Provides the mean map information, and is computed using the variance value of the environment model prediction $\sigma^2$ at $n$ locations inside the target area.

$$MDE = \frac{1}{n} \sum_{i=1}^{n-1} DE(i) \tag{28}$$

$$DE(i) = \frac{1}{2} \ln \left( 2\pi e \sigma^2(i) \right) \tag{29}$$

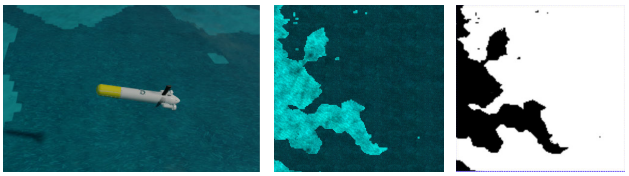
#### 2) STANDARD DEVIATION OF THE DIFFERENTIAL ENTROPY (SDE)

Provides the standard deviation of the pixelwise map information, also computed using the variance value of the environment model prediction $\sigma^2$ at $n$ locations inside the target area.

$$SDE = \sqrt{\frac{1}{n} \sum_{i=1}^{n-1} [DE(i) - MDE]^2} \tag{30}$$

#### 3) AREA UNDER THE CURVE OF THE RECEIVER OPERATING CHARACTERISTICS (AUC ROC)

This metric is used only on the simulation tests, using the simulation groundtruth GT represented in Fig. 8b. Such GT is a precise mono-chrome representation of the simulation environment since, in fact, is the source for building the



(a) (b) (c)

**FIGURE 8.** (a) Screenshot containing a Sparus II AUV in the *StoneFish* environment, the (b) synthetic seabed image containing *P. oceanica* and sand used for simulation and (c) the seabed groundtruth.

seabed textures distribution containing *P. oceanica* and sand. Given such mono-chrome image GT and a mean prediction (obtained from the last GP model) with the same shape and resolution, the ROC diagram is obtained by plotting the pixelwise true positive rate (TPR) with respect to the pixelwise false positive rate (FPR) using a varying discretization threshold for the mean GP prediction. The AUC ROC provides the area under the ROC curve.

$$TPR = \frac{TP}{TP + FN} \tag{31}$$

$$FPR = \frac{FP}{TN + FP} \tag{32}$$

where TP, FP, TN, FN represent the number of *true positives*, *false positives*, *true negatives* and *false negatives* for a given threshold value in the interval [0, 1].

#### 4) COVERAGE PERCENT (CP)

Provides a percentage measure of the recorded data extension. It is computed as the recorded area $A_r$ divided the total area $A_t$ of the target region, where the recorded area $A_r$ is obtained as the total number of samples $n_c$ contained in the target region multiplied by the squared sample resolution $R$.

$$CP = 100 \cdot \frac{A_t}{A_r} = 100 \cdot \frac{A_t}{n_c * R^2} \tag{33}$$

### B. SIMULATION

The simulations have been performed in the ROS-based simulator *StoneFish* [54] using the Turbot AUV dynamics and software architecture. Fig. 8 shows the synthetic seafloor image used, a recreation of the environment found in Illetes, a beach spot located in Palma Bay. The ground truth GT shown in Fig. 8c is a black/white image built from an original aerial picture of an area in Palma Bay. The simulated seabed shown in Fig. 8b is formed by adding texture (*P. oceanica* and sand) on this aforementioned original GT image, which means that there is no error between the black/white GT and the texture image that occupies the simulated seabed.

This section describes the results obtained using the AVIG framework with the DAR strategy, and compares the performance if instead of using the DAR strategy on the planning module we use a sampling based strategy (SBSRE*) or a predefined lawn mower (LM) pattern path.

**TABLE 1.** Planning module configuration using the DAR strategy.

| Parameter | Value |
|---|---|
| Planning time | $10s$ |
| Information function | $DE$ |
| Alpha (Utility) | 1.0 |
| Beta (Reward) | 1.0 |
| Discount factor | 0.8 |
| Learning rate | 0.9 |
| Rollouts number | 32 |
| Epsilon | 0.01 |
| Number of nodes | 20,000 |
| Neighbor distance $d_1$ | $1.5m$ |

#### 1) DAR STRATEGY

Table 1 shows the set of configuration parameters used in these simulated experiments. Moreover, in order to generalize to different target areas we propose to set by default the distance budget $\mathcal{B}$ for the rollouts to the quarter of the target area perimeter, and the neighbor distance $d_2$ to a quarter of $\mathcal{B}$.

#### 2) SAMPLING-BASED STRATEGY (SBSRE*)

We use a modified version of SBSRE sampling strategy proposed by Viseras *et al.* [31]. The original strategy executes sequentially (1) an update of the GP model hyperparameters using the recorded data, (2) a search station (SS) process based on a RRT algorithm where a path is proposed to reach the highest utility location (station) within a given distance budget, (3) an IPP process based in a RRT* algorithm which tries to get an optimized path to the previously selected station location, and (4) execution of the highest utility path (SS or IPP). We take the GP model update–i.e. step (1)–out of the loop, executed on the map estimation module as an independent thread, in order to provide a more fluent behavior and only stop the robot at the end of a path for SS and IPP. Moreover, we used the same planning time (5s for SS and 10s for IPP) and information function (DE) proposed originally and scaled the distance budget, expand distance and path resolution proportionally to the square root of the target region area, Table 2.

#### 3) PREDEFINED LAWN MOWER PATH (LM)

We programmed a lawn mower path (LM) to cover the target area with an 6m distance between transects. Such type of pattern is extensively used to get a partial representation within a reachable time budget.

#### 4) NAVIGATION

Due to the different path typologies, the three strategies used a different configuration of the LOSCTE controller, as shown in Table 3.

- The minimum speed determines the AUV speed at the sampling locations for the DAR and SBSRE* strategies, and is set to a sufficiently high value to allow a fluent movement of the AUV. For the LM case, this minimum speed is set to a value lower than the setting for the

**TABLE 2. SBRSE* scaled parameters.**

| | Original | Modified |
|---|---|---|
| Area $[m^2]$ | 18 | 1,114 |
| Budget $[m]$ | 2.0 | 15.7 |
| Path resolution $[m]$ | 0.1 | 0.8 |
| Expand distance $[m]$ | 0.1 | 0.8 |

**TABLE 3. Configuration of the LOSCTE controller used for navigation.**

| LOSCTE configuration | DAR | SBSRE* | LM |
|---|---|---|---|
| Minimum speed $[m/s]$ | 0.2 | 0.2 | 0.05 |
| Maximum speed $[m/s]$ | 0.5 | 0.3 | 0.2 |
| Lookahead distance $[m]$ | 2.0 | 2.0 | 5.0 |
| Speed transition distance $[m]$ | 3.0 | 3.0 | 3.0 |
| Velocity ratio | 0.1 | 0.1 | 0.1 |

DAR and SBSRE to ensure a proper tracking of the pre-programmed path in the extreme of the transects.

