Received: 1 November 2015 | Revised: 28 February 2017 | Accepted: 11 March 2017

DOI: 10.1002/rob.21722

REGULAR ARTICLE

WILEY

Adaptive continuous-space informative path planning for online environmental monitoring

Gregory Hitz[1] | Enric Galceran[1] | Marie-Ève Garneau[2] | François Pomerleau[3] | Roland Siegwart[1]

[1]Autonomous Systems Lab, ETH Zurich, 8092, Zurich, Switzerland

[2]Limnological Station, University of Zurich, 8802, Kilchberg, Switzerland

[3]Autonomous Space Robotics Lab, University of Toronto, Toronto, M3H 5T6, Canada

Correspondence
Gregory Hitz, Autonomous Systems Lab, ETH Zurich, 8092 Zurich, Switzerland.
Email: hitzg@ethz.ch

Funding information
Natural Sciences and Engineering Research Council of Canada (NSERC)
Swiss National Science Fund
CR2212–130023

Abstract

Autonomous mobile robots are increasingly employed to take measurements for environmental monitoring, but planning informative, measurement-rich paths through large three-dimensional environments is still challenging. Designing such paths, known as the informative path planning (IPP) problem, has been shown to be NP-hard. Existing algorithms focus on providing guarantees on suboptimal solutions, but do not scale well to large problems. In this paper, we introduce a novel IPP algorithm that uses an evolutionary strategy to optimize a parameterized path in continuous space, which is subject to various constraints regarding path budgets and motion capabilities of an autonomous mobile robot. Moreover, we introduce a replanning scheme to adapt the planned paths according to the measurements taken *in situ* during data collection. When compared to two state-of-the-art solutions, our method provides competitive results at significantly lower computation times and memory requirements. The proposed replanning scheme enables to build models with up to 25% lower uncertainty within an initially unknown area of interest. Besides presenting theoretical results, we tailored the proposed algorithms for data collection using an autonomous surface vessel for an ecological study, during which the method was validated through three field deployments on Lake Zurich, Switzerland. Spatiotemporal variations are shown over a period of three months and in an area of 350 m × 350 m × 13 m. Whereas our theoretical solution can be applied to multiple applications, our field results specifically highlight the effectiveness of our planner for monitoring toxic microorganisms in a pre-alpine lake, and for identifying hot-spots within their distribution.

KEYWORDS

active sensing, environmental monitoring, informative path planning, marine robotics

1 | INTRODUCTION

To preserve environmental and human health, scientists need to characterize and monitor the physical, chemical, geological, atmospherical, and biological processes that occur in the Earth's ecosystems. These processes, whether natural or human-made, are complex because they evolve in both space and time, and they are all interrelated. Thus, their adequate observation and quantification require simultaneous measurement of diverse parameters over three dimensions and over adequate time spans in order to capture their spatial and temporal variability. Despite this complexity, scientific data-gathering campaigns still depend largely on arduous manual sampling procedures. For instance, in the particular case of monitoring of aquatic environments, even though oceanographers and limnologists* routinely use sensors to measure key parameters, such as pressure, temperature, dissolved oxygen, and chlorophyll fluorescence (a proxy for phytoplankton† abundance),[1] most sampling campaigns depend on manual water collection and deployment of sensors at specific depths from a boat.[2,3] As a result, datasets that provide large spatial and temporal coverage are still rare because such traditional data acquisition procedures are expensive and time consuming.

Autonomous mobile robots equipped with sensors emerge as a suitable solution, and are increasingly deployed to collect high-resolution data over large areas for applications such as gas detection,[4,5] geological surveys,[6,7] as well as for monitoring around[8–10] and within water bodies.[4,11–13] However, even though autonomous robots can turn tedious manual data collection into an automated, precise, and repeatable sampling strategy, datasets that provide a representative three-dimensional (3D) depiction of a given environment remain scarce because, even despite recent developments, it is still arduous

* Limnology is the study of inland water bodies.

† Nonmotile, microscopic algae living in fresh and marine waters.

to automatically plan efficient informative (that is, information-rich) paths in large 3D environments.

We present an IPP algorithm that works in continuous space, where it can interpret the measured data online and reason about relevant measurement locations. This enables the robot to adaptively replan the remainder of the sampling mission *in situ* to focus on features of interest. In this paper, the feature is a cyanobacteria, an aquatic microorganism that perform photosynthesis, meaning that it harvests energy from sunlight using pigments. Even though cyanobacteria are microscopic organisms, they can accumulate and form scums that can be seen by the naked eye. Such mass developments is named "bloom."

1.1 | Monitoring toxic cyanobacteria

Lakes are especially impacted by climate change, and its effects can be detected at the level of organisms, biological communities, as well as on the overall lake thermal structure.[14] An example of such climate-induced alteration is seen in Lake Zurich, Switzerland, where global warming has caused a weaker winter mixing of the water column, which in turn increased the abundance of the cyanobacteria *Planktothrix rubescens (P. rubescens)*.[15] *P. rubescens* is found in deep lakes of the Northern Hemisphere, and is a known producer of toxins called microcystins.[16] Figure 1-*Left* shows the scale of this microorganism. Microcystins are noxious to most organisms, including fish, cattle, and humans, and thereby constitute a major public health issue.[17] The presence and recent increase of *P. rubescens* in Lake Zurich is a serious concern, especially when considering that the lake is a popular recreational site and supplies drinking water for the 1.5 million people living in its vicinity.

P. rubescens is light sensitive and thus grows at depth in the metalimnion,* where it accumulates during late summer and early fall.[18] Using the ASV *Lizhbeth* shown in Figure 1-*Right*, which was designed specifically for lake sampling,[19] we documented the *P. rubescens* distribution in the metalimnion over a two-dimensional (2D) cross-section in Lake Zurich during five consecutive days.[3] Results showed that *P. rubescens* was found between 6 and 16 m depth, where it can be distributed quite uniformly along the transversal section on one day, and be found aggregated in patches on one shore the next day.[3] The analysis of the sampling we conducted demonstrated that spatial variations occur on both horizontal and vertical axes, and therefore 3D representations should provide a finer view of these changes. A detailed depiction is particularly relevant when studying the distribution of a microscopic organism (∼1 mm) in a large water body such as Lake Zurich (68 km^2).

1.2 | Informative path planning

Capturing variable environmental phenomena with an autonomous robot is a challenging task. The robot needs to travel along an efficient path in order to maximize the collected information, while respecting constraints such as energy, time or travel distance budgets. This is formally known as the IPP problem, which aims at maximizing the knowledge about a continuous scalar field $f(x)$ within a certain environment $\mathcal{E}$ from *in situ* measurements collected by a mobile robot equipped with an adequate sensor suite. This is achieved by planning an optimal path P^* for the robot:

$$P^* = \text{argmax}_{P \in \Psi_\mathcal{E}} \, u(\text{SAMPLE}(P)) \quad (1)$$

$$\text{s.t. } \text{COST}(P) \leq B.$$

The variable P defines a path from the space of all possible paths $\Psi_\mathcal{E}$ within the environment $\mathcal{E}$, the function $\text{SAMPLE}(\cdot)$ provides the finite set of measurement locations along this path, and $u(\cdot)$ is the utility (informativeness) of these measurements. The cost of the path is given by the $\text{COST}(\cdot)$ function and cannot exceed a predefined budget B. In contrast to metric path planning, IPP usually produces paths which are much longer than the minimal length path from the start to the goal, that is, paths which aim at exploring the entirety of a given environment. If the utility function is monotonically increasing (e.g., entropy), the budget constraint is fundamental in real applications as the optimal path would be infinitely long without it.†

In the basic setting of the problem, such sampling paths can be planned *a priori*, if the optimized utility u does not depend on the actual *value* of the measurements but only their *location*. This is the case, for instance, when a Gaussian process (GP) is used to model the quantity under investigation.‡ The objective function u in Equation 1 then only depends on the set of measurement locations x_i produced by $\text{SAMPLE}(\cdot)$, but not on the actual value of the underlying field at these locations $f(x_i)$. However, for many field applications it is desirable to define a specific interest in certain values (in our case, high abundances of cyanobacteria). This triggers a requirement for planning (and replanning) the path during the execution of the sampling mission, and thereby imposes constraints on the computation time. The applications of mobile sensing agents to collect data in all kinds of environments have rapidly increased during the last decades. Yet, most of the deployed systems have limited energy budgets to respect, making the planning of efficient measurement paths a central problem.

The goal of the field experiment was to identify areas where *P. rubescens* abundance exceeds a target value, hereafter called the threshold value, because *P. rubescens* does not occur uniformly in the lake. There are many large zones in a given sampling area where virtually no cyanobacteria can be observed. These empty areas are of lower interest to the limnology experts, and thus we prefer to spend most of the path budget on collecting measurements in areas with higher densities. Instead of covering the sampling area uniformly, we seek to (1) identify the areas of interest, which have concentration levels higher than the threshold, and to (2) take more detailed measurements within these areas. The adaptive planning scheme presented in this work aims at both by covering the sampling area sufficiently to identify the regions of interest and to focus the remaining traveling budget on these areas.

* Layer of a lake that displays the sharpest temperature gradient.

† Assuming non-zero sensor noise.

‡ This holds for any model which correlates measurements based on their position.

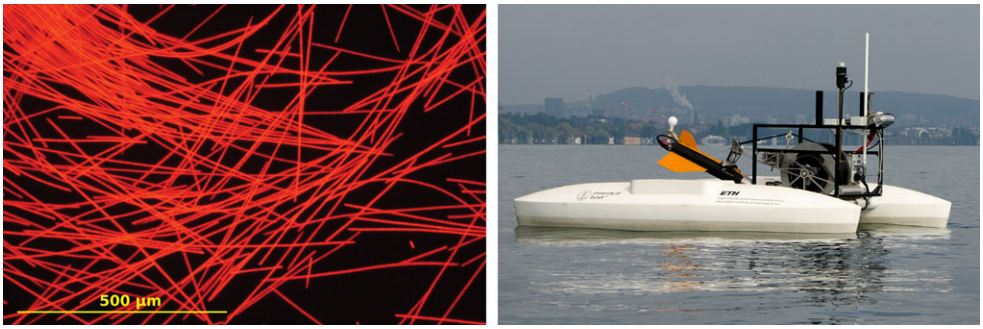

FIGURE 1 Subjects of interest for the environmental monitoring campaign. *Left*: Micrograph of autofluorescent filaments of *P. rubescens* under green light (image courtesy of Thomas Posch, University of Zurich, Switzerland). *Right*: The autonomous surface vessel (ASV), Lizhbeth, used for the monitoring on Lake Zurich, Switzerland

1.3 | Contributions

This paper presents a novel approach to the IPP problem which, instead of incrementally constructing a solution on a graph, uses a set of complete paths in continuous space together with an evolutionary strategy (ES) method to maximize information gain. Furthermore, a replanning scheme is presented to adapt the path according to the measured data, allowing to focus automatically on regions of interest, while ensuring that the robot sufficiently explores the area under study. Our method was extensively tested both in simulations and in field deployments using an ASV equipped with a suite of sensors to monitor *P. rubescens*.

The key contributions of this work are:

1. A new method for planning informative paths with the following properties:
   - operates in continuous space rather than on a graph,
   - obeys budget constraints,
   - uses constant memory and small computation time,
   - can provide any-time solutions, which are useful in real-world adaptive applications.
2. An adaptive replanning scheme to replan paths in order to identify regions of interest in the scalar field and explore them in more detail.
3. The implementation and evaluation of our IPP algorithm on an ASV for data acquisition in field deployments.

The data collected during the field deployments allowed to provide detailed depictions of the 3D distribution of the toxic cyanobacteria in the lake, which is a valuable asset to limnology experts.

The remainder of this paper is organized as follows. Section 2 presents prior studies relevant to our work. Preliminaries and theory of our proposed method are detailed in Section 3 and Section 4, respectively. We report on simulation results and parameter studies in Section 5, and present results from experimental validation in the field in Section 6. Finally, we discuss our findings in Section 7 and close the article with concluding remarks in Section 8.

2 | RELATED WORK

Path planning for mobile robots has been studied extensively in the last decades. However, there is a clear distinction between shortest path planning problems and IPP, which rests upon the discrepancy of the objective function. The problem of planning the shortest path from location A to B can be addressed by solving small path segments individually. The well-known *Dijkstra's* algorithm for planning the shortest paths is based on this concept and combined with a dynamic programming approach.[20] This approach is feasible due to the *modularity* of the objective function (usually Euclidean distance), meaning that the sum of the objective value of two segments is equal to the objective value of the concatenation of the two segments. However, the objective functions used in IPP are usually *submodular* and thus exhibit a diminishing return property. This property encodes the fact that measurements are less informative if other measurements were taken close-by.[21] Therefore, the objective value of a path segment is not constant but depends on all other path segments. In contrast to distance-based path planning, the length of the path is not the objective of the planning task, but is rather subject to a budget constraint, which limits the path from becoming too long. This constraint makes IPP challenging, since without it, we could plan an arbitrarily long path in order to collect a very large number of measurements.

The submodularity of the objective function arises from the spatial (and potentially temporal) correlation of the measurement locations. The sensor-network research community knows the problem well, in which a finite set of appropriate locations must be selected in order to install a static network of sensors that observe a given phenomenon.[22] Even though these fixed sensor sites do not need to be spatially arranged to create a feasible and efficient path for a mobile robot, this task has been shown to be non-deterministic polynomial-time (NP)-hard.[23] Nemhauser et al.[24] developed a heuristic approach that provides a provably near-optimal solution by iteratively choosing the best location.

By connecting the potential measurement sites to a graph, in such way that a robot can travel along the edges, the sensor site selection problem can be extended into the IPP problem. The greedy strategy was adapted to the IPP setting in the work by Chekuri and Pal.[25] Their *recursive greedy algorithm* recurses over possible middle points of a path and provides near-optimal solutions at the cost of *quasi-polynomial*

computation times. More recently, Binney et al.[26] have presented a generalized version of that algorithm, which allows for taking samples along the edges of the graph and for handling time-varying fields. Computing optimal solutions is impractical for any but the smallest problem instances. Binney and Sukhatme[27] have previously shown that branch and bound (BNB) methods can significantly decrease computation times, and thus push the feasibility limit to slightly larger graphs. In the context of this current paper, we use their approach to compare our method against a graph-based planner.

The principles of rapidly exploring random trees[28] have been adopted to the informative setting by Hollinger and Sukhatme.[29] While the submodularity of the objective function prevents the use of the rewiring step of the original algorithm, Hollinger and Sukhatme[29] demonstrate how the exponential growth of the tree can be limited with BNB techniques. Their approach has similarities to ours, as it does not require a predefined graph that spans the environment.

Alternatively, greedy approaches with limited look-ahead can be used.[11,30] These approaches have shorter computation times, enabling their application in adaptive settings, but provide no guarantees regarding optimality due to their limited planning horizon. Another simplification is to ignore the objective function's submodularity and assume that it is modular. Zhang and Sukhatme[31] showed that this yields good results if the planning graph is spaced looser than the length scale of the measured function. The same simplification enabled Yilmaz et al.[32] to formulate the planning task as a mixed integer linear programming (MILP) problem, for which they incorporated a large variety of constraints, such as path length budgets, communication range limits, and obstacle avoidance. A variation of the IPP problem is presented by Meliou et al.,[33] who aimed at minimizing the path length, while achieving a constant minimal informative goal. Even though this inversion is interesting from a modeling point of view, the fact that the resulting path length is unknown *a priori* is unpractical in a robotic application.