- The maximum speed was set to the speed required for sampling in the LM case in order to have homogeneous data sampling along all the path. A high maximum speed causes lower path following accuracy in the extreme of the sections if sharp turning angles exist. We set a maximum speed value higher for the DAR than for the SBSRE* for two reasons: (1) the DAR strategy provides paths smoother than SBSRE*, which are less prone to sharp turns, and (2) considering that the SBSRE* provides paths with distance between sample locations of $0.8m$ (expand distance parameter), and that the speed transition distance is fixed to $3.0m$, when going through a section in normal conditions (low cross track error) the SBSRE* wouldn't have enough distance to reach the maximum speed, in contrast the DAR provides sufficient distance between sampling points to benefit from such maximum speed.

The speed at the sampling locations also depends on the light conditions. In order to record non-blurred images the vehicle speed $s_x$ has to be adjusted accordingly to the light conditions and navigation altitude according to

$$s_x \leqslant \frac{ps_x \cdot blur_{max}}{E}, \tag{34}$$

where $E$ is the exposure time, $blur_{max}$ is the maximum blurring admitted (usually taken as 1 pixel in order to be insignificant). And $ps_x$ is the pixel size in meters in the advancing direction, defined as

$$ps_x = \frac{2 \cdot a \cdot \tan\left(0.5 \cdot FOV_x\right)}{r_x}, \tag{35}$$

where $a$ is the vehicle altitude. And $FOV_x$ and $r_x$ are the field of view and the image resolution, respectively, in the advancing direction $x$.

Fig. 9 shows the resulting speeds for the three tests during an intermediate interval of $7min$ of the simulation tests. While the DAR oscillates between $0.1m/s$ and $0.5m/s$, SBSRE* reaches sequentially $0.2m/s$ when a mission is enabled and

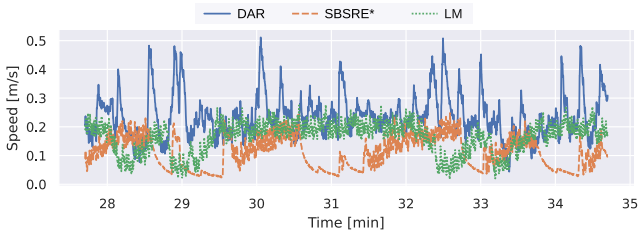



**FIGURE 9. Resulting AUV speed in the forward direction for the tree strategies tested in a *7min* time interval.**

stops when the mission is disabled, and LM reaches sequentially $0.2m/s$ while going through a transect and lowers to $0.05m/s$ at the end of the transects. The mean AUV speed in the advancing direction obtained for the entire mission tests is $0.23m/s$, $0.11m/s$ and $0.16m/s$ for the DAR, SBSRE* and LM, respectively.

### 5) DATA PROCESSING

Fig. 10 shows an example of the images gathered during simulation and the resulting segmentation using the high resolution output of the CNN. The segmentation results are very good on the boundaries and sand regions.

However, even when using the high resolution output of the CNN, some artifacts appear in some areas covered with synthetic *P. oceanica*, since this kind of images were not included in the CNN training process, and, therefore, some noise is expected to be produced at the segmentation output. In any case, the intrinsic characteristics of a GP for environment learning from semantic data points makes it a good fit for modeling a physical process under noisy input data conditions; heterogeneous data regions will produce high variance values on the GP prediction. Moreover, using the proposed adaptive replaner we aim to visit, or revisit, high variance (uncertainty) regions, leveraging exploration vs. exploitation. With exploration we refer to navigate on high variance non-visited regions, with exploitation we refer to navigate on high variance visited regions. Hence, corrupted data is handled in an effective way by means of the same mathematical and stochastic background that forms the system, firstly by the GP regression and secondly by the planner. As a consequence, the process was continued without retraining the CNN.

Furthermore, in these simulations, the pipeline for producing the samples depth was modified to use a simulated depth camera instead of the disparity of the AUV stereo images. We generated the segmented point clouds, combining the point cloud obtained from the depth camera and the segmented image. The semantic point clouds are published at a $0.290Hz$ frequency, and are completely flat due to the flat environment used and the null noise in the simulated depth image.

### 6) MAP ESTIMATION

Table 4 shows the configuration parameters used for the map estimation module. Moreover, the sampling resolution and

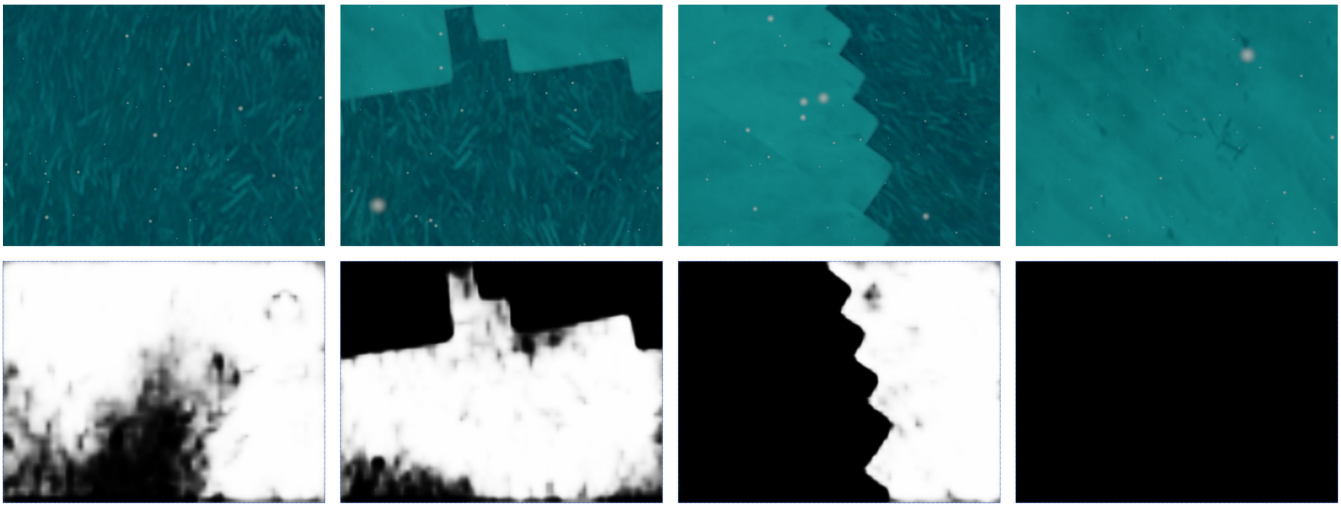


**FIGURE 10.** Images recorded during the 3D simulation tests and their corresponding high resolution segmentation image.

**TABLE 4.** Mapping module configuration.

| | |
|---|---|
| Model | SGPMC |
| Kernel | Matern 32 |
| Kernel scale | 30 |
| Likelihood | Beta |
| Likelihood Scale | 0.5 |
| Optimizer | Scipy |
| Iterations | 4,000 |
| $N_S$ | 2,000 |
| $N_{IP}$ | 200 |

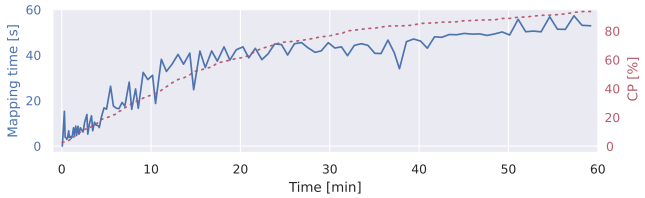



**FIGURE 11.** Resulting mapping time and coverage percent (CP) obtained during the AVIG execution in simulation using DAR algorithm.

the induction points density, computed from the target area extension and considering a maximum number of samples $N_S$ and induction points $N_{IP}$ as described in Remark 2, resulted in $0.75m$ and $0.18m^{-2}$ respectively.

Fig. 11 shows the resulting mapping times obtained from an execution of the AVIG framework using the DAR planning strategy. Such mapping time include the sampling time and the learning of the GP hyperparameters. Considering that the latter is the higher contributor and that it is proportional to the number of training samples and inducing points, the mapping time is proportional to the data coverage and not to the gathered data. Hence, the mapping time will converge as the data coverage converges, using the maximum number of samples and number of inducing points.

In this case, the mapping time seems to converge around $55s$. The fact that, at the beginning of the exploration the mapping frequency is higher, allowing fast adaptation of the IPP planner to the changing map estimation, is a good consequence of the behaviour described above. Since at the last part of the exploration the map is partially known, and the high information areas are already located, a low frequency of the map estimation module can be tolerated. However, even at the end of the exploration we require a bound on the map estimation time in order to integrate newer data into the belief within a reachable time.

### 7) PLANNING

The rollout distance budget $\mathcal{B}$ and the neighboring distance $d_2$ were computed using the target area extension, as considered at the beginning of this section. And resulted in $33.8m$ and $8.5m$, respectively. The Fig. 12, 13 and 14 represent the results obtained for the simulation tests.

Fig. 13 represents the executed paths for the three tests (DAR, SBSRE* and LM); the green shade represents the raw data density, which is normalized for each test. Since the frequency of the data acquisition is equal for the three tests, the quantity of data gathered during the same time interval is also equal. However, the distribution of such data differs. The raw data density distribution using DAR shows that a large part of the meadow boundary presents the highest data density, and that the lowest density regions belong to homogeneous data regions. The high density region on the left corresponds to an area where noisy data was acquired as we will see further in the analysis. The SBSRE* raw data density distribution is irregular, whilst it shows high density values in some boundary regions, it does not visit some others, and over records some regions due to being stopped while planning the next path. Using a predefined LM patter results in evenly distributed data on the transects, and higher density spots on the section limits.

Whilst the LM tests provide more data density in the extreme of the section maneuvers due to a slow vehicle speed required for precise path following, the DAR and