Graph-based methods generally suffer from the fact that a discretization of the continuous workspace is used. The choice of the density of graph has a large impact on the results. Our approach circumvents this by optimizing a parameterized path in continuous space. This mitigates the risk of using an inadequately sized graph.

Many applications aim not only at gaining information (i.e., exploring the environment), but also at exploiting the gained information in the form of application-specific interests. This requires to adapt the plan online because new measurements need to be taken into consideration. The works of Singh et al.[34] investigate methods to split the environment into sections, which are considered uncorrelated. By planning paths for each section and concatenating them, they were able to plan paths in large environments. In the aforementioned work of Zhang and Sukhatme,[31] the environment is implicitly separated by the loose spacing of the graph and thus ensures sufficient exploration, while the objective function is favoring exploitation. As we focus on highly correlated environments, such a separation is unpractical. An entirely different approach is to formulate reactive behaviors based on curiosity. Girdhar and Dudek[35] have implemented a curiosity-based visual exploration scheme on an autonomous underwater vehicle (AUV) to collect imagery in shallow waters. Most recently, Lim et al.[36] have presented a novel approach to adaptive IPP, which provides solutions in polynomial-time with optimality guarantees. In contrast to our work, the authors aimed at identifying a single true hypothesis without considering a limited travel budget for the robot.

Using IPP for the classical simultaneous localization and mapping (SLAM) scenario is similar to our setting, as we also consider the use of a mobile sensor to perceive information about the environment. In a SLAM context however, the operation space of the robot is typically more complex in its topology. This often leads to situations where only few places in the map are likely give more information, such as multiple unknown ends in a hallway. This has led to a considerable amount of work dedicated to deterministic and heuristic approaches, primarily geometric in nature and aiming at finding unknown frontiers in the map.[37,38] More rigorous, probabilistic, approaches have also been presented,[39–42] where the objective can be formulated based on information theoretic quantities and which allow to handle the exploitation-exploration trade-off more generically.

Path planning with continuous representations of paths in the form of splines has been discussed for various applications, such as optimizing AUV paths in the presence of currents,[43] respecting steering curvature constraints,[44] or smoothing results from graph-based planners.[45,46] Choi and How[47] studied the application of continuous representations for planning informative paths to make measurements for weather forecasts. In contrast to our work, they defined the splines in the robot's state space. Most related to our approach is the work presented by Marchant and Ramos,[48] which defines a similar information-theoretic objective and uses spline-based continuous paths. They use Bayesian optimization (BO) to optimize the parameters of the spline path but they only consider single spline segments, whereas our approach used multiple segments. This results in more complex paths, which allows for collecting more informative measurements. Furthermore, we also incorporate constraints on the robot travel budget so we can realistically plan during field experiments.

3 | PRELIMINARIES

Before describing our path planning method, we present two basic concepts on which our approach is based. First, we introduce Gaussian process regression[49] that we use to build non-parametric models of the scalar field of interest. Second, we briefly summarize the covariance matrix adaptation for the evolutionary strategy (CMA–ES) optimization method,[50] which we use in our planning routine.

3.1 | Modeling the underlying scalar field

Our goal is to investigate and model a scalar field within the environment, and therefore we need a suitable mathematical representation for it. We assume the field of interest to be a continuous function in d-dimensional space: $f : \mathcal{E} \mapsto \mathbb{R}$, where $\mathcal{E} \subset \mathbb{R}^d$ is the environment within which we want to make observations. However, we are only able to sample the function at a finite set of locations; thus a regression model is required to infer the function value at unvisited locations. GPs have been employed frequently for this purpose,[27,29,30] because they provide a probabilistic and non-parametric method to model the underlying field. Formally, a GP represents a distribution over

functions and is fully defined by a mean function $\mu(\mathrm{x})$ and a covariance function $k(\mathrm{x}, \mathrm{x}')$, the latter encoding the correlation of two measurements at locations x and x′:[49]

$$f(\mathrm{x}) \sim \mathcal{GP}(\mu(\mathrm{x}), k(\mathrm{x}, \mathrm{x}')).$$

Both the mean function and the covariance function usually depend on a set of static parameters, which are referred to as hyperparameters and denoted Θ. Given a set $\mathcal{A}$ of t noisy measurements $y = f(\mathrm{x}) + n$ with $n \sim \mathcal{N}(0, \sigma_n)$ and $\mathrm{x} \in \mathcal{A}$, we can evaluate the mean (Equation 2), covariance (Equation 3), and variance functions (Equation 4) of the posterior distribution as follows:[49]

$$\mu_A(\mathrm{x}) = \mu(\mathrm{x}_A) + \mathrm{k}_A(\mathrm{x})(K_A + \sigma_n I)^{-1}(y - \mu(\mathrm{x}_A)) \tag{2}$$

$$k_A(\mathrm{x}, \mathrm{x}') = k(\mathrm{x}, \mathrm{x}') - \mathrm{k}_A(\mathrm{x})^T(K_A + \sigma_n I)^{-1}\mathrm{k}_A(\mathrm{x}) \tag{3}$$

$$\sigma_A(\mathrm{x}) = k_A(\mathrm{x}, \mathrm{x}), \tag{4}$$

where $\mathrm{k}_A(\mathrm{x}) = [k(\mathrm{x}_1, \mathrm{x}), k(\mathrm{x}_2, \mathrm{x}), \dots, k(\mathrm{x}_t, \mathrm{x})]^T$ and $K_A = [k(\mathrm{x}, \mathrm{x}')]_{\mathrm{x},\mathrm{x}' \in A}$ is the covariance matrix of the measurement set $\mathcal{A}$. The choice of the mean function m and covariance function k and the corresponding set of hyperparameters Θ is essential for a good representation of f. For the applications presented in this work, a *squared exponential* covariance function with dimension-specific length scales was used. It is defined as

$$k(\mathrm{x}, \mathrm{x}') = \sigma_f \exp\left(-(\mathrm{x} - \mathrm{x}')\Sigma_\ell^{-1}(\mathrm{x} - \mathrm{x}')^T\right),$$

where Σ_ℓ defines a diagonal matrix of length scales $[\ell_1, \dots, \ell_d]$, and σ_f is a scaling factor. The resulting set of hyperparameters $\Theta = [\ell_1, \dots, \ell_d, \sigma_f, \sigma_n]$ can be optimized by training data to match the characteristics of f with various methods.[49]

As outlined in Section 1.2, the objective of IPP is to maximize the informativeness of the planned measurements along the path. In order to quantify this informativeness of a set of potential measurement sites $\mathcal{X}$ based on some previous measurements $\mathcal{A}$, we use mutual information,[51] which can be formulated analytically given the GP regression model:

$$u(\mathcal{X}, \mathcal{A}) = \mathrm{MI}(f; \mathcal{X} \mid \mathcal{A}) = \frac{1}{2}\log\left(\left|I + \sigma_n^{-2}K_A^X\right|\right),$$
$$\text{where } K_A^X = [k_A(\mathrm{x}, \mathrm{x}')]_{\mathrm{x},\mathrm{x}' \in X}. \tag{5}$$

3.2 | Optimization with the CMA–ES

CMA–ES is a generic optimization routine that is based on the concepts of evolutionary algorithms. It has been successfully applied to high-dimensional, gradient-free, and non-linear optimization problems.[50] As an evolutionary strategy, CMA–ES operates by iteratively evolving generations of solution candidates, called *offsprings*. CMA–ES is designed to solve *real-valued* problems and thus applies a normally distributed random variable to breed a new generation (as opposed to genetic algorithms, in which mutation and cross-over steps are used to modify binary or integer encoded offsprings). Algorithm 1 briefly outlines the method. The algorithm expects an objective function OBJECTIVE and an initial guess $\mathbf{m}_0$. Furthermore, it requires parameters specifying the population size λ_{CMA}, the initial step size σ_{CMA}, and a convergence criterion c. At each iteration, a new generation of candidate solutions $\mathrm{x}^{(i,\dots,\lambda_{\mathrm{CMA}})}$ is sampled from a multivariate Gaussian distribution according to a mean offspring $\mathbf{m}$ and covariance matrix C (Line 5). The newly sampled generation of offsprings is evaluated (Line 6) and used to update the mean and covariance matrix (Line 7). Furthermore, the step size parameter σ_{CMA}, which is used to scale the covariance matrix, is updated according to the *evolution path* of the solutions of the iterations.

The performance of the method depends on the UPDATE procedure, which defines the new mean, covariance matrix, and step size parameter. A detailed discussion of this step goes beyond the scope of this work and the interested reader is referred to the in-depth review by Hansen,[50] in which the necessary parameters for optimal updates are derived and a theoretical convergence analysis is presented. Here, the parameters suggested by Hansen[50] were used, and only the ones denoted as input to Algorithm 1 were adjusted to the optimization problem at hand. CMA–ES will be used as a global optimizer within the continuous-space informative path planner (CIPP) approach.

4 | THEORY

In this section, we propose a novel approach to plan informative paths by optimizing parameterized paths in continuous space. We call

ALGORITHM 1 CMA–ES

Input: Objective function OBJECTIVE, initial offspring $\mathbf{m}_0$, population size λ_{CMA}, initial scaling σ_{CMA}, convergence threshold c

Output: optimized solution $\mathbf{x}^*$

1: $C \leftarrow I$ // Initialize the covariance matrix to identity
2: $\mathbf{m} \leftarrow \mathbf{m}_0$ // Initialize the mean offspring
3: **while** $\mathrm{tr}(\sigma_{\mathrm{CMA}}^2 C) > c$ **do** // Iterate until convergence
4: **for** i in $1, \dots, \lambda_{\mathrm{CMA}}$ **do** // Sample a new generation
5: $\mathbf{x}^{(i)} \leftarrow \mathcal{N}(\mathbf{m}, \sigma_{\mathrm{CMA}}^2 C)$ // Sample a new offspring
6: $f^{(i)} \leftarrow \mathrm{OBJECTIVE}(\mathbf{x}^{(i)})$ // Evaluate the fitness of the offspring
7: $\mathbf{m}, C, \sigma_{\mathrm{CMA}} \leftarrow \mathrm{UPDATE}(\mathbf{x}^{(1,\dots,\lambda_{\mathrm{CMA}})}, f^{(1,\dots,\lambda_{\mathrm{CMA}})}, \mathbf{m}, C, \sigma_{\mathrm{CMA}})$ // Update the mean, covariance matrix and scaling factor
8: $\mathbf{x}^* \leftarrow \mathbf{m}$ // Return the best solution candidate

Note that the objective function can take further user-defined arguments.

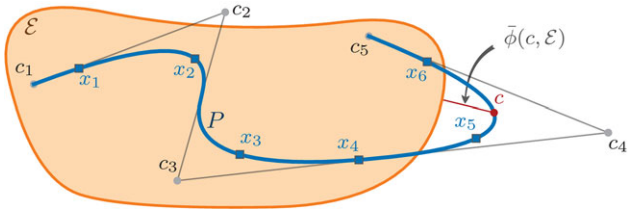

FIGURE 2 Schematic example of a path P (blue) defined by the control points $c_1, \dots, c_5$ in an environment $\mathcal{E}$ (orange). The control points are connected by gray lines to aid visual interpretation. The path exceeds the environment (indicated in red). This boundary excess is measured using the distance function $\bar{\phi}$ for a point c. Along the path six measurements $x_1, \dots, x_6$ (squares) are taken

this approach CIPP. The fundamental concept is to evolve a set of parameterized paths instead of incrementally building solutions on a discrete graph. Thus, we first apply the generic optimization method CMA–ES and provide a parameterization of a path, which can be optimized (see Section 4.1). Then, we formulate the objective function and consider budget and boundary constraints in Section 4.2. Furthermore, we propose an adaptive replanning scheme, which enables online replanning during a sampling mission in Section 4.3.

4.1 | Path parameterization

We parameterize a path P as a cardinal B-spline[52] of order k with n control points $[\mathbf{c}_1, \dots, \mathbf{c}_n]$. Furthermore, we define the B-spline to be *clamped*, meaning that we constrain the start and end points of the curve so they coincide with the first $\mathbf{c}_1$ and the last $\mathbf{c}_n$ control points. Note that this is a deliberate design choice because it simplifies the start- and end-point constraints during the optimization procedure. A schematic example of such a path is depicted in Figure 2. The path (shown in blue) is defined by a quadratic B-Spline with five control points (i.e., $\mathbf{c}_1$ to $\mathbf{c}_5$). We also define a cost function $\text{COST}(P)$ to compute the cost C_P of traveling along the path P. The choice of the cost function is platform dependent. While Euclidean distance is the simplest and most generic approach, more elaborate methods (e.g., computing energy requirements or travel time) might be required in real-world applications to get adequate results. For holonomic systems traveling at constant speed, these three metrics (i.e., distance, travel time, and energy consumption) are equivalent. In addition, we define a sampling function $\text{SAMPLE}(P)$, which provides a set $\mathcal{X}_P$ of discrete sampling locations along the path. Note that this function is also application dependent. Here, we consider a constant sampling frequency of the sensor and compute the spacing of the measurement sites with the traveling speed of the robot. In the toy example shown in Figure 2, a constant traveling speed was assumed, resulting in equally spaced measurement sites $x_1, \dots, x_6$. Note that the function $\bar{\phi}(c, \mathcal{E})$, which is highlighted in Figure 2, will be explained in the following section (Section 4.2). In this paper, we will denote the length of a path as $|P|$, and $P(\mathbf{p})$ will refer to a path built up by a list of control points $\mathbf{p}$. We will use the notation $P|_d$ to refer to a point at a distance d along the path P. According to this, we denote a subsection of path P from points $P|_{d_1}$ to $P|_{d_2}$ by $P|_{d_1}^{d_2}$. Furthermore, we assume that these functions are limited to the length of the path (i.e., $P|_d := P|_{\min(\max(d,0),|P|)}$).