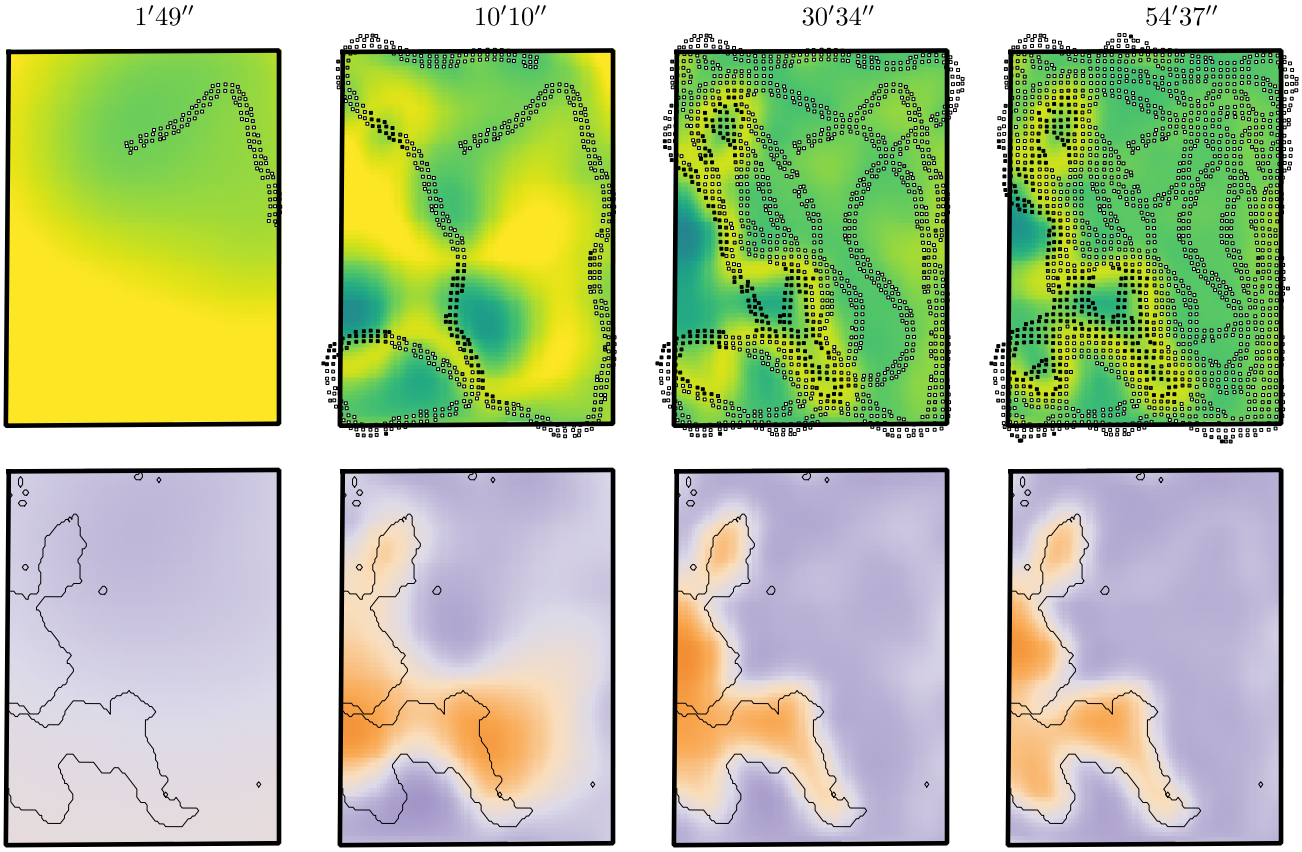



**FIGURE 12.** Representation of four different instants during the execution of the DAR algorithm in simulation. In the top row, the square markers represent the samples (black squares for background, white squares for *P. oceanica* detection) with resolution *R* used to train the GP; and the color-map represent the predicted variance map, blue for low variance regions and yellow for high variance regions. In the bottom row the black line represents the contour of the segrass meadow generated from the groundtruth image, and the color-map represent the predicted map; purple for *P. oceanica* and orange for background.

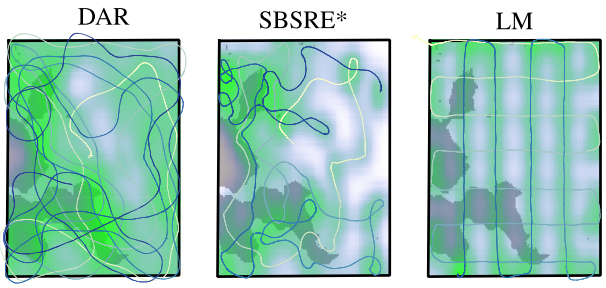



**FIGURE 13.** Paths followed and raw data density obtained during the execution of the 3D simulation tests; from left to right, following DAR, SBSRE and LM strategies. The color in the path represents time, starts in yellow and ends in dark blue. The raw data density is represented with the green shade. And the background represents the groundtruth, the white area contains *P.oceanica*.

SBSRE* tests provide higher data density in high uncertainty areas, such as meadow bounds or heterogeneous data regions, being the former faster and more fluid visiting high uncertainty areas. In contrast with DAR, that penalizes path turns in the DF-MCTS planner, the SBSRE* results in complex paths, since the path turns are not considered during planning.

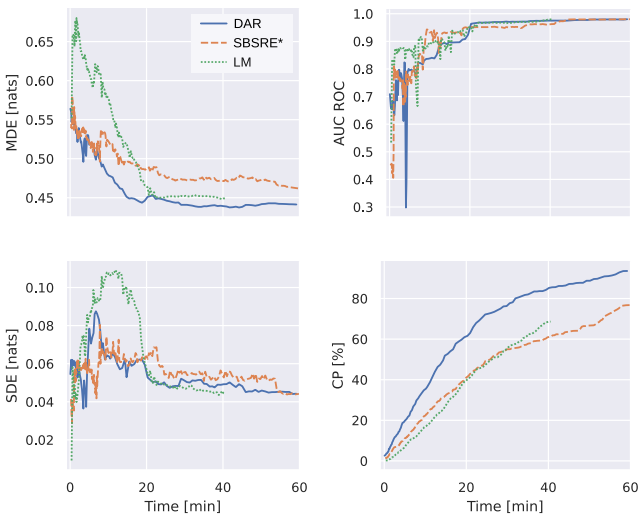



**FIGURE 14.** Results obtained using the DAR, SBSRE* and LM in simulation.

Fig. 14 presents the results obtained during the three tests in terms of the MDE, SDE, AUC ROC and CP metrics described at the beginning of the section. The DAR strategy presents

the fastest reduction of the MDE metric and a low SDE, which means that the AUV efficiently retrieves data from high uncertainty regions and reduces the entropy of the predicted seagrass distribution. The SBSRE* provides a fast reduction of the MDE at the beginning of the mission. At the beginning it reduces the entropy by targeting the regions with highest predicted variance values (the SDE maintains low values). However, approximately after the 10th minute, the MDE reduction rate gets slower. The LM results in a final value of the MDE lower than the SBSRE*. However, during the mission execution, it achieves the highest MDE and SDE values, which means that generated predictions provide high uncertainty. In terms of CP, the DAR strategy provides the best results. Thanks to the smoothness of the informative paths computed, the robot is able to achieve higher speeds that result in an increased CP. At the beginning of the exploration the AUC ROC metric is very unstable due to that the mapping process has very limited data for training. In particular, at the instant 4'25min of DAR test the AUC ROC of the estimated map shows a large negative peak. As more data is recorded the AUC ROC improves and stabilizes at minutes 20′, 22′, and 33′ for the DAR, LM and SBSRE*, respectively. Which means that, from those points the map estimation module has sufficient data to learn a proper model.

Moreover, Fig. 12 represents the acquired data and the online estimated maps at four different time instants. As the AUV navigates the target area, the map estimation gets more accurate, starting by a naive prediction at the time instant 1′49″ to a more confident prediction at 45′37″, where only the boundary regions present high uncertainty value (yellow on the top row), and the *P. oceanica* distribution (purple on the bottom row) is predicted with high precision, given the mapping resolution used. Notice how the right part of the map presents noisy samples (false negative samples) that increase the uncertainty value and attracts the robot navigation to gather more data, path in Fig. 13.

### C. FIELD TESTS

This section describes the field tests performed to evaluate the proposed exploration framework and the discussion of the results. The objective of the tests was to obtain a fast representation of a large environment contained in the target area, delimited by a yellow line in Fig. 15.

The experiments have been carried out at the south of Mallorca Island. The target area has an extension of $10,212m^2$, and was located in a shallow water region with $[3.5, 5.5]m$ depth to assess the mapping performance by comparing the online estimated maps with the existing aerial images of the area. We set a time budget for the exploration of $90min$.

#### 1) NAVIGATION

Since the experiments were carried out with the AUV navigating on surface, a precise localization was assumed, using GPS measurements which bound the navigation error of the EKF localization filter. Regarding the mission control, a default configuration for the LOSCTE was used with minimum and maximum speeds of $0.1m/s$ and $0.6m/s$, respectively, look-ahead distance of $4m$, speed transition distance of $3m/s$ and velocity ratio of $0.1m/s$. This configuration resulted in a mean AUV speed in the advancing direction of $0.27m/s$.


**FIGURE 15.** Field test target area ($10,212m^2$) on top of an aerial image from the *Instituto Geodesico Nacional* (PNOA 2018 campaign 39.326284, 2.986278).

#### 2) DATA PROCESSING

Fig. 16 shows an example of the kind of images acquired together with their segmentation output. They show a good segmentation performance using the low resolution output of the CNN. The image segmentation achieved a frequency of 0.461 $Hz$, providing sufficient overlap between successive semantic point clouds, which is illustrated in Fig. 17.

#### 3) MAP ESTIMATION

The map estimation module was configured with the same settings as the simulation tests, Table 4. These settings provided a sample resolution $R$ of $2.26m$ and an $IPD$ of $0.02m^{-2}$. Fig. 18 shows part of the grid samples processed for map estimation during the test. And Fig. 19 illustrates the resulting mapping time and the coverage percent. The figure shows a correlation between coverage and mapping time, resulting in $40s$ of mapping time after $90min$ of mission, with a CP around 60%.

#### 4) PLANNING

The planning module was configured as in simulation tests (Table 1). The distance budget $B$ and the maximum neighbor distance $d_2$ were automatically set using the perimeter of the target area as considered for simulation, resulting in $92m$ and $23m$ respectively. Considering that the distance to maximum speed configured in the LOSCTE controller is inferior to $d_2$, such neighbor distance allows the AUV to reach maximum speed between sampling locations. The Fig. 20, 21 and 22 represent the results obtained during the execution of the AVIG framework.

Fig. 21 represents the entire executed AUV exploration path. The color in the path represents time, starts in yellow and ends in dark blue. The figure also represents the raw data density with a green shade, and the hand-labeled contours of the seagrass meadow interpreted from an aerial image.

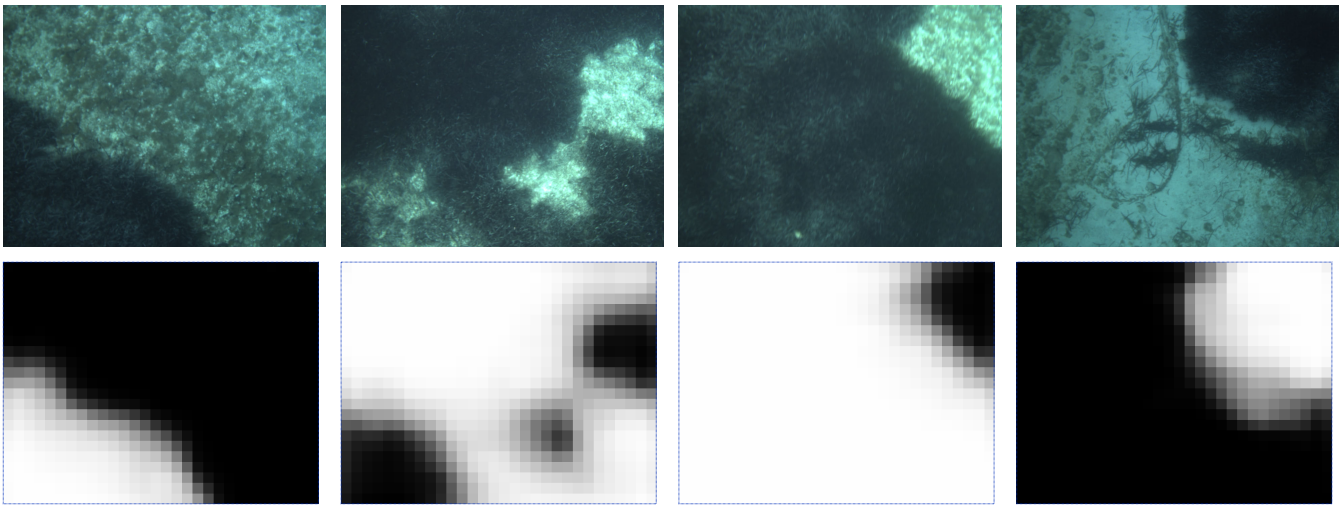


**FIGURE 16.** Field test image examples and their corresponding low resolution segmentation output.

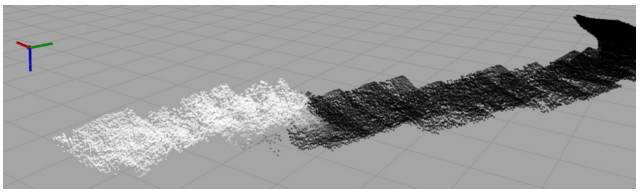


**FIGURE 17.** Sequential semantic point clouds projected to global coordinates; white pixels represent areas covered with *P. oceanica*, and black pixels represent background areas.

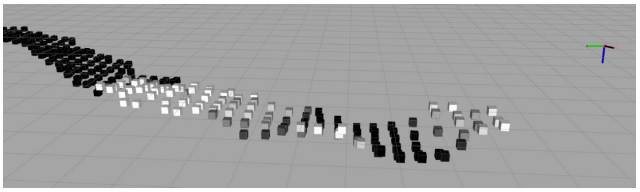


**FIGURE 18.** Samples used for online environment learning; white cubes represent areas covered with *P. oceanica*, and black cubes represent background areas.

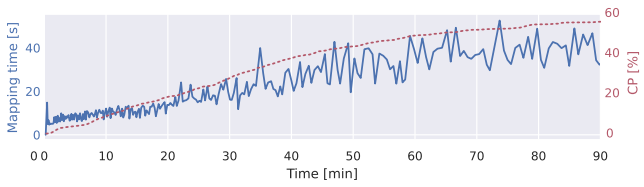



**FIGURE 19.** Resulting mapping time and coverage percent obtained during field testing the AVIG framework.

Moreover, Fig. 20 shows five representations of the gathered data and the estimated map in different time instants during the exploration. The top row of the figure shows as square markers the samples used to train the GP and the predicted variance map with a color-map, dark blue for low variance regions and yellow for high variance regions. The bottom row represents the predicted seagrass distribution with a color-map where purple indicates presence of *P. oceanica* and orange indicates background.

Finally, Fig. 22 represents the MDE and SDE of the online estimated maps along the path. Such metrics have been obtained during the exploration for every mapping module iteration.

The results provided show that, as the AUV gathers more data, it is able to generate a more accurate model of the environment–i.e., more accurate seagrass distribution in terms of variance and predicted value–, which results in an improved fitness of the computed paths, that focus the AUV navigation in areas with high uncertainty. In this test the navigation is heavily focused on the seagrass meadow because it presents an heterogeneous distribution containing small patches of background. In contrast, the data collected outside of the meadow contain homogeneously sensed background, resulting in low uncertainty.

The likelihood function used for GP modelling takes the credit of such high variance in heterogeneous data regions. The beta function used for the likelihood representation provides higher beliefs for extreme data samples; either *P. oceanica* or background. However, since the resolution size used for sampling is too large to capture the data complexity in heterogeneous data regions, such regions produce intermediate sample values that result in high uncertainty. This is a key result that enables keeping high entropy values in meadow boundaries and patchy regions to reinforce the data gathering in such areas.