4.2 | Continuous-space informative path planning

In order to make use of the path parameterization introduced above, the IPP problem as presented in Equation 1 needs to be reformulated. The optimization is now defined over the path parameters and further constraints are required:

$$\mathbf{p}^* = \text{argmax}_{\mathbf{p} \in \mathbb{R}^{nd}} \, u(\text{SAMPLE}(P(\mathbf{p}))) \tag{6}$$

$$\text{s.t.} \quad \text{COST}(P(\mathbf{p})) \leq B \tag{7}$$

$$P(\mathbf{p}) \in \Psi_{\mathcal{E}} \tag{8}$$

$$\mathbf{c}_1 = \text{constant} \tag{9}$$

$$\mathbf{c}_n = \text{constant}, \tag{10}$$

where $\mathbf{p}$ is a vectorized version of the list of control points of the path:

$$\mathbf{p} = \text{vec}([\mathbf{c}_1, \dots, \mathbf{c}_n]). \tag{11}$$

Along with the previously defined budget constraint (Equation 7), we need to impose a boundary constraint to ensure that the complete path remains within the environment $\mathcal{E}$ (Equation 8). Additionally, we might want to impose constraints regarding the start and end points of the path (Equations 9 and 10). This is especially useful in real-world applications, where the deployment and recovery of the robot is restricted to predefined locations. The start/end point constraints can be easily implemented by omitting these points in the vectorization step:

$$\mathbf{p} = \text{vec}([\mathbf{c}_2, \dots, \mathbf{c}_{n-1}]).$$

To incorporate the remaining constraints into the objective function, we use multiplicative penalty factors g_{budget} and g_{boundary}:

$$\bar{u}(\mathbf{p}, \mathcal{A}, B, \mathcal{E}) = u(\mathcal{X}_P, \mathcal{A}) \, g_{\text{budget}}(\mathbf{p}, B) \, g_{\text{boundary}}(\mathbf{p}, \mathcal{E}). \tag{12}$$

The penalty factors are defined as follows:

$$g_{\text{budget}}(\mathbf{p}, B) = \epsilon(B - C_P),$$

$$g_{\text{boundary}}(\mathbf{p}, \mathcal{E}) = \epsilon\left(\int_0^{|P|} \bar{\phi}(P|_s, \mathcal{E}) ds\right), \tag{13}$$

where $\epsilon : \mathbb{R} \to (0, 1]$ defines a smoothed, one-sided step function

$$\epsilon(x) = \begin{cases} e^{(-x^2/(2c_h^2))} & \text{if } x < 0 \\ 1 & \text{otherwise} \end{cases},$$

and $\bar{\phi}$ defines the negative part of the signed distance function

$$\bar{\phi}(\mathbf{c}, \mathcal{E}) = \begin{cases} -\inf_{\mathbf{c}' \in \mathcal{E}} \|\mathbf{c} - \mathbf{c}'\|_2 & \text{if } \mathbf{c} \notin \mathcal{E} \\ 0 & \text{otherwise} \end{cases},$$

which returns the smallest distance of a point $\mathbf{c}$ to the environment $\mathcal{E}$. Figure 2 shows a path exceeding the environment $\mathcal{E}$ partially and indicates the distance function $\bar{\phi}$ for a point $\mathbf{c}$. The boundary constraint needs to be converted into a penalty term for two reasons. First, simply restraining the control point of the path from exceeding the environment is too conservative, because it restricts the B-spline path from accessing parts of the environment. Figure 2 shows such a case, where the control point $\mathbf{c}_2$ lies outside of the boundaries, yet the corresponding section of the path does not violate the constraint. Second, the translation of the constraint to a penalty term leads to an unconstrained optimization problem (Equation 12), which can be maximized with generic global maximization heuristics. For this purpose, we use the CMA–ES routine, as introduced above (see Section 3.2).

4.3 | Adaptive replanning

Adaptive replanning is necessary when the objective depends on the actual measurements. The objective described in Equation 12 only depends on the location of measurements and not on their value. However, in real-world applications, it can be very valuable to define value-dependent objectives, such as finding the maximum of f,[48] classifying level sets,[11] or focusing on specific value ranges. More intuitively formulated, adaptive replanning is necessary when we are concerned not just with *where* we measure, but also *what* we measure. In our work, we focus on high-valued regions of interest as we are particularly interested in areas with high density of cyanobacteria. We formalize this interest by specifying a threshold f_{th}, which separates the interesting value range (above) and the uninteresting value range (below). Furthermore, we modify the objective function (Equation 12) in such a way that locations yielding measurements lower than the threshold f_{th} are ignored:

$$u_{\text{adaptive}} = \text{MI}(\mathcal{X}_I \mid \mathcal{A}), \tag{14}$$

where $\mathcal{X}_I$ denotes the set of *interesting* sampling locations:

$$\mathcal{X}_I = \{\mathrm{x} \mid \mathrm{x} \in \mathcal{X} \wedge f(\mathrm{x}) \geq f_{\text{th}}\}.$$

ALGORITHM 2 CIPP

Input: start position $\mathbf{c}_1$, end position $\mathbf{c}_n$, number of control points n, budget B, environment $\mathcal{E}$, prior measurements $\mathcal{A}$

Output: optimized control points $\mathbf{c}^*_{1,\dots,n}$

1: $\mathbf{c}_{1,\dots,n} \leftarrow$ INITALIZE_PATH$(\mathbf{c}_1, \mathbf{c}_n, n, B, \mathcal{E})$ // Create an initial solution
2: $\mathbf{m}_0 \leftarrow \text{vec}(\mathbf{c}_{1,\dots,n})$ // Vectorize the list of points
3: $\mathbf{p}^* \leftarrow$ CMA-ES$(\bar{u}, \mathbf{m}_0, \lambda_{\text{CMA}}, \sigma_{\text{CMA}}, c, \mathcal{A}, B, \mathcal{E})$ // Call CMA–ES
4: $\mathbf{c}^*_{1,\dots,n} \leftarrow \text{vec}^{-1}(\mathbf{p}^*)$ // Convert the optimized solution to a list of points

Finally, we formally define the CIPP procedure in Algorithm 2. As it makes use of the CMA–ES optimization routine, it comes down to two major steps. First, an initial solution must be generated (Line 1). Second, the optimizer is called (Line 3) to produce the final solution (Line 4). The initial solution is provided to the CMA–ES optimizer as the initial mean offspring $\mathbf{m}_0$. The initialization (i.e., the INITIALIZE_PATH function) can be implemented in various ways. However, it is important that it provides a feasible solution that does not conflict with the budget constraint nor the boundary constraint. For our work, we have defined it by a simple heuristic, which initializes the control points on a straight line from the start to the end point. If the start and end coincide, the path is initialized with two straight lines, going from the start to the farthest point in the environment and back to the end point. If this results in a path which is longer than the budget B, it is scaled such that it has a length of B. This approach is only guaranteed to provide valid paths for convex and obstacle-free environments, which are assumptions on the environment that are both met by the field applications considered in this work. For more complex, non-convex environments, one could replace the straight lines with shortest path solutions produced by a metric path planning algorithm.

Since $f(\mathrm{x})$ is unknown, we have to use the GP model to make an estimate of $f(\mathrm{x})$. For this, we incorporate the model uncertainty $\sigma^2_{\text{A}}(\mathrm{x})$ and scale it by a design parameter β:

$$\mathcal{X}_I = \{\mathrm{x} \mid \mathrm{x} \in \mathcal{X} \wedge \mu_A(\mathrm{x}) + \beta\sigma_A(\mathrm{x}) \geq f_{\text{th}}\}. \tag{15}$$

The design parameter β can be used to specify with what certainty a point has to lie below the threshold before it is classified. We have adopted this uncertainty-aware classification principle from Ref. 53, who used similar sets to maximize GPs, as well as from works by Gotovos et al.[54] that focused on level set estimation. In these studies, the full set is classified into two subsets: one is lower than the threshold $\mathcal{X}_-$ and one is higher $\mathcal{X}_+$. In our work, we make use of the lower set $\mathcal{X}_- = \{\mathrm{x} \mid \mathrm{x} \in \mathcal{X} \wedge \mu_A(\mathrm{x}) + \beta\sigma_A(\mathrm{x}) \leq f_{\text{th}}\}$, as the set of interesting locations can be seen as its complement: $\mathcal{X}_I = \mathcal{X} \backslash X_-$.

Replanning the path can only be done once new measurements have been taken *in situ*, and consequently it is usually subject to a real-time constraint. To accommodate the limited planning time, we propose a replanning scheme in a receding-horizon fashion, which makes local modifications to a prior global path P_g. Given that the global path is

planned prior to the deployment of the robot (i.e., independent of any values measured *in situ*), it aims purely at exploring the environment uniformly. The local path P_ℓ has additional budget available at each iteration i to exploit the collected information about f.

impose an additional constraint, which we formulate as an additional penalty factor in the objective function:

$$g_{\text{progress}} = \epsilon \left(h - \left\| P|_{d_{\text{tr}}} - \mathbf{c}_n^{i+1} \right\|_2 \right).$$

ALGORITHM 3 The adaptive replanning scheme

Input: Global path P_g, planning horizon h, travel distance d_{tr}, budget factor λ, environment $\mathcal{E}$
Output: Set of measurements $\mathcal{X}$

1: $B_\ell \leftarrow h(1+\lambda)$ // Budget for the local planner
2: $d_{\text{tr},g} \leftarrow d_{\text{tr}}/(1+\lambda)$ // The global equivalent of the traveling distance
3: $\mathbf{c}_1^0 \leftarrow P_g(0)$ // Variable initializations
4: $i \leftarrow 0$
5: $B_{\text{spent}} \leftarrow 0$
6: $\mathcal{X} \leftarrow \emptyset$
7: **while** True **do**
8: $B_\ell^i \leftarrow \min(B_\ell, B - B_{\text{spent}})$ // The budget for the local planner, limited to the overall budget.
9: $\mathbf{c}_n^i \leftarrow P_g|_{h+i\,d_{\text{tr},g}}$ // The goal point for this iteration.
10: $\mathbf{c}_n^{i+1} \leftarrow P_g|_{h+(i+1)d_{\text{tr},g}}$ // The goal point of the next iteration for the progress constraint.
11: $d_{\text{pr}}^i \leftarrow \min(h, B_g - i d_{\text{tr},g})$ // The distance for the progress constraint.
12: $P_\ell^i \leftarrow \text{CIPP}(\mathbf{c}_1^i, \mathbf{c}_n^i, B_\ell^i, \mathcal{E}, \mathcal{X}, \mathbf{c}_n^{i+1}, d_{\text{tr}}, d_{\text{pr}})$ // Execute the CIPP planner to compute a local path.
13: $\mathcal{X} \leftarrow \mathcal{X} \cup \text{SAMPLE}(P_\ell^i|_0^{d_{\text{tr}}})$ // Sample along the first part of the local path.
14: $\mathbf{c}_1^i \leftarrow P_\ell^i|_{d_{\text{tr}}}$ // Update variables.
15: $B_{\text{spent}} \leftarrow B_{\text{spent}} + d_{\text{tr}}$
16: $i \leftarrow i + 1$
17: **if** $B_\ell^i \leq d_{\text{tr},g}$ **then** // Break once all budget is spent.
18: **break**

Algorithm 3 outlines the iterative procedure and Figure 3 provides a schematic description of the first two iterations on a toy example. The fundamental concept of the replanning scheme is to distribute some extra budget fairly along a global path. For this, we split the total budget B into a global part $B_g = \lambda B$ and an adaptive part $B_a = (1-\lambda)B$, where $\lambda \in (0, 1)$ is a design parameter. While the global budget is used to plan the global path P_g, the adaptive budget is to be spent incrementally. The local path P_ℓ is planned over a horizon h on the global path P_g from the current position $\mathbf{c}_1^i$ to the target point $\mathbf{c}_n^i$ (Line 12). For this path, a budget of $B_\ell = h(1+\lambda)$ is available. Both the horizon h on the global path P_g and the local path P_ℓ are highlighted in Figure 3-*Left*. Only a subpart of length d_{tr} of the local path is executed (Lines 13 and 14) before a new local path is planned in order to allow for timely adaptations to the measured values (see Fig. 3-*Right*). As a consequence, the starting point $\mathbf{c}_1^i$ lies on the global path only in the first iteration. Therefore, in subsequent iterations, the target point $\mathbf{c}_n^i$ is defined on the global path at the horizon distance from a virtual reference point $\hat{\mathbf{c}}_1^i$ on the global path (Line 9). This reference point is incremented by a scaled version of the traveling distance: $d_{\text{tr},g} = d_{\text{tr}}/(1+\lambda)$ (indicated in Fig. 3-*Right*). While the budget for the local path B_ℓ is constant, the shortest distance from the start point $\mathbf{c}_1^i$ to the end point $\mathbf{c}_n^i$ depends on the path of the last iteration $P_\ell{}^{i-1}$. To prevent this distance from getting too large, we

This factor penalizes the objective value if the distance from the (variable) starting point of the next iteration $\mathbf{c}_1^{i+1} = P|_{d_{\text{tr}}}$ and the end point of the next iteration $\mathbf{c}_n^{i+1}$ exceeds the horizon h, and enforces the minimal required progression along the global path. The progress constraint is highlighted in Figure 3 by the red lines. Formally, it requires that the necessary parameters (the reference point $\mathbf{c}_n^{i+1}$, the travel distance d_{tr}, and the progress constraint distance d_{pr}) are passed to the objective function of CIPP (Line 12). In Algorithm 3, these parameters are updated iteratively on Lines 10 and 11.

5 | EXPERIMENTAL VALIDATION IN SIMULATION

In this section, we present a performance evaluation of the proposed planning algorithm, as well as the replanning scheme, where we analyze the influence of the key parameters. We show a qualitative example of CIPP, present results of its comparison against two competing methods in Sections 5.1 and 5.2, and study its key parameters in Section 5.3. The adaptive replanning scheme is evaluated in Section 5.4.

Before discussing the evaluation and its results, a brief, qualitative, example of the execution of the CIPP algorithm is presented in

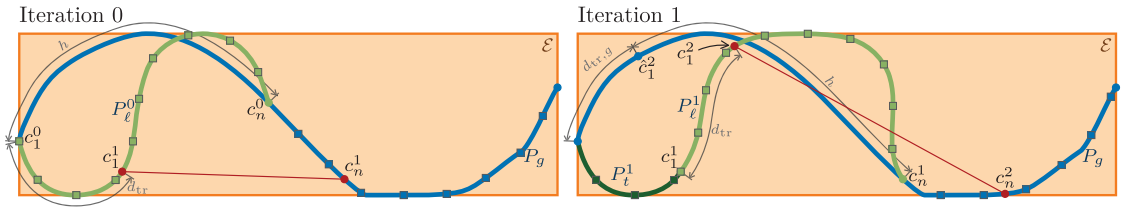

FIGURE 3 Schematic explanation of the adaptive replanning scheme. The first two iterations are shown. Given a global path P_g (blue) within the environment $\mathcal{E}$ (orange), a local path P_ℓ^i (light green) is planned at iteration i over a horizon of h from the starting point c_0^1 toward the target point c_n^i. After traveling a distance of d_{tr} and reaching c_1^1, a new iteration $i+1$ is started. The target point c_n^{i+1} is advanced to the virtual reference point $\hat{c}_1^2$ by $d_{\mathrm{tr},g}$ along the global path. The squares indicate sites where measurements were taken (dark green) and sites where they will be taken, i.e., the locally (light green) and globally planned sites (blue). In both iterations, the progression along the global path is constrained (red line)