The MDE and SDE show a fast convergence during the initial 30*min*, that corresponds to the results illustrated in Fig.20; the AUV performs a fast exploration of the target area up to that time increasing the map informativeness. Then, the exploration framework exploits the environment knowledge by gathering data mostly in heterogeneous data regions. Whilst the AUV gathers more data in such regions, the use of

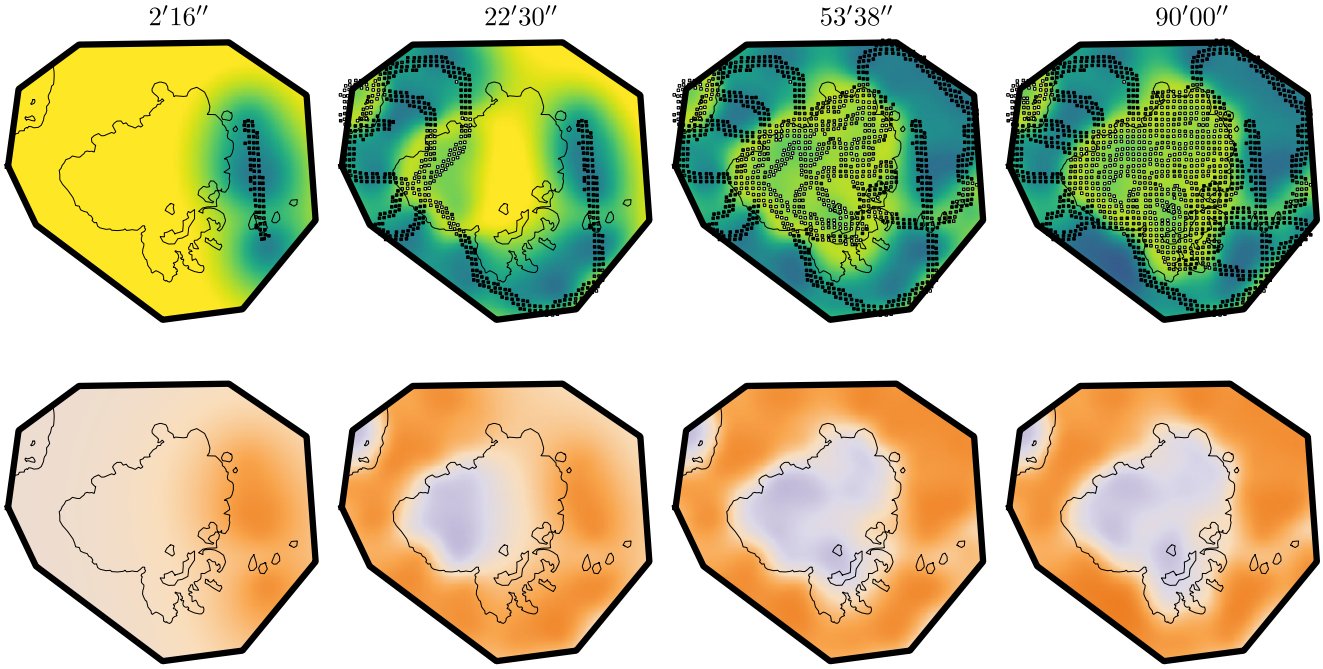



**FIGURE 20.** Representation of five different instants during the execution of the DAR algorithm in field test. In the top row, the square markers represent the samples (black squares for background, white squares for *P. oceanica* detection) used to train the GP and the color-map represent the predicted variance map; blue for low variance regions and yellow for high variance regions. In the bottom row the black line represents the contour of the seagrass meadow obtained from the groundtruth image, and the color-map represent the online predicted map; purple for *P. oceanica* and orange for background.

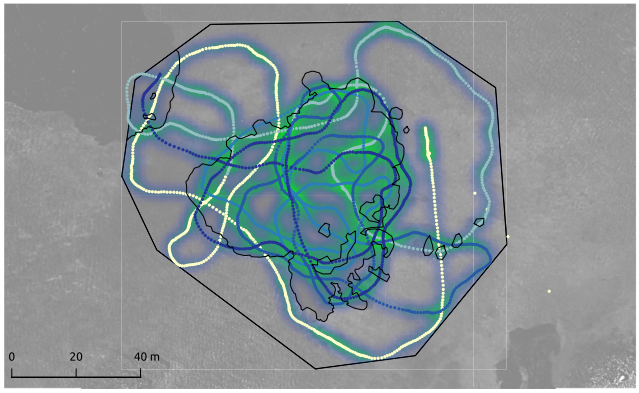



**FIGURE 21.** Path followed and raw data density obtained in field test following DAR strategy. The color in the path represents time, starts in yellow and ends in dark blue. The raw data density is represented with the green shade.

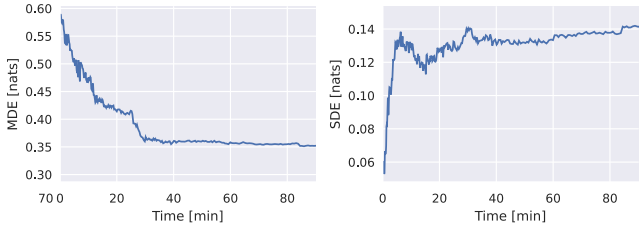



**FIGURE 22.** Results obtained in field test.

a fixed resolution size of the samples used for training bound the MDE reduction.

## VII. CONCLUSION AND FUTURE WORK

This work presents the novel design of a adaptive visual IG framework (AVIG) that integrates a novel decision-time adaptive replanning behavior (DAR). Such DAR behavior is coupled with a DF-MCTS strategy for IPP, and joins two advantages of graph-based and sampling-based methods: (a) initializes a node network to set neighbor relations between sampling locations (which reduces computation during online execution), and (b) samples paths in the node network through tree search following a decision-time strategy (that provides near-optimal solutions in an anytime manner). The AVIG framework has been successfully integrated in a ROS-based software architecture and tested in simulation and in field. The results show that using this framework the vehicle is driven to cover the region with higher uncertainty given by the GP model, whilst producing smooth paths easy to follow. Regions with increased GP variance imply that the gathered information is highly uncertain or incomplete thus it need to be revisited. The DAR behavior with DF-MCTS provided the fastest MDE reduction when compared with the very relevant strategy proposed by Viseras *et al.* [31] and with a predefined lawn mower pattern mission. The results show that the variable distance between sampling nodes and the penalization of sharp turns in the reward function provide smooth paths that are secure to be followed by the LOSCTE controller.

Additional future work will look to (i) integrate prior environmental knowledge (if existing) to the map estimation module by fusing a GP model trained with data from previous IG

missions, and (ii) test the AVIG framework on deeper regions using USBL positioning and acoustic communications.

## REFERENCES

[1] L. Paull, S. Saeedi, M. Seto, and H. Li, "AUV navigation and localization: A review," *IEEE J. Ocean. Eng.*, vol. 39, no. 1, pp. 131–149, Jan. 2014.

[2] E. Guerrero-Font, F. Bonin-Font, P.-L. Negre-Carrasco, M. Massot-Campos, and G. Oliver-Codina, "USBL integration and assessment in a multisensor navigation approach for AUVs," *IFAC-Papers Line*, vol. 50, no. 1, pp. 7905–7910, Jul. 2017.

[3] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual SLAM algorithms: A survey from 2010 to 2016," *IPSJ Trans. Comput. Vis. Appl.*, vol. 9, no. 1, pp. 1–11, Dec. 2017.

[4] J. Li, M. Kaess, R. M. Eustice, and M. Johnson-Roberson, "Pose-graph SLAM using forward-looking sonar," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2330–2337, Jul. 2018.

[5] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, Apr. 2017.

[6] L. Telesca, A. Belluscio, A. Criscoli, G. Ardizzone, E. T. Apostolaki, S. Fraschetti, M. Gristina, L. Knittweis, C. S. Martin, G. Pergent, A. Alagna, F. Badalamenti, G. Garofalo, V. Gerakaris, M. L. Pace, C. Pergent-Martini, and M. Salomidi, "Seagrass meadows (Posidonia oceanica) distribution and trajectories of change," *Sci. Rep.*, vol. 5, no. 1, Dec. 2015, Art. no. 12505.

[7] A. Abadie, M. Pace, S. Gobert, and J. A. Borg, "Seascape ecology in posidonia oceanica seagrass meadows: Linking structure and ecological processes for management," *Ecol. Indicators*, vol. 87, pp. 1–13, Apr. 2018.

[8] J. Binney and G. S. Sukhatme, "Branch and bound for informative path planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 2147–2154.

[9] A. Candela, D. Thompson, E. N. Dobrea, and D. Wettergreen, "Planetary robotic exploration driven by science hypotheses for geologic mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 3811–3818.

[10] N. R. J. Lawrance, J. J. Chung, and G. A. Hollinger, "Fast marching adaptive sampling," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 696–703, Apr. 2017.

[11] G. Hitz, A. Gotovos, F. Pomerleau, M.-E. Garneau, C. Pradalier, A. Krause, and R. Y. Siegwart, "Fully autonomous focused exploration for robotic environmental monitoring," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 2658–2664.

[12] R. N. Smith, P. Cooksey, F. Py, G. S. Sukhatme, and K. Rajan, "Adaptive path planning for tracking ocean fronts with an autonomous underwater vehicle," in *Proc. Int. Symp. Exp. Robot.*, vol. 109, 2016, pp. 761–775.

[13] L. Heng, A. Gotovos, A. Krause, and M. Pollefeys, "Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 1071–1078.

[14] V. Karolj, A. Viseras, L. Merino, and D. Shutin, "An integrated strategy for autonomous exploration of spatial processes in unknown environments," *Sensors*, vol. 20, no. 13, pp. 1–27, 2020.

[15] M. G. Jadidi, J. V. Miro, G. Dissanayake, M. G. Jadidi, J. V. Miro, and G. Dissanayake, "Gaussian processes autonomous mapping and exploration for range-sensing mobile robots," *Auto. Robots*, vol. 42, no. 2, pp. 1–18, 2017.

[16] M. Ghaffari Jadidi, J. Valls Miro, and G. Dissanayake, "Sampling-based incremental information gathering with applications to robotic exploration and environmental monitoring," *Int. J. Robot. Res.*, vol. 38, no. 6, pp. 658–685, May 2019.

[17] D. Rao, A. Bender, S. B. Williams, and O. Pizarro, "Multimodal information-theoretic measures for autonomous exploration," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 4230–4237.

[18] S. Kodgule, A. Candela, and D. Wettergreen, "Non-myopic planetary exploration combining *in situ* and remote measurements," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 536–543.

[19] T. Regev and V. Indelman, "Decentralized multi-robot belief space planning in unknown environments via identification and efficient re-evaluation of impacted paths," *Autonom. Robot.*, vol. 42, no. 14, pp. 691–713, Jul. 2017.

[20] A. Viseras, Z. Xu, and L. Merino, "Distributed multi-robot information gathering under spatio-temporal inter-robot constraints," *Sensors*, vol. 20, no. 2, pp. 1–25, 2020.

[21] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-MCTS: Decentralized planning for multi-robot active perception," *Int. J. Robot. Res.*, vol. 38, nos. 2–3, pp. 316–337, 2019.

[22] J. A. Caley and G. A. Hollinger, "Data-driven comparison of spatio-temporal monitoring techniques," in *Proc. MTS/IEEE Washington*, Oct. 2016, pp. 1–7.

[23] G. A. Hollinger and G. S. Sukhatme, "Sampling-based robotic information gathering algorithms," *Int. J. Robot. Res.*, vol. 33, no. 9, pp. 1271–1287, Aug. 2014.

[24] K. Ma, L. Liu, H. K. Heidarsson, and G. S. Sukhatme, "Data-driven learning and planning for environmental sampling," *J. Field Robot.*, vol. 35, no. 5, pp. 643–661, Aug. 2018.

[25] G. Hitz, E. Galceran, M.-È. Garneau, F. Pomerleau, and R. Siegwart, "Adaptive continuous-space informative path planning for online environmental monitoring," *J. Field Robot.*, vol. 34, no. 8, pp. 1427–1449, Dec. 2017.

[26] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning* (Adaptive Computation and Machine Learning). Cambridge, MA, USA: MIT Press, 2005.

[27] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies," *J. Mach. Learn. Res.*, vol. 9, no. 2, pp. 235–284, 2008.

[28] E. Guerrero-Font, F. Bonin-Font, M. Martin-Abadal, Y. Gonzalez-Cid, and G. Oliver-Codina, "Sparse Gaussian process for online seagrass semantic mapping," *Expert Syst. Appl.*, vol. 170, May 2021, Art. no. 114478.

[29] J. Das, F. Py, J. B. J. Harvey, J. P. Ryan, A. Gellene, R. Graham, D. A. Caron, K. Rajan, and G. S. Sukhatme, "Data-driven robotic sampling for marine ecosystem monitoring," *Int. J. Robot. Res.*, vol. 34, no. 12, pp. 1435–1452, Oct. 2015.