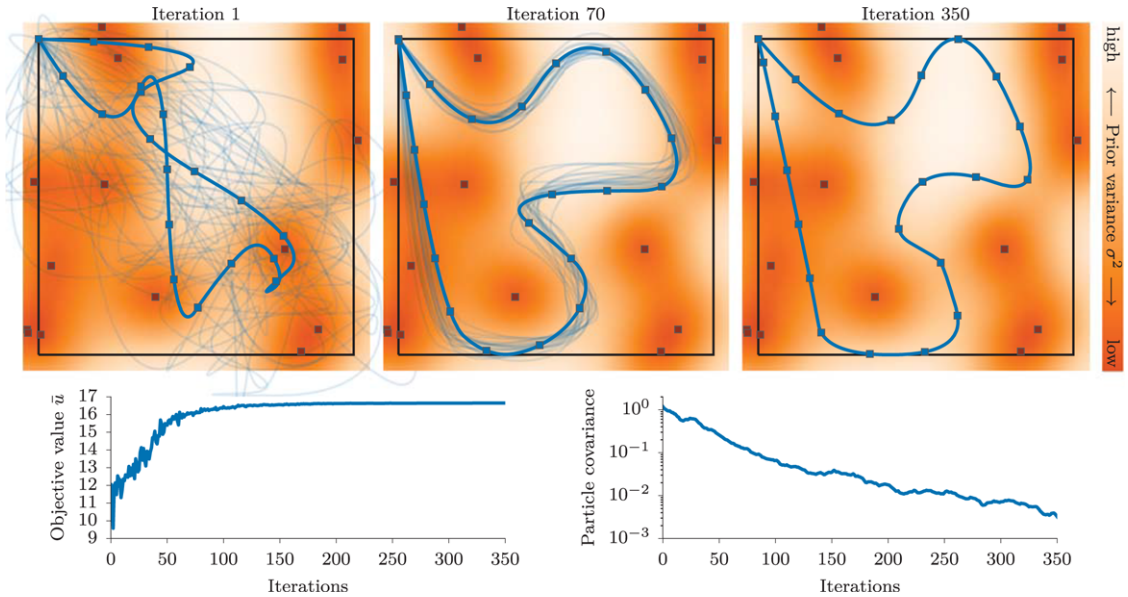

FIGURE 4 Example run of CIPP on a square environment (black boundaries) with set of prior measurements (orange squares). Dark orange corresponds to high prior model certainty or, in other words, low prior variance. *Top*: First, 70th, and last iterations. All offsprings of the current generation are shown as transparent blue paths. The best path of the current iteration is shown in opaque blue, and measurements taken along it are indicated by blue squares. The objective value $\bar{u}$ and the trace of the covariance matrix of the offsprings are shown in the *Bottom-Left* and *Bottom-Right*, respectively

Figure 4. Essentially, it is a basic scenario of a square environment that displays a set of randomly-placed prior measurements. The path representations of the offsprings (i.e., candidate paths) of the current generation are shown by the transparent blue lines for the first, the 70th and the last iteration. At each iteration, a fixed number of offsprings is evaluated using the objective function, which allocates an objective value to each single offspring. The evaluated offsprings are then used to update the internal covariance matrix of the CMA–ES optimizer. Each iteration brings a new generation of offsprings, which has its own mean offspring. The mean offspring **m** is shown in Figure 4 as a solid blue line for each iteration. Initially, the optimizer spreads offsprings randomly throughout the planning space (see Fig. 4-*Left*), which often leads to paths with very poor objective value. However after further iterations, the candidate paths converge to a (locally) optimal solution. Qualitatively, the solution shown in Figure 4 performs well, as the new measurements (blue squares) tend to cover the areas where prior variance was high or, conversely, prior certainty was low (white areas). The graph on the bottom left of Figure 4 shows the evolution of the objective value of the mean offspring. It exhibits a quick plateauing, which corresponds to large initial improvements and small refinements at later iterations. The graph on the right-hand side shows the decrease of the trace of the offsprings' covariance, which indicates that the algorithm converges.

5.1 | Comparison against graph solutions

Because the CMA–ES method is not guaranteed to converge to the global maximum, and given that we cannot formulate lower bounds on the achieved objective value, we evaluate the method empirically in simulations. No lower bound on performance means that our method could perform arbitrarily bad. To assess the performance globally, we would ideally compare against optimal solutions. To the best of our knowledge however, there is no IPP method that produces optimal results when operating in continuous space. Thus, we have to resort to a graph-based approach to compare against. We compare the CIPP against the branch and bound (BNB) method proposed by Binney and

Sukhatme[27] because the latter provides globally optimal solutions (up to discretization). The performance of the BNB approach is optimal only on the graph, whereas our method operates in continuous space. Thus, the primary goal of the comparison is to show that CIPP does not perform significantly worse.

The BNB method operates on a graph $\mathcal{G} = (V, E)$ defined by a set of vertices V and a set of edges E, which connects the vertices. The key part of the solution given by Binney and Sukhatme[27] is the formulation of an upper bound for a partial solution, which limits the potential maximal objective value that could be achieved from a partial path. The fact that the objective function is assumed to be submodular requires the upper bound to include the measurement sites on all edges, which could potentially be reached given the remaining budget. Even though this upper bound is relatively loose, their method clearly performs faster than pure brute force approaches. Unfortunately, their approach requires the choice of a graph design, which has a significant influence on the performance of the method. Although only few authors discuss the reasons why they have selected a certain graph design, in real-world applications this choice is critical: graphs with dense vertex distributions largely increase the complexity of the problem, whereas too sparse graphs might limit the solution space too drastically.

We chose the densest graph that still allowed for reasonable running times in order to make the comparison in our evaluation as fair as possible (Fig. 5). We used a square grid graph of 6 by 6 vertices separated by an edge length d_e, which is similar to the ones used for the evaluation in the original work of Binney and Sukhatme.[27] While the budget B_{BNB} for the maximal path length is defined as a number of edges for BNB, we formulate the budget for CIPP as the corresponding multiple of the edge length: $B_{\text{CIPP}} = B_{\text{BNB}} d_e$. This ensures that both methods produce paths of equal length and allows for a fair comparison of the information gathered. Along the paths, we take equidistant samples at a distance of $d_e/2$ from each other, which is equivalent to 2 samples per edge on the graph. For the evaluation, we created 10 scenarios with a set of 10 prior measurements sampled from a uniform distribution. In all scenarios, the start point and end point of the path have been fixed to the bottom left and top right corners, respectively. The restriction to even path length budgets is imposed by the structure of the graph and by the choice of the start and end points (i.e., there are no feasible paths from start to end in the given graph with odd length). Because the CMA–ES method is not deterministic (due to the random sampling of the offsprings), we ran the CIPP 50 times for each scenario and path length budget. This resulted in a total of 50 runs for the BNB solver and 2, 500 runs for the CIPP algorithm.

As an example of those evaluation runs, Figure 5-*Left* shows one of the scenarios used for the evaluation with 10 randomly placed prior measurements (orange squares). The figure also shows the graph for the BNB solver (gray), the BNB solution (purple), and two of the 50 CIPP solutions corresponding to the 10^{th} and 90^{th} percentile in terms of objective value (light and dark green respectively). The dark green CIPP solution roughly follows the BNB solution, while the light green one first explores the center region of the environment, and then goes to the upper left corner. In terms of objective values the darker CIPP path outperforms the purple BNB path. This is possible because BNB only provides an optimal solution on the given graph whereas CIPP is not restricted to that graph.

The graphs on the right of Figure 5 show the quantitative comparison of the achieved objective values and the required computation times. To compare the objective values of runs with different path budgets, we normalized the 50 CIPP results with the corresponding BNB value. Thus, values above 100% indicate a better performance of the continuous-space solution. The CMA–ES optimizer requires a convergence threshold c, which trades off the achieved objective value and the required computation time. We have used two different threshold values, one for early solutions and the other for more refined results, respectively, denoted *coarse* and *fine* in Figure 5. The results shown in Figure 5 lead to two conclusions. First, CIPP provides similar results as the graph-based approach on these evaluation scenarios. Albeit CIPP might theoretically perform arbitrarily bad due to the lack of lower bounds on the performance, this comparison empirically shows that the method provides competitive results. Due to its continuous operation space, CIPP sometimes provides solutions that are better than the optimum on the graph. This, however, is primarily due to the choice of the resolution of the graph. Second, the comparison of the computation times shows that the CIPP approach scales almost linearly, whereas with the BNB technique the computation times increases almost exponentially. For very small budgets (i.e., for a path budget of 14 edges), the BNB solution has only very few possible solutions (note that a minimal path length of 10 edges is needed to connect the start and end points). But for larger budgets, the number of possible paths increases exponentially and thus the required computation time also increases quickly. However, this comparison of computation times has to be considered carefully because the objectives of the two methods are different: the BNB aims at providing guaranteed optimal solutions on a discrete graph, while the CIPP provides locally optimal solutions in continuous space.

In terms of complexity, the two algorithms scale differently. While the BNB approach, or any other graph-based solution, scales in terms of the graph size (i.e., the number of vertices and edges), the CIPP primarily scales by the number of control points that are used to parameterize the path. To obtain finer grained results, the complexity will inevitably increase because the BNB requires a denser graph, and the CIPP needs more control points. Graph-based solutions also scale by the total path budget, and since longer paths contain more edges, they are more complex to optimize. This is not the case for CIPP, as the length of the B-spline path is independent of the number of control points. However, for topologically more complex spaces or if the kinodynamic constraints on the vehicles are more significant, one might require substantially more control points to sufficiently parameterize the path.

5.2 | Comparison against BO

The performance of the optimization routine CMA–ES was compared to the one of BO. The use of BO for optimizing the parameters of a spline-based path representation has been suggested by Marchant and Ramos.[48] Their work is in line with ours because they applied a gradient-free global optimization routine for optimizing the

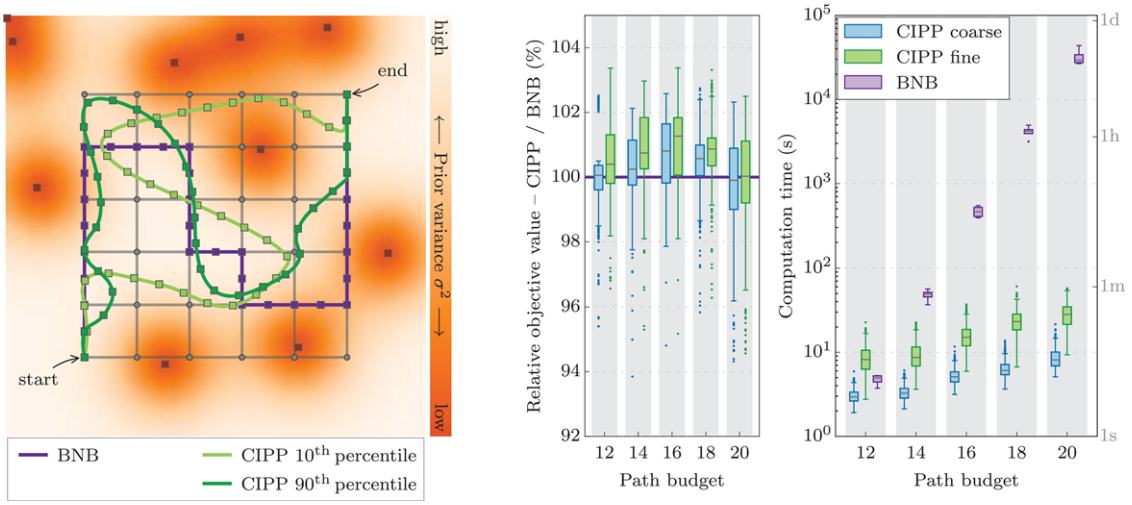

FIGURE 5 *Left*: Example of an evaluation scenario, where the following elements are shown: 10 randomly sampled prior measurements (orange squares) that define the prior certainty (the darker the orange, the lower the prior variance), a 6 × 6 graph (gray) with edge length d_e on which the BNB solution was computed (purple), and two of the 50 CIPP solutions (bright and dark green), which correspond to the 10th and 90th percentile in terms of objective value. *Middle*: Comparison of the achieved objective value. For each path budget, the CIPP solutions are shown relative to the BNB solution. *Right*: Comparison of the computation times. The CIPP was run with two different convergence thresholds yielding coarse (light blue) and finer (light green) results. Note the logarithmic scale of the *Computation time* axis.

parameters of a continuous path in an information collection scenario. Nonetheless, their work differs from our approach in three aspects. First, they integrate the utility function over the path, whereas we evaluate the utility function for a discrete set of measurement points. Our choice is primarily motivated by the usage of a sensor with limited recording rate, which only provides point measurements along the path. Second Marchant and Ramos,[48] do not limit the length of the planned path. In our approach however, a limited path budget is fundamental for meaningful field applications. Finally, in their work they limit the number of optimization variables to five by planning for a single element of a cubic spline curve. Our method aims at planning paths containing multiple control points, which results in a larger number of optimization variables and gives more fine-grained control of the planned path.

Despite these differences, BO can be used to maximize our objective function (Equation 12) because this generic optimization framework can handle highly non-convex objective functions.[55] We have compared the two optimization routines on simulated scenarios, which were similar to the ones employed to compared our approach against the graph-based one, as described above. We used 2D square environments (10 m by 10 m in size) and placed 15 prior measurements at random locations within these bounds. We have set the starting point for the path to the bottom left corner and used a budget of 140 m. The SAMPLE function was chosen to output equidistant measurements at 2 m intervals along the paths.

The results are shown in Figure 6 that depicts the optimized objective value and the required computation time. The comparison of the two methods leads to two main insights. First, CMA–ES consistently outperformed BO and more distinctively so at higher numbers of control parameters (i.e., higher dimensional search space). Second, the required computation time was significantly lower for CMA–ES. Note that all experiments were run until convergence. This difference in performance is most likely caused by either over or under fitting of the GP model, which BO uses to approximate the objective functions. The penalty terms for budget and boundary excess introduce steep edges, which might be difficult to model appropriately with stationary covariance functions. In the work of Marchant and Ramos,[48] no such penalty terms were used and thus their effects on performance were not studied. Furthermore, for higher dimensions of the search space in the optimization problem (in our case, the number of control points in the path), an increasing number of samples is required to build a meaning full Bayesian model of the objective function and accurately train the hyperparameters.

5.3 | Parameter evaluation

There are two main parameters that significantly influence the solutions produced by CIPP: (1) the number of control points of the path and (2) the degree of the B-spline used for the path representation. To evaluate the dependency of our method on these parameters, an empirical study was conducted. We used similar scenarios to the ones described above, i.e., square environments of 20 m by 20 m with 15 randomly placed prior measurements. The CIPP algorithm was applied with linear, quadratic, and cubic B-splines for nine path budgets (40, 50, … , 120) and nine sets of control points (3, 5, … , 19). For each setting, 50 repetitions were made, which resulted in a total of 12, 150 (50 × 9 × 9 × 3) runs. The SAMPLE function was chosen to output equidistant samples at distances of 2 m, meaning that paths of different length generate different numbers of samples. To make the results comparable over various path lengths, the final objective values of all runs with the same path length were normalized as per the maximum of their objective values.

The results of this experiment are summarized in Figure 7, where the normalized medians of the achieved objective values along with the median of the required calculation times are presented. The graphs demonstrate that more control points generally result in better

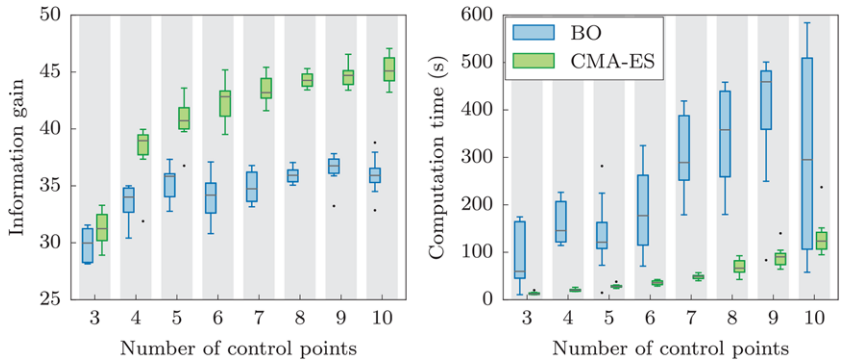

FIGURE 6 Comparison of CMA–ES and BO for planning informative paths with a path budget of 140 m for a varying number of control points. All experiments were evaluated on 10 randomly created square environments of 10 m edge length. *Left*: The achieved final objective values for both methods. *Right*: The required computation time to reach convergence

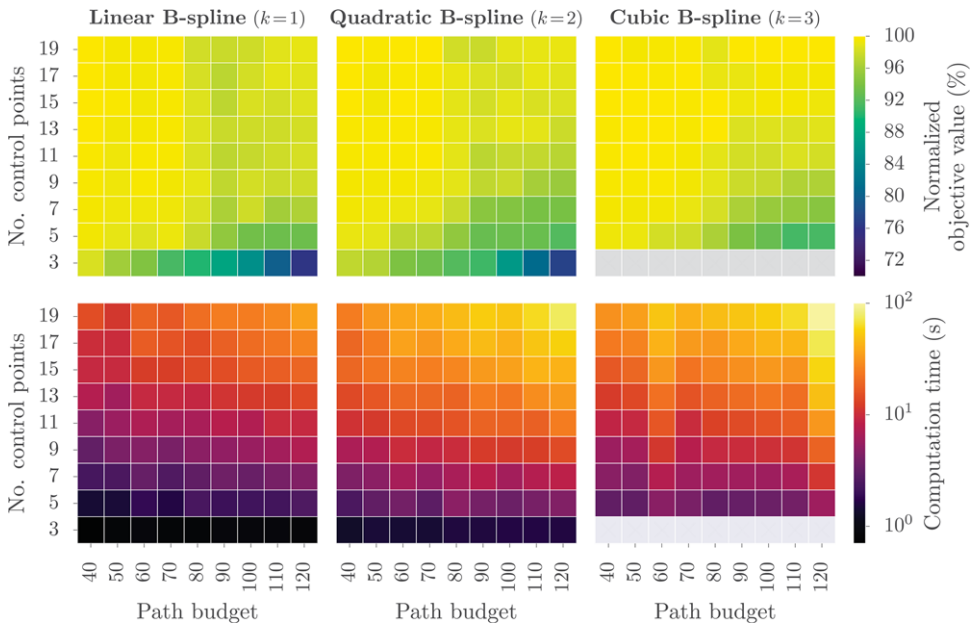

FIGURE 7 Performance evaluation results of the CIPP algorithm for varying path budgets and number of control points. For cubic B-splines, at least 4 control points are required, hence the lowest row is invalid on the corresponding graphs. The results are averaged values over 10 runs. *Upper row*: The achieved final objective value. To compare paths of different lengths, each column has been normalized by the maximal value separately. Large values (yellow) correspond to good results, smaller values to worse performance (dark blue). *Lower row*: The required computation times. Longer computation times (bright orange) are needed to optimize larger numbers of control points, whereas low numbers can be optimized very quickly (black). Note the logarithmic scale on the *Computation time* colorbar

solutions (Fig. 7-*Upper row*). However, above a certain number, additional control points do not bring further improvements. The comparison of the degree of the B-spline shows that there are surprisingly few differences with regard to this parameter. Overall, higher degrees seem to produce slightly better results, particularly for larger path budgets. On the other hand, the linear B-splines seem to require a lower minimal number of control points to achieve good results. Regarding computation times, there are two obvious trends, which are independent of the degree of the path representation (Fig. 7-*Lower row*). The computation time increases for both larger path budgets and larger numbers of control points. When considering the spline degree, a slight trend indicates that lower degrees require lower computation times.

Two main conclusions can be drawn from this evaluation. First, the number of control points must not lie below a certain minimal value. This lower bound is, however, dependent on the path budget and presumably also on the size of the environment. Increasing the number of control points significantly above this threshold value only provides marginal improvements at the cost of increasing complexity and computation times. Second, the comparison of the order of the path representation reveals that they differ in the minimal number of control points required, but are able to provide similar results. This suggests that the choice of a path representation can be made based on application- or platform-specific needs.

5.4 | Adaptive replanning evaluation

Our evaluation of the adaptive replanning scheme is twofold. On the one hand, we examine its ability to focus on user-specified, value-dependent, regions of interest in various settings. On the other hand,

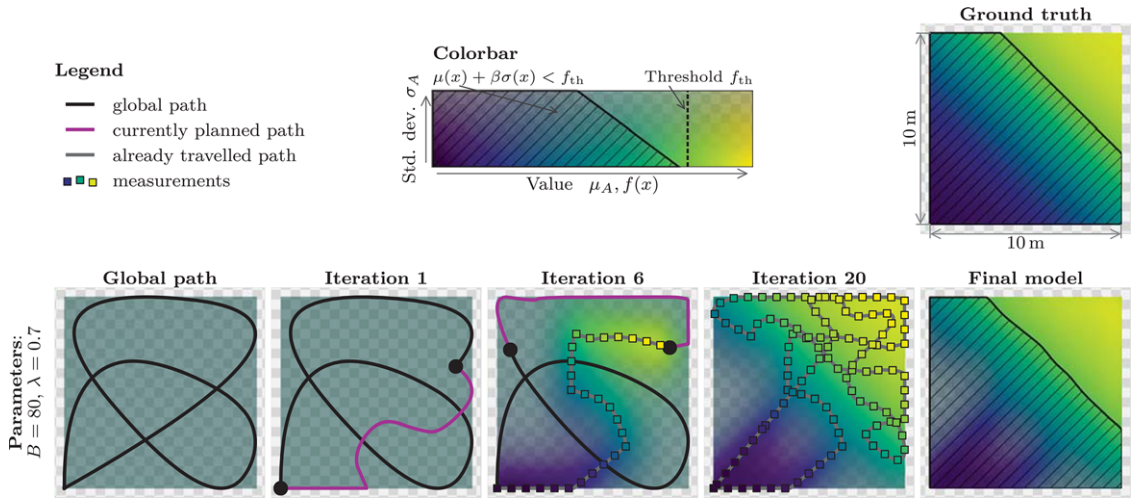

FIGURE 8 Simulation results of the adaptive replanning scheme on a simple toy example. *Top row*, *left*: The legend for the types of paths and measurement symbols. *Top row*, *middle*: The colorbar. Dark blue represents low values of the target parameter, whereas yellow represents high values. The opacity represents the uncertainty of the GP model, where an increase in uncertainty (here standard deviation, σ_A) is represented by an increase in transparency. The checkerboard background was added to support the visual interpretation of the opacity/transparency. The sections hatched in black indicate the uninteresting areas, which are lower than the threshold f_{th}. *Top row*, *right*: The simulated ground truth $f(x)$ in a 10 m × 10 m environment. *Bottom row*: The global path (in black on the *Left*). Three snapshot views at the beginning, after six iterations and at the end of the simulation with budget B = 80 m and budget factor $\lambda = 0.7$. The final model with the classification of the uninteresting areas (*Right*)

we want to quantify the gain of replanning online. For both evaluations, we used simulations in rectangular environments that are similar to the ones in the previous sections. However, since the specification of the region of interest is based on the measured values of $f(x)$, we need to simulate the process of acquiring measurements as well. We modeled the ground-truth distributions $f(x)$ using randomly drawn samples from a GP. A very simple example of such a ground-truth distribution is shown in Figure 8-*Top right*. The threshold f_{th} was set to the 75$^{\text{th}}$ percentile of the data range (Fig. 8-*Top middle*), such that $f(x)$ is higher than the threshold for 25% of the total area. To account for the uncertainties in estimating the hyperparameters of the GP, we used a modified set of hyperparameters for the ground truth process. The horizon distance h was set to 20% of the budget, and the traveling distance d_{tr} to 5% of the budget. This means that all runs consisted of 20 iterations, since 5% of the budget was consumed at each iteration. For all simulations, the start and end points were set to the lower left corner. This choice was made to meet typical field deployment requirements, where the robot is deployed and recovered at a predefined location.

The progression of the adaptive replanning scheme on a simplistic simulated environment is shown in Figure 8. The ground truth that was used to simulate measurements is shown in the top right corner. The graphs on the bottom row show (from left to right) the global path, the replanning scheme at the first, the sixth, and the last step and the posterior of the GP model. The hatched areas correspond to the classification of the uninteresting areas. All graphs show the global path in black and the planned path at the current iteration in purple. While the simulation progresses, the set of the collected samples (colored squares) along the traveled path (in gray) increases in size, which leads to a more accurate GP model. The posterior of the GP model (bottom right) shows that the classification of the uninteresting parts allows the algorithm to focus on the areas of interest by spending most of its extra path budget near the top-right corner of the environment.

We used two measures to evaluate the simulations quantitatively. We first describe the classification of the uninteresting areas, which leads to the detection of the areas of interest. We quantify this by the F_1-score using the comparison of the ground truth and the final classification according to Equation 15 on a dense grid of 50 × 50 points. The second measure is the difference in uncertainty $\Delta\sigma^2$ in the area of interest and in the rest of the total area to evaluate the exploiting behavior of the algorithm:

$$\Delta\sigma^2 = \frac{\bar{\sigma}^2(\mathrm{x\_}) - \bar{\sigma}^2(\mathrm{x}_I)}{\bar{\sigma}^2(\mathrm{x\_})}, \tag{16}$$

where $\bar{\sigma}^2(\cdot)$ evaluates the mean variance and $\mathrm{x\_}$ and x_I denote the uninteresting and interesting parts, respectively.

Ten random scenarios were used for the evaluation of the algorithm, which was run with a range of parameters. The two measures described above were averaged over the ten scenarios. The total budget B was set to 60, 80, 100, and 120 m, and the budget factor λ, which balances the static and adaptive components, was set to 0.6, 0.7, 0.8, and 0.9. Figure 9 shows the first, the sixth, and the last iterations of two different runs in one of the simulated environments. These two examples represent the extreme cases of the parameter set. The upper one is more static ($\lambda = 0.9$), leading to a path that is very similar to the global path. In combination with the limited total budget (B = 60 m), this gives a path that can only explore the area without exploiting the collected information. Given the limited additional budget, the algorithm is only able to make small modifications to the global path. In contrast, the second example has a larger budget (B = 120 m) and a high level of adaptivity ($\lambda = 0.6$). Consequently, the resulting path is able to, first, explore the area, and then, focus on the detected areas of interest.

The results for the F_1-score and the relative uncertainty reduction $\Delta\sigma$ are presented in Table I. The resulting F_1-scores lead to two observations (Table I-*Left*). First, the quality of the classification results

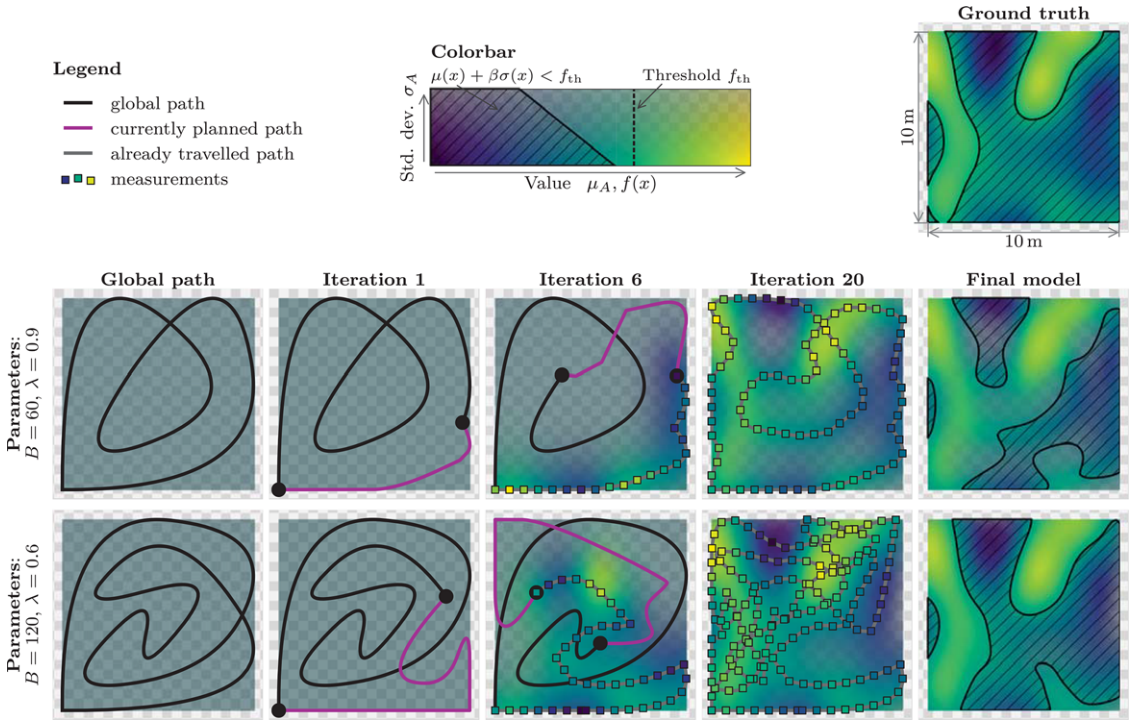

FIGURE 9 Graphic representation of two runs of the adaptive replanning scheme executed on a 2D environment that was used for the evaluation. *Top row, left*: The legend for the types of paths and measurement symbols. *Top row, middle*: The colorbar. Dark blue represents low values of the target parameter, whereas yellow represents high values. The opacity represents the uncertainty of the GP model, where an increase in uncertainty (here standard deviation, σ_A) is represented by an increase in transparency. The checkerboard background was added to support the visual interpretation of the opacity/transparency. The sections hatched in black indicate the uninteresting areas, which are lower than the threshold f_{th}. *Top, right*: The simulated ground truth $f(x)$ in a 10 m × 10 m environment. *Middle row*: The global path (in black) which is planned offline (*Left*). Three snapshot views at the beginning, after six iterations, and at the end of the simulation with budget B = 60 m and budget factor $\lambda = 0.9$. The final model with the classification of the uninteresting areas (*Right*). *Bottom row*: Similar, but with different parameters: $B = 120$, $\lambda = 0.6$

TABLE I Results of the evaluation of the adaptive replanning scheme. Darker shades of gray correspond to better results. *Left*: The mean of the F_1-scores over ten scenarios. *Right*: The mean of the uncertainty reduction $\Delta\sigma$ in the area of interest

F_1-score

Path budget B \ Budget factor λ	0.6	0.7	0.8	0.9
120	0.91	0.89	0.89	0.89
100	0.89	0.88	0.87	0.87
80	0.85	0.84	0.87	0.85
60	0.78	0.75	0.76	0.75

Uncertainty reduction $\Delta\sigma$

Path budget B \ Budget factor λ	0.6	0.7	0.8	0.9
120	0.23	0.25	0.22	0.094
100	0.16	0.16	0.15	0.071
80	0.11	0.1	0.04	-0.017
60	0.082	0.088	-0.024	0.019

improves with larger budgets. This is plausible because longer paths generally mean more measurements and better coverage of the total area. The fact that also the highest budget settings do not result in a perfect classification score of 1 is due to the modeled sensor noise. In areas where the function value $f(x)$ lies close to the threshold value f_{th}, the uncertainty bounds of $\beta\sigma(x)$ can preclude a good classification. The final configuration of the example shown in the right bottom row of Figure 9, exhibits such a situation, where there are many measurements but no complete classification. Second, and more surprisingly, the value of the budget factor λ only has a small effect on the resulting classification score. However, even in settings with small λ values, which allow for larger deviations from the global path, the goal is not the accurate classification but the detection of areas of interest. This is one of the essential differences between this present work and similar ones, which aimed at accurately estimating level sets.[11,54] While accurate knowledge about the position of level set boundaries is as well valuable, it often leads to dense accumulations of measurements along the boundaries (as they are the region of interest). However, it is often more desirable to distribute these measurements not only at the boundaries, but also within one of the sets, i.e., focus on a value-dependent region of interest.

Similar to the classification scores, the relative variance reduction $\Delta\sigma^2$ in the areas of interest clearly increases with larger budgets (Table I-*Right*). Again, this is related to the fact that longer paths simply yield more measurements. However, the results also show a slight dependency of the relative variance difference $\Delta\sigma^2$ on the budget

factor λ. If there is not much room for adaptations (larger λ) along the execution of the paths, the resulting uncertainty is similar in both of the areas of interest and the remaining parts (small relative variance reduction $\Delta\sigma^2$). However, with smaller λ values, the procedure is successful in focusing on the desired target areas and the variance in these sections is significantly lower (up to 25%) than in the remaining areas (corresponding to a large relative variance reduction $\Delta\sigma^2$).

The presented parameter evaluation shows that the desired behavior is indeed achieved, and that the planning algorithm successfully identifies areas of interest and is able to focus further on them. While both parameters play an important role, the difficulty to tune them varies. The total path budget B is usually defined by the platform or deployment constraints. The budget factor (λ), on the other hand, needs to be tuned manually. Our experience indicates that values between 0.6 and 0.75 lead to the best results.

6 | FIELD EXPERIMENTS

The CIPP algorithm was implemented on an ASV for 3D data collection during three field experiments in Lake Zurich. The platform and the experimental setup are first described. Then, field results are presented, and both robotics and limnological aspects are discussed.

6.1 | The platform and sampling area

Our main target application is the monitoring of cyanobacteria abundance in Lake Zurich. For this purpose, we employed *Lizbeth*, a catamaran-type ASV (see Fig. 1-*Right*), which can deploy a commercial sensor suite,* henceforth simply called *probe*. The probe is equipped with a sensor that measures the fluorescence of phycoerythrin, a red pigment that is typical of this cyanobacteria. Phycoerythrin fluorescence is used to estimate *P. rubescens* abundance, here expressed as relative fluorescence units (RFU). A typical high value in summer is around 9 RFU, but values of up to 25 RFU can also be observed.[3] The threshold value used during our experiment was set to 9 RFU.

The robot moved the probe between 7 m and 20 m depth in the water column with a winch system. This range is based on previous observations that *P. rubescens* is found between 7 m and 16 m depth in late summer and early fall.[3,56] The boat is able to localize itself via a GPS receiver, and the pressure sensor of the probe provides accurate depth readings. This setup allows the boat to control the position of the probe similar to an underwater vehicle, with the benefit of accurate localization coming from the surface vessel. The system is described in more detail in earlier work.[19]

We made several adjustments prior to the implementation on the system of the above presented methodology. First of all, the operation space for these experiments is 3D, as opposed to the 2D scenarios we used for evaluation in Section 5. Technically, this has no implication to CIPP, except for how coordinates of the control points are concatenated to an offspring vector (Equation 11). Thus, the planned paths define the desired motion of the sensor. To execute this motion, the boat performs the required horizontal movement and the winch is used to control the depth of the sensor accordingly. Because our ASV operates with a simple line-following controller, we used linear B-splines instead of higher order ones. This design choice of keeping the low level controller to deal with curvature was motivated by our parameter evaluation in Section 5.3 (recall Fig. 7). This results in sequences of straight lines, which can easily be followed by the controller of the boat. Furthermore, to allow for smooth and efficient operation in the field, we started to replan before arriving to the next replanning point (up to 2 minutes in advance), such that the newly planned path is available by the time the boat arrives to the target position. Otherwise, the boat would have to stand still once arrived to the replanning point, waiting for the planning algorithm to compute the new path. As the CIPP procedure can provide any-time solutions, this can easily be implemented. A similar *planning-ahead* scheme was used in earlier work.[11] The initialization for the global 3D path was done analog to the 2D version we used in the simulations presented in Section 5: From the starting point a straight line was used to the furthest corner of the environment and back to the starting point.

Three field experiments were conducted in August, September, and October 2015 (Table II). The sampling area was located close to the shore, a very heterogeneous area characterized by wave action and a sharp increase in the lake depth ranging from 25 m to 95 m as shown in Figure 10-*Right*. We hypothesize that the *P. rubescens* distribution would be impacted by these features, and thus we should see differences in abundance when moving away from the shore. Measurements were taken from 7 m to 20 m and, consequently, the sampling area had a cuboid shape of 350 m × 350 m × 13 m (Fig. 10-*Left*).

The total path budget for the complete mission was set to 4.5 km according to the capacity of the batteries of the boat. The budget allocated for the global path was 3 km, and the remaining 1.5 km was used for the local path adaptation by setting λ to 0.66. The planning horizon h was set to 1 km and the traveling distance d_{tr} to 250 m. A squared exponential kernel was defined for the GP model, with length scales of 150 m in x and y directions, respectively, and 1.5 m in vertical direction. These hyperparameters were optimized to fit earlier datasets taken during research works in both biology[3] and robotics.[11] A field experiment lasted on average 3 h 23 min during which the probe traveled between 4.6 km and 4.8 km within the water body. The total distances were slightly higher than the specified total budget of 4.5 km due to controller inaccuracies and occasional breaks to let other boats pass.

6.2 | Experimental results

First, we describe in depth the results of a single mission realized on August 3, 2015. Figure 11 shows the path of the probe during this first field experiment (*Left and middle*). The spheres depict the measurements that were collected along the path. Qualitatively,† the evolution of the paths shows two phases: an explorative phase and a focused phase. During the first phase of the experiment, the path lead the probe along the boundaries of the sampling area with zigzag motions to cover

* We used a YSI 6600 v2 probe, which is commonly used by limnologists.

† The temporal instability of the *P. rubescens* field prevents us from collecting ground truth information.

TABLE II Summary of the field deployments in Lake Zurich

Date	Starting time	Duration	Traveled distance*	Avg. air temperature**	Avg. wind speed**
August 3, 2015	10:23	3 h 41 min	4.7 km	25.7° C	1.97 m/s
September 21, 2015	10:48	3 h 32 min	4.8 km	16.1° C	1.75 m/s
October 27, 2015	11:39	2 h 57 min	4.6 km	9.8° C	1.50 m/s

*Distance traveled by the probe within the water body.
**Data source: http://www.tecson-data.ch/zurich/mythenquai/, accessed October 10, 2015.

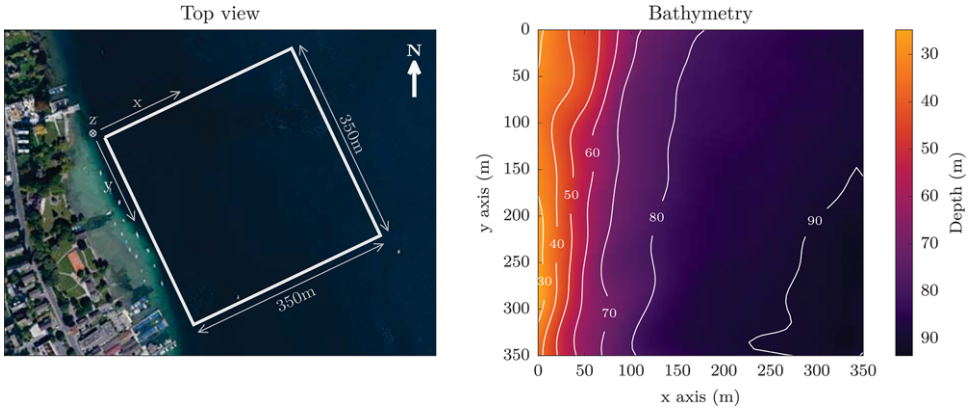

FIGURE 10 Overview of the field deployment environment. *Left*: Top view of Lake Zurich showing the sampling area where field trials were conducted. The local coordinate frame $\{x, y, z\}$ is indicated. Image courtesy: Google Earth, Image Landsat. View position: 47°19'14.90 N 8°33'22.49 E. *Right*: The bathymetry map of the testing area in local coordinates with isocontour lines in white

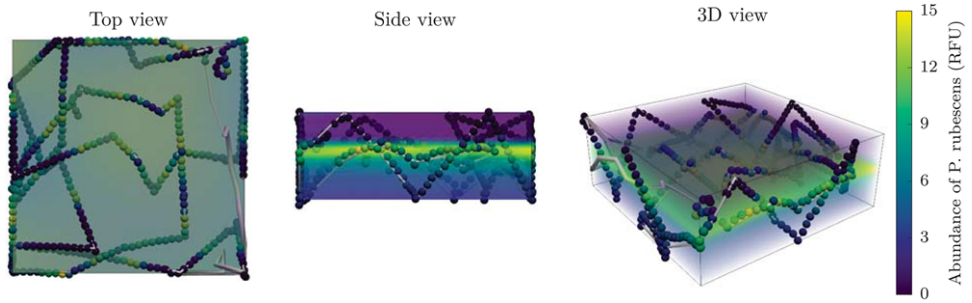

FIGURE 11 Views of the resulting path of the field experiment conducted in Lake Zurich on August 3, 2015. The planned path is shown in light gray. *P. rubescens* abundance is presented in relative fluorescence units (RFU) in the colorbar (*Left*). The spheres depict the measurements that were taken, and the cloudy volume shows the mean of the posterior GP regression model (low values are transparent for visualization purposes)

the entire depth range. In the course of this explorative phase, large portions of the space were classified as uninteresting due to low abundances of *P. rubescens*. Afterwards, the algorithm planned paths that focused on the layer of high cyanobacteria abundances, which reached a maximum of 15.2 RFU that day and was located at around 13 m of depth. The discrete RFU measurements of the probe are used in our GP regression model to provide a continuous, 3D, representation of the *P. rubescens* abundance, which is depicted in Figure 11. The set of slices in Figure 12-*Left* provides a more detailed representation of the GP posterior. The slices are equally positioned along the *x*-axis, as shown in Figure 12-*Right*.

The variance of the GP posterior is represented by transparency in Figure 12 and was used to estimate the region of interest. Outside of the layer of cyanobacteria, the posterior variance has only been reduced enough to be certain that the cyanobacteria abundance is lower than the threshold (hence, higher transparency in the graph). The planner successfully collected more measurements within the layer of *P. rubescens* (low transparency in the graph) than a simple coverage grid would have. The inspection of the GP posterior reveals that the *P. rubescens* layer was quite homogeneous and located between 12 m and 15 m depths. As mentioned in the introduction, the accumulation of cyanobacteria in the metalimnion is the typical distribution observed during the warm summer months.

The algorithm focused on the detected region of interest after having taken enough measurements to classify the bottom and top layers. During the last third of the sampling mission, 83% of the taken measurements had a value higher than the predefined threshold (i.e., within the area of interest), which corresponds to the low variance of the GP posterior within the layer of cyanobacteria. These results demonstrated the successful deployment of the presented algorithm and provided a novel point of view on the spatial distribution of *P. rubescens*.

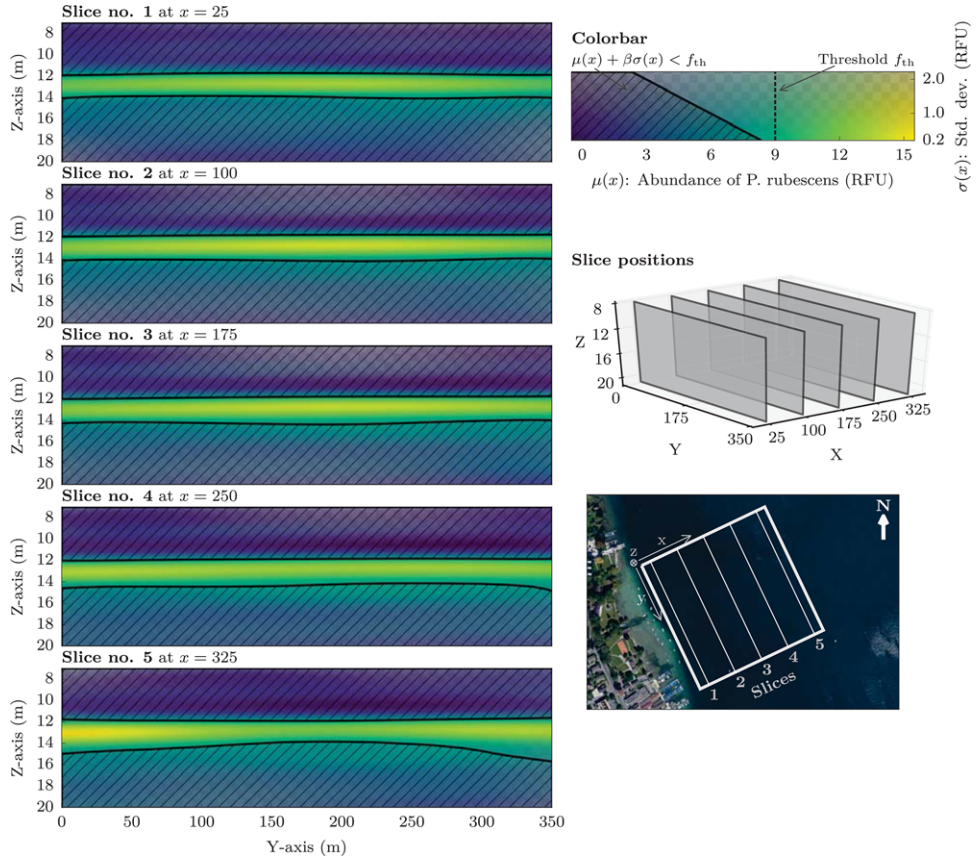

FIGURE 12 *P. rubescens* abundance during the field experiment conducted in Lake Zurich on August 3, 2015. *Right column*, *top*: The colorbar, indicating both the threshold value f_{th} and the actual classification boundaries; RFU stands for relative fluorescence units. As in Figures 8 and 9, the transparency depicts the uncertainty of the model, and the uninteresting areas are hatched with black lines. *Right column*, *middle*: 3D depiction of the slicing planes' positions. *Right column*, *bottom*: Top-view of the slicing planes in the sampling area. *Left column*: Slicing *yz*-planes, equally spaced by 75 m along the *x*-axis showing the resulting 3D GP model. The *z*-axis is the depth measured from the surface of the lake

Second, we present the overall results from the three-month study. When comparing the first field experiments to the ones conducted on September 21 and October 27, the seasonal evolution in *P. rubescens* abundance becomes evident (Fig. 13). *P. rubescens* is still located in the metalimnion in September, but the abundance is then maximal and reached 25.7 RFU. Mean abundances started from 4.4 RFU in August, reached 7.8 RFU in September and decreased to 5.1 RFU in October. During that last month, the thermal stratification was eroded due a decline in ambient temperature (Table II) combined with the stronger fall winds, which mixes the upper water column and thereby entrains *P. rubescens* from surface to a depth of 15 m. The 3D reconstructions computed with our CIPP algorithm accurately depicted the abundances and distribution of *P. rubescens* in Lake Zurich during summer and fall, as described previously.[15,57]

The 3D representations also allowed to discern new information on *P. rubescens* distribution because the spatial resolution reached with the ASV is much higher than when using traditional sampling methods. The finer resolution confirmed our hypothesis, and showed that there was a difference in *P. rubescens* distribution along a distance of 300 m (i.e., from slice no. 1 to slice no. 5) during the peak abundance in September. *P. rubescens* was more abundant near-shore, where the wave action was the strongest and the lake the shallowest (depth between 25 m to 52 m, Fig. 10-*Right*), and abundance decreased gradually away from the shore (Fig. 13). Contrary to what was expected, the waves did not interfere with the accumulation of *P. rubescens* in a dense layer. Two reasons could explain the higher biomass accumulation along the shore: first, runoffs from the land may have brought extra nutrients and/or other essential compounds to promote cyanobacterial growth; second, a basin-wide internal wave may have caused the buildup of biomass along the shore, as observed in July 2011.[3] This clearly demonstrates how a path planner that augments sampling coverage can help to generate new hypotheses and foster our understanding of complex natural phenomena.

Finally, having a path that focuses on the (initially unknown) areas of interest enables an appropriate distribution of measurements in the sampling area. It provides the means to maintain a minimal level of certainty in uninteresting areas, while maximizing the model certainty within the areas of interest. In comparison to statically planned paths, which focus on even coverage, more information about the regions of interest can be gained that way and less measurements are squandered within the uninteresting areas. The use of the uncertainty bounds for the classification of uninteresting areas prevents the algorithm from making too eager decisions and allows to deal with noisy sensor data.

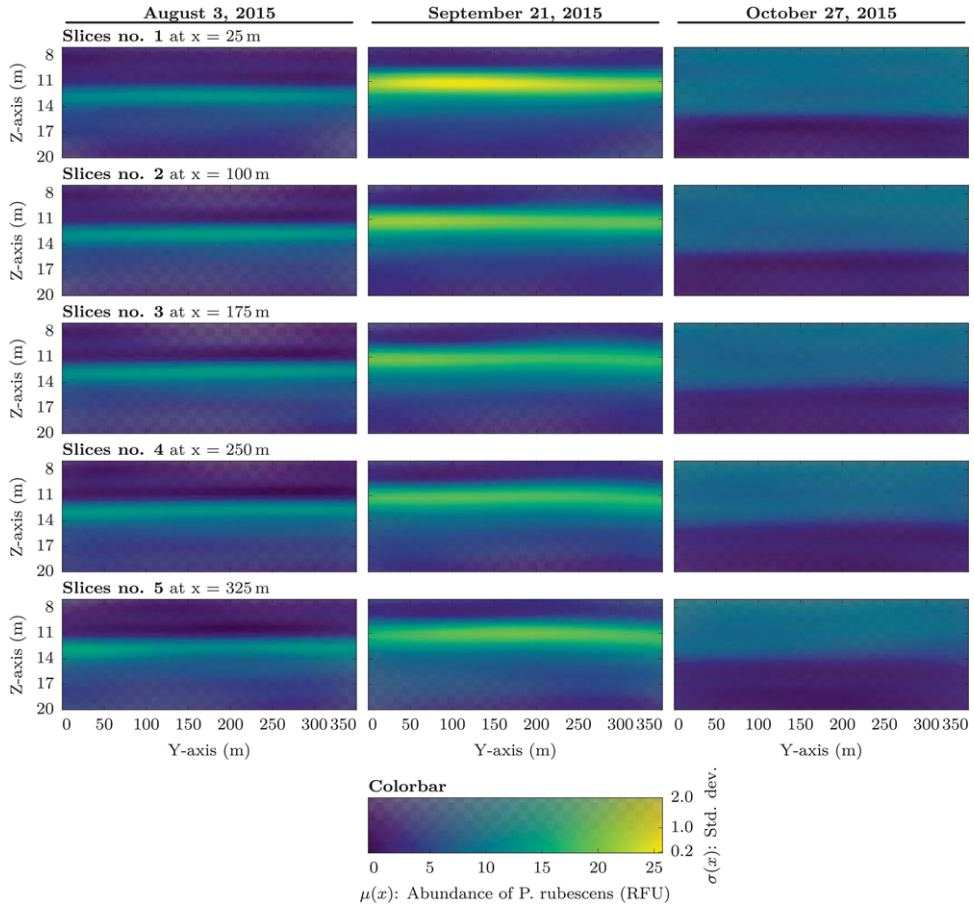

FIGURE 13 The resulting 3D GP models that illustrate *P. rubescens* abundance during the three field experiments conducted in Lake Zurich in summer and fall 2015. *P. rubescens* abundance is presented in RFU. As in Figures 8 and 9, the transparency depicts the uncertainty of the model. Slicing *yz*-planes are equally spaced by 75 m along the *x*-axis. The *z*-axis is the depth measured from the surface of the lake

7 | DISCUSSION

The following sections highlight considerations for parameter choices in field deployments, discuss the usage of CMA–ES as a global optimizer for IPP, and present extensions to the basic CIPP algorithm.

7.1 | Choosing appropriate parameters

In the light of the parameter study presented in Section 5 and the learnings from the field deployments, a discussion of the main parameters of the proposed planning method is presented in the following. Namely, the number of control points along a path, the path budget B, the replanning horizon h, the traveling distance d_{tr}, the confidence scale β and the budget factor λ are discussed.

Planning in continuous space The evaluation described in Section 5 has shown that planning informative paths in continuous space can provide better results than optimal solutions on graphs. For our comparisons, we have chosen a simple, yet popular, graph on a rectangular grid. Depending on the application and the shape of the environment, such a graph might not suffice. If motion constraints for the robot need to be encoded into the graph, for instance fixing rotation constraints by adding a dimension for heading, the size of the graph can grow dramatically quickly. When using large path budgets, this results in long computation times and, more importantly, in large trees of partial solutions that need to be stored in memory. In contrast, the CIPP approach circumvents this problem by propagating a limited generation of offsprings, which each represents a complete path from the start to the goal. However, while the B-spline representation defines a continuous path, its set of control points can be seen as an equivalent to the graph in discrete space approaches. Simulations have shown that below a certain minimal number of control points, the solution quality quickly deteriorates (see Fig. 7). While this minimal number is generally unknown, the results also indicate that the use of too many control points has no negative effect on the quality of the solution. Choosing too many control points increases the computation time, as it basically augments the dimensionality of the optimization problem. Designing a graph that is too dense can, on the other hand, have drastic consequences also for algorithms with sub-exponential computation times.[26]

Environment size We investigated the influence of the degree and the number of control points in the B-spline representation. These two parameters are indeed more critical if the environment $\mathcal{E}$ and the budget B is fixed. The budget for field applications is often determined by the maximal endurance of the robotic platform, which can be considered to be known and constant. The environment, however, is not necessarily predetermined. Especially in aquatic applications, the

environment is often restricted artificially to a certain subspace because considering full lakes or bays as sampling areas drastically exceeds manageable scales for the deployed robots. Thus, we restricted our field deployment to a cuboid within the lake. Similar restrictions were applied in other works.[13,26,29,58] While such restrictions can be justified by expert knowledge in certain cases, their choice is often dictated by the robots maximal energy capacity. This restriction can be reformulated to the question: What is the largest possible region of interest, within which a robot can collect meaningful scientific data given its path budget? There is no clear answer to this question, and the typical approach for finding appropriate parameters is to run simulations prior to the deployment. However, as a rule of thumb, the circumference of the environment should not be larger than the budget (meaning that the robot should at least be able to do a single round trip). For objectives that require adaptive replanning, the circumference should maximally be half the size of the budget, otherwise the robot will not be able to exploit the collected information.

Replanning horizon The adaptive replanning scheme, which was introduced above (Section 4), was evaluated with respect to the two main parameters, i.e., the total path budget B and the budget factor λ. However, there are two further, more application dependent parameters, which can have a strong impact on the results. The first of these two is the planning horizon h that dictates how short-sighted the local planner operates. The larger the horizon the better, and ideally the rest of the mission would be replanned. However, this is often infeasible in online applications, where computation times must usually be limited. The second parameter is the traveling distance d_{tr}, which defines the distance for which we travel on a locally planned path before replanning again. The choice of this parameter is less clear, because too small values result in only little information gains in between replanning runs. Furthermore, it can lead to greedy paths, as the local planner always plans over the entire horizon h and is not aware that only a very small fraction of it is executed. On the other hand, too large d_{tr} values might result in too static behavior, which ignores important collected information. As a vague rule of thumb, we have found that the traveling distance d_{tr} should be in the order of magnitude of the largest length scale of the kernel function of the GP, and the horizon h should roughly be 5–10 times the traveling distance.

Classifying regions of interest Identifying regions of interest based on a certain range values depends mostly on the confidence parameter β because it scales the confidence bounds that are used in the classification according to Equation 15. On the one hand, small values will lead to overconfident and inaccurate classification, and the remaining measurements will be spread only partially in the true regions of interest. On the other hand, too large values will result in uncertain classification and poor focus on the desired targets. In our evaluation, we have not analyzed the impact of β on the performance, but have rather adopted the concept of classification with confidence bounds from the works of Gotovos et al. and Srinivas et al.[53,54] (see their articles for more in-depth discussions on β). As for the design parameter λ (i.e., the budget factor), its value depends on how much one trust the global path. With a value of one, the algorithm will degenerate to the execution of a trajectory following exclusively the global path, whereas with a value of zero it will produce a trajectory independent of the global path. In our environmental monitoring application, the limnologists were able to estimate between which depths *P. rubescens* could be found because they manually sampled a single point a few days before. Our global path design relied on a uniform distribution of *P. rubescens* within those depth ranges. Given that our hypothesis was that the distribution of *P. rubescens* would be influenced by the shore, which covered a small percentage of the environment selected, a small flexibility was given to the algorithm ($\lambda = 0.66$) to deviate from its global path. In the case where the only data available by the limnologists would have been collected months before the autonomous survey, a lower value of λ would have been used to cope with a very approximate initial position of *P. rubescens* in the lake.

7.2 | Planning with CMA–ES

Optimization parameters While CMA–ES features several internal control parameters, optimal settings were presented in the literature[50] and only a few problem-specific parameters need to be set. In particular, these are the population size λ_{CMA} and the initial step size σ_{CMA}. Both of them have a significant influence on the results of the optimization.

The population size primarily defines the number of offsprings, which are drawn from the domain of the objective function. Hence, the higher the dimensionality of the problem the more offsprings are required.[59] demonstrate that choosing $\lambda_{\text{CMA}} \geq 4 + \lfloor 3 \ln(d) \rfloor$ is necessary for the convergence of the optimizer on a d-dimensional problem. Larger population sizes are able to sample the configuration space better and will, in general, yield better results. However, more offsprings per generation also correspond to more frequent calls to the objective function, which results in longer running times. The increase in running time is not necessarily linear, because fewer iterations might be necessary to reach convergence. On the problem at hand, we have found population sizes in the range of d to $2d$ to be a good trade-off.

The choice of the initial step size σ_{CMA} mainly controls the initial spread of the offsprings, i.e., how well the domain is covered. Setting it to a too small value might prohibit the optimizer from sufficiently exploring the optimization domain and might let it converge to a local optimum. However, its choice is application-dependent but is particularly related to the size of the environment $\mathcal{E}$ because it directly scales the covariance matrix, which lets the offsprings diverge from the initial path. Furthermore, anisotropic environments, such as the cuboid that we used in the field experiments, can be problematic. As the covariance matrix is initially isotropic (it is initialized as identity), most of (if not all) the offsprings exceed the boundaries (at least) along the shortest axes. If the chosen σ_{CMA} is too small, the environment might not be adequately explored along the longest axes. To circumvent this issue, a simple normalization of the optimization domain by the maximal extent of the environment has shown to be very effective. With such normalization, we were able to set the initial step size σ_{CMA} to values between 0.25 and 0.3 for both the simulations shown in Section 5 and the field results in Section 6.

Premature convergence It has been shown that CMA–ES converges to a solution in general.[50] However, there is no guarantee that an optimal solution will be found. In fact, the solution will always be

a local optimum, but could be an arbitrarily bad one. Such premature convergence is likely to occur if the initial scaling factor σ_{CMA} is too small. As already discussed above, the choice of a sufficiently large σ_{CMA} can be difficult. Here, we achieved reliable results by normalizing the search space. However, if the user is willing to invest more computation time to improve potentially bad solutions, several approaches are available. First, σ_{CMA} can be increased to larger values. Second, larger numbers of offspings λ_{CMA} can improve the solution quality. Finally, heuristics can be used. For example, one can increase the scaling factor σ_{CMA} after convergence and check if CMA–ES converges to the same optimum, or one can use multiple and varying initialization parameters.

Workspace topology The topology of the environment $\mathcal{E}$ where we define the IPP problem has an influence on the optimization result. In this work, we have only considered convex and obstacle-free environments. While we have introduced this constraint due to the initialization of the optimization routine (see Section 4), such environments are also easiest to handle during the optimization because the domain of feasible paths (i.e., paths which do not violate any of the constraints) is more homogeneous. In fact, it is unclear to what extent the CIPP can handle obstacles or highly non-convex environments. This would require further investigation, and environments with very complex topologies would require special treatment. Such treatment could include procedures to find reasonable and feasible initializations for CMA–ES or ways to avoid sampling many infeasible offsprings due to the non-homogeneous boundaries of the environment. One potential way of addressing these issues might be to split the environment into simpler sections. However, many real-world works on IPP have only considered simple environments[5,29,31,32,47,58] and also for the application of *P. rubescens* monitoring simple environment shapes (such as the cuboid used in this work) are completely sufficient.

7.3 | Extensions to CIPP

Even though we have not considered them in our evaluation and field deployment, we briefly present three extensions to the CIPP procedure to handle multi-robot applications, platform-specific constraints, and temporal constraints within the environment.

Multi-robot applications Considering that we have m robots, which are each given a budget B_i, $i = 1, \dots, m$ the goal is now to plan m paths. For this we can reformulate the objective function (Equation 12) as follows:

$$\bar{u}(\mathbf{p}) = u\left(\bigcup_{i=0}^{m} \text{SAMPLE}\left(P_i(\mathbf{p})\right), \mathcal{A}\right) \prod_{i=0}^{m} \left(g_{\text{budget}}(P_i(\mathbf{p}), B_i)\, g_{\text{boundary}}(P_i(\mathbf{p}), \mathcal{E}_i)\right) \tag{17}$$

To account for the submodularity of the objective function u, we evaluate the samples from all paths jointly and apply the penalty factors for the constraints on all paths. Note that the robots can have varying environments $\mathcal{E}_i$, which do not necessarily need to intersect. The offspring vector $\mathbf{p}$ is now built up by concatenating the control points of all paths:

$$\mathbf{p} = \text{vec}\left(\left[\mathbf{c}_1^1, \dots, \mathbf{c}_n^1, \mathbf{c}_1^2, \dots, \mathbf{c}_n^2, \mathbf{c}_1^m, \dots, \mathbf{c}_n^m\right]\right),$$

where $\mathbf{c}_j^i$ denotes the j-th control point of the path for the i-th robot. Start- and end-point constraints can be implemented analogously to the single robot case by simply omitting the corresponding points in the offspring vector.

Platform-dependent constraints Depending on the robotic platform that is used to collect the data, specific limitations might exist. One typical limitation is imposed by non-holonomic systems, which limits the curvature of the path. Some systems, for example fixed-wing aerial vehicles, might have even more restrictive constraints on feasible paths. Encoding such constraints in graphs can be difficult and usually implies the addition of dimensions, and thus, increases the complexity. For the CIPP approach, however, such constraints can be imposed directly using the derivatives of the continuous B-spline representation of the path. The constraint can then be included into the objective function by adding an additional penalty factor.

Time-dependent environments Depending on the application, it might be necessary to take into account the temporal dynamics of the environment. In aquatic habitats, the avoidance of rush hours on high traffic channels has been considered.[60] To incorporate such constraints into CIPP, the environment has to be described as a function of time $\mathcal{E} = \mathcal{E}(t)$ and the corresponding penalty term g_{boundary} needs to be evaluated at the time at which the robot passes the corresponding point on the path. For that we introduce the function TRAVEL(P, t), which describes the robot dynamics and provides the position of the robot after t:

$$g_{\text{boundary}}(P, \mathcal{E}) = \epsilon\left(\int_0^{t_{\max}} \bar{\phi}(\text{TRAVEL}(P, t), \mathcal{E}(t))dt\right).$$

8 | CONCLUSION

The novel CIPP algorithm detailed in this work is based on an ES technique that is used to optimize a parameterized path in continuous space. The ability to plan paths in continuous space rather than on discrete graphs has shown to provide competitive results, which can even outperform optimal paths on graphs in certain situations. In addition, planning in continuous space abolishes the difficulties of designing an adequate graph on which a path is planned. The optimization of the path is done with the CMA–ES method, which is a generic global optimization routine. Modifications specific to the problem at hand (such as using *k-opt* heuristics[61] to avoid the formation of loops) might provide further improvements and are thus an interesting avenue for future work.

The CIPP method was presented along with a replanning scheme that allows for online adaptation of a path as a reaction to newly acquired measurements. This way, the planner is able to both detect and focus on value-dependent regions of interest. The simulation results show that this allows to build models with lower uncertainty within these regions of interest. However, this selective acquisition of data requires the procedure to trade off explorative behavior, aiming at full covering the environment and exploitative focusing on the

regions of interest once they are detected. In the presented approach this trade-off is controlled by a design parameter, whose influence was studied in a parameter analysis. Our results show that local adaption to the path yield up to 25% lower variance within the regions of interest as long as the statically planned, prior path is reasonably covering the environment. Furthermore, we demonstrated that the number of control points of the path parameterization influences the performance of the planning method. The results allow us to conclude that a certain minimal number of control points is required to achieve good results. However, adding further points has no effect on the quality of the solution but only increases the complexity of the optimization problem. While we have chosen the number of control points manually for our field experiments, future work should focus on automatically determining this parameter. One potential solution would be to start with a low number of points, and iteratively add control points until no further improvement can be made.

The proposed planning routine has been formulated generically and could be applied to a variety of autonomous systems, such as a drone for gas distribution mapping (e.g., Ref. 5) or a rover for ground ice mapping (e.g., Ref. 6). In this work, the algorithm was implemented on a catamaran-type ASV for monitoring the 3D cyanobacteria distribution in Lake Zurich. The field results indicate that the algorithm can provide refined models within value-dependent regions of interest and orient limnology experts toward more precise research questions. Not only field data corroborated long-term observations, but they also provided new details to our knowledge of the spatial distribution of the toxic cyanobacteria in the lake. Such extensive, efficient, and adaptive data collection procedures foster the overall understanding of ecosystems and contribute to predict the behavior of processes occurring within them. In a near-future, autonomous robots could ease the production of environmental impact assessments while at the same time refine our understanding of the impacts of natural processes (e.g., volcanic eruptions) and anthropogenic-induced phenomena (e.g., climate change) on the environment and public health.

In our work, the replanning scheme was designed to focus on value-dependent regions of interest. Other targets, such as finding the global minimum or maximum of the field or estimating level sets, could also be implemented and might provide interesting avenues for future research. Our field results provided a new angle on the *P. rubescens* abundances in 3D space, but the model could be augmented considerably with more frequent and/or denser data collection. This could eventually allow to model the temporal dynamics of the abundances on a finer spatial scale. Estimates from such a model could be provided as prior knowledge to the planner. Ideally, continuous sampling missions would be carried out throughout the year, providing a constantly updated model of the abundance state. The GP model on which our work is based allows naturally for such extensions. While there are several challenges in scaling the GP model to handle the ever increasing data set, further research is necessary to provide a level of autonomy of the robot (in our case, including interaction with other lake traffic participants, autonomous energy management, etc.), which enables long-term environmental monitoring with reasonably low manpower.

ACKNOWLEDGMENTS

This work was funded by the Swiss National Science Fund (grant number CR2212-130023). F. Pomerleau was supported by a Postdoctoral Fellowships granted by the Natural Sciences and Engineering Research Council of Canada (NSERC). The authors wish to thank the Limnological Station of the University of Zurich, in particular Thomas Posch, Jakob Pernthaler, and Eugen Loher, for their support and for the generous provision of their technical equipment. Furthermore, the authors would like to thank Abel Gawel, Renaud Dubé, and Mark Pfeiffer for their invaluable help during field trials.

REFERENCES

1. Dunbabin M, Marques L. Robots for environmental monitoring: Significant advancements and applications. *IEEE Robot Autom Mag*. 2012;19(1):24–39.
2. Bork P, Bowler C, de Vargas C, Gorsky G, Karsenti E, Wincker P. Tara Oceans studies plankton at planetary scale. *Science*. 2015;348(6237):873–873.
3. Garneau M-E, Posch T, Hitz G, et al. Short-term displacement of *Planktothrix rubescens (cyanobacteria) in a pre-alpine lake observed using an autonomous sampling platform*. *Limnol Oceanogr*. 2013;58(5):1892–1906.
4. Dunbabin M, Grinham A. Experimental evaluation of an autonomous surface vehicle for water quality and greenhouse gas monitoring. In *IEEE International Conference on Robotics and Automation (ICRA)*. Anchorage, USA: 2010:5268–5274.
5. Neumann P, Asadi S, Lilienthal A, Bartholmai M, Schiller J. Autonomous gas-sensitive microdrone: Wind vector estimation and gas distribution mapping. *IEEE Robot Autom Mag*. 2012;19(1):50–61.
6. Barfoot TD, Furgale PT, Osinski GR, Ghafoor N, Williams KK. Field testing of robotic technologies to support ground ice prospecting in martian polygonal terrain. *Planet Space Sci*. 2010;58(4):671–681.
7. Muscato G, Bonaccorso F, Cantelli L, Longo D, Melita CD. Volcanic environments: Robots for exploration and measurement. *IEEE Robot Autom Mag*. 2012;19(1):40–49.
8. Ore J-P, Elbaum S, Burgin A, Detweiler C. Autonomous aerial water sampling. *J Field Robot*. 2015;32(8):1095–1113.
9. Nuske S, Choudhury S, Jain S, et al. Autonomous exploration and motion planning for an unmanned aerial vehicle navigating rivers. *J Field Robot*. 2015;1–22.
10. Griffith S, Pradalier C. A spatially and temporally scalable approach for long-term lakeshore monitoring. In *Proceedings of the 10th International Conference on Field and Service Robots (FSR-2015)*. Toronto, Canada: 2015:1–14.
11. Hitz G, Gotovos A, Pomerleau F, et al. Fully autonomous focused exploration for robotic environmental monitoring. In *IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China: 2014:2658–2664.
12. Smith RN, Das J, Heidarsson H, et al. USC CINAPS builds bridges: Observing and monitoring the southern California Bight. *IEEE Robot Autom Mag*. 2010;(Special issue on Marine Robotic Systems):20–30.
13. Smith RN, Schwager M, Smith SL, Jones BH, Rus D, Sukhatme GS. Persistent ocean monitoring with underwater gliders: Adapting sampling resolution. *J Field Robot*. 2011;28(5):714–741.
14. Shimoda Y, Azim ME, Perhar G, et al. Our current understanding of lake ecosystem response to climate change: What have we really learned from the north temperate deep lakes? *J Great Lakes Res*. 2011;37(1):173–193.

15. Posch T, Köster O, Salcher MM, Pernthaler J. Harmful filamentous cyanobacteria favoured by reduced water turnover with lake warming. *Nat Clim Change*. 2012;2(11):809–813.
16. Sivonen K, Jones G. Cyanobacterial toxins. In: *Toxic Cyanobacteria in Water: A Guide to Their Public Health Consequences, Monitoring and Management*. London: E & FN Spon; 1999:41–111.
17. Codd GA, Morrison LF, Metcalf J. Cyanobacterial toxins: Risk management for health protection. *Toxicol Appl Pharm*. 2005;203(3):264–272.
18. Walsby AE, Schanz F. Light-dependent growth rate determines changes in the population of *Planktothrix rubescens over the annual cycle in Lake Zürich, Switzerland*. *New Phytologist*. 2002;154:671–687.
19. Hitz G, Pomerleau F, Garneau M-E, et al. Autonomous inland water monitoring: Design and application of a surface vessel. *IEEE Robot Autom Mag*. 2012;19(1):62–72.
20. Dijkstra EW. A note on two problems in connexion with graphs. *Numer Math*. 1959;1:269–271.
21. Krause A, Guestrin C. Submodularity and its applications in optimized information gathering. *ACM T Intell Syst Technol (TIST)*, 2011;2(4):32.
22. Krause A, Singh A, Guestrin C. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *J Mach Learn Res*. 2008;9:235–284.
23. Ko C-W, Lee J. An exact algorithm for maximum entropy sampling. *Oper Res*. 1995;43(4):684–691.
24. Nemhauser GL, Wolsley LA, Fisher ML. An analysis of approximations for maximizing submodular set functions. *Math Program*. 1978;14:265–294.
25. Chekuri C, Pal M. A recursive greedy algorithm for walks in directed graphs. In *IEEE Symposium on Foundations of Computer Science (FOCS)*. Pittsburgh: USAP; 2005:245–253.
26. Binney J, Krause A, Sukhatme GS. Optimizing waypoints for monitoring spatiotemporal phenomena. *Int J Robot Res*. 2013;32(8):873–888.
27. Binney J, Sukhatme GS. Branch and bound for informative path planning. In *IEEE International Conference on Robotics and Automation (ICRA)*. St. Paul, USA: IEEE; 2012:2147–2154.
28. Karaman S, Frazzoli E. Sampling-based algorithms for optimal motion planning. *Int J Robot Res*. 2011;30(7):846–894.
29. Hollinger GA, Sukhatme GS. Sampling-based robotic information gathering algorithms. *Int J Robot Res*. 2014;33(9):1271–1287.
30. Singh A, Ramos F, Whyte HD, Kaiser WJ. Modeling and decision making in spatio-temporal processes for environmental surveillance. In *IEEE International Conference on Robotics and Automation (ICRA)*. Anchorage, Alaska: 2010:5490–5497.
31. Zhang B, Sukhatme GS. Adaptive sampling for estimating a scalar field using a robotic boat and a sensor network. In *IEEE International Conference on Robotics and Automation (ICRA)*. Roma, Italy: 2007:3673–3680.
32. Yilmaz NK, Evangelinos C, Lermusiaux PFJ, Patrikalakis NM. Path planning of autonomous underwater vehicles for adaptive sampling using mixed integer linear programming. *IEEE J Oceanic Eng*. 2008;33(4):522–537.
33. Meliou A, Krause A, Guestrin C, Hellerstein JM. Nonmyopic informative path planning in spatio-temporal models. In *National Conference on Artificial Intelligence, volume 22*. Canada: Vancouver; 2007:602–607.
34. Singh A, Krause A, Kaiser WJ. Nonmyopic adaptive informative path planning for multiple robots. In *International Joint Conference on Artificial Intelligence (IJCAI)*. Pasadena, USA: 2009:1843–1850.
35. Girdhar Y, Dudek G. Modeling curiosity in a mobile robot for long-term autonomous exploration and monitoring. *Auton Robot*. 2015:1–12.
36. Lim ZW, Hsu D, Lee WS. Adaptive informative path planning in metric spaces. *Int J Robot Res*. 2016;35(5):585–598.
37. Yamauchi B. A frontier-based approach for autonomous exploration. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*. 1997:146–151.
38. LaValle SM. *Planning Algorithms*. Cambridge: Cambridge University Press; 2006.
39. Charrow B, Kahn G, Patil S, et al. Information-theoretic planning with trajectory optimization for dense 3D mapping. *Robot: Sci Syst (RSS) XI*, 2015a;3–12.
40. Charrow B, Liu S, Kumar V, Michael N. Information-theoretic mapping using Cauchy-Schwarz quadratic mutual information. In *IEEE International Conference on Robotics and Automation (ICRA)*. Washington: 2015b:4791–4798.
41. Julian BJ, Karaman S, Rus D. On mutual information-based control of range sensing robots for mapping applications. *Int J Robot Res*. 2014;33(10):1375–1392.
42. Heng L, Gotovos A, Krause A, Pollefeys M. Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments. In *IEEE International Conference on Robotics and Automation (ICRA)*. Seattle: 2015:1071–1078.
43. Witt J, Dunbabin M. Go with the flow: optimal path planning in coastal environments. In *Proceedings of the Australasian Conference on Robotics & Automation (ACRA)*. Canberra, Australia: 2008:1–9.
44. Berglund T, Brodnik A, Jonsson Hk, Staffanson M, Söderkvist I. Planning smooth and obstacle-avoiding B-spline paths for autonomous mining vehicles. *IEEE T Autom Sci Eng*. 2010;7(1):167–172.
45. Connors J, Elkaim G. Analysis of a spline based, obstacle avoiding path planning algorithm. In *IEEE Vehicular Technology Conference*. Melbourne, Australia: 2007:2565–2569.
46. Jung D, Tsiotras P. On-line path generation for unmanned aerial vehicles using B-spline path templates. *J Guid Control Dyn*. 2013;36(6):1642–1653.
47. Choi H-L, How JP. Continuous trajectory planning of mobile sensors for informative forecasting. *Automatica*. 2010;46(8):1266–1275.
48. Marchant R, Ramos F. Bayesian optimisation for informative continuous path planning. In *IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China: 2014:6136–6143.
49. Rasmussen CE, Williams CKI. *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: The MIT Press; 2006.
50. Hansen N. The CMA evolution strategy: A comparing review. In: Lozano JA, Larra naga P, Inza In, Bengoetxea E, eds. *Towards a New Evolutionary Computation - Advances in the Estimation of Distribution Algorithms, volume 192 of Studies in Fuzziness and Soft Computing*. Springer Berlin / Heidelberg; 2006:75–102.
51. Cover TM, Thomas JA. Elements of information theory. Hoboken, New York: John Wiley & Sons; 2006.
52. De Boor C. *A Practical Guide to Splines. Mathematics of Computation*. New York: Springer; 1978.
53. Srinivas N, Krause A, Kakade SM, Seeger MW. Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. *IEEE T Inform Theor*. 2012;58(5):3250–3265.
54. Gotovos A, Casati N, Hitz G, Krause A. Active learning for level set estimation. In *International Joint Conference on Artificial