[30] G. A. Hollinger, A. A. Pereira, and G. S. Sukhatme, "Learning uncertainty models for reliable operation of autonomous underwater vehicles," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 5593–5599.

[31] A. Viseras, D. Shutin, and L. Merino, "Robotic active information gathering for spatial field reconstruction with rapidly-exploring random trees and online learning of Gaussian processes," *Sensors*, vol. 19, no. 5, p. 1016, Feb. 2019.

[32] A. Krause and C. Guestrin, "Submodularity and its applications in optimized information gathering," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 4, pp. 1–20, Jul. 2011.

[33] R. Marchant and F. Ramos, "Bayesian optimisation for intelligent environmental monitoring," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 2242–2249.

[34] O. M. Cliff, R. Fitch, S. Sukkarieh, D. L. Saunders, and R. Heinsohn, "Online localization of radio-tagged wildlife with an autonomous aerial robot system," *Robot., Sci. Syst.*, vol. 11, pp. 1–9, Jan. 2015.

[35] D. R. Thompson, D. S. Wettergreen, and F. J. C. Peralta, "Autonomous science during large-scale robotic survey," *J. Field Robot.*, vol. 28, no. 4, pp. 542–564, Jun. 2011.

[36] K. Yang, S. K. Gan, and S. Sukkarieh, "A Gaussian process-based RRT planner for the exploration of an unknown and cluttered environment with a UAV," *Adv. Robot.*, vol. 27, no. 6, pp. 431–443, Feb. 2013.

[37] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, and J. Nieto, "An efficient sampling-based method for online informative path planning in unknown environments," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1500–1507, Apr. 2020.

[38] C. Xiong, H. Zhou, D. Lu, Z. Zeng, L. Lian, and C. Yu, "Rapidly-exploring adaptive sampling Tree*: A sample-based path-planning algorithm for unmanned marine vehicles information gathering in variable ocean environments," *Sensors*, vol. 20, no. 9, p. 2515, Apr. 2020.

[39] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[40] S. M. LaValle, *Planning Algorithms*, vol. 9, no. 48. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[41] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[42] M. Popović, T. Vidal-Calleja, G. Hitz, J. Chung, I. Sa, R. Siegwart, and J. Nieto, "An informative path planning framework for UAV-based terrain monitoring," *Auto. Robots*, vol. 44, pp. 889–911, Feb. 2020.

[43] P. Stankiewicz, Y. T. Tan, and M. Kobilarov, "Adaptive sampling with an autonomous underwater vehicle in static marine environments," *J. Field Robot.*, vol. 38, no. 4, pp. 572–597, Jun. 2021.

[44] M. Seeger, "Gaussian processes for machine learning," *Int. J. Neural Syst.*, vol. 14, no. 2, pp. 69–106, 2004.

[45] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[46] Stanford Artificial Intelligence Laboratory. *Robotic Operating System*. Accessed: Feb. 2020. [Online]. Available: https://www.ros.org
[47] M. Carreras, C. Candela, D. Ribas, N. Palomeras, L. Magí, A. Mallios, E. Vidal, E. Vidal, and P. Ridao, "Testing SPARUS II AUV, an open platform for industrial, scientific and academic applications," *Instrum. Viewpoint*, no. 18, pp. 54–55, Nov. 2015.
[48] N. Palomeras, A. El-Fakdi, M. Carreras, and P. Ridao, "COLA2: A Control Architecture for AUVs," *IEEE J. Ocean. Eng.*, vol. 37, no. 4, pp. 695–716, Oct. 2012.
[49] T. I. Fossen and K. Y. Pettersen, "On uniform semiglobal exponential stability (USGES) of proportional line-of-sight guidance laws," *Automatica*, vol. 50, no. 11, pp. 2912–2917, Nov. 2014.
[50] *Stereo Image Processing*. Accessed: Feb. 2020. [Online]. Available: http://http://wiki.ros.org/stereo_image_proc
[51] M. Martin-Abadal, E. Guerrero-Font, F. Bonin-Font, and Y. Gonzalez-Cid, "Deep semantic segmentation in an AUV for online posidonia oceanica meadows identification," *IEEE Access*, vol. 6, pp. 60956–60967, 2018.
[52] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.
[53] J. Hensman, A. G. D. G. Matthews, M. Filippone, and Z. Ghahramani, "MCMC for variationally sparse Gaussian processes," 2015, *arXiv:1506.04000*. [Online]. Available: https://arxiv.org/abs/1506.04000
[54] P. Cieslak, "Stonefish: An advanced open-source simulation tool designed for marine robotics, with a ROS interface," in *Proc. Marseille*, Jun. 2019, pp. 1–6.

**ERIC GUERRERO** received the B.Sc. degree in mechanical engineering and the M.Sc. degree in robotics and automatic control from the Polytechnic University of Catalonia (BarcelonaTech). He is currently pursuing the Ph.D. degree in intelligent and adaptive methods for characterization of marine environments with the University of Balearic Islands. His research interests include underwater robotics, machine learning, and computer vision, and in research topics, such as intelligent control architectures and autonomous navigation.

**FRANCISCO BONIN-FONT** received the degree in telecommunications engineering from the Polytechnic University of Catalonia, Barcelona, in 1996, and the Ph.D. degree in computer engineering from the University of the Balearic Islands, in 2012. He was with the industry of information technology services addressed to bank business for ten years, before he initiated his academic activities. He has participated as a Technician and a Researcher in ten projects funded by the Spanish Scientific Council and the European Commission. He is also a Senior Lecturer with the Department of Mathematics and Computer Science, University of the Balearic Islands. He has authored or coauthored over 50 papers, among journals, book chapters, and conference proceedings in the field of image processing and underwater robotics, during his research activities with the Systems, Robotics and Vision Group, University of the Balearic Islands.

**GABRIEL OLIVER** received the degree in physics from the Universitat Autònoma de Barcelona, and the Ph.D. degree in computer science from the Universitat Politècnica de Catalunya. He is currently a Professor with the Department of Mathematics and Computer Science, Universitat de les Illes Balears, where he is the Director of the Systems, Robotics and Computer Vision Group (SRV). He has been the responsible of research projects granted by the European Commission, the Spanish Government, and the Regional Administration, mainly related to underwater robotics. His major research interests include marine robotics, real-time vision, and control architectures for autonomous mobile robots. Since 2017, he has been a member of the IFAC Marine Systems Technical Committee. He has been the Spanish Chapter Chair of the IEEE Oceanic Engineering Society, from 2007 to 2014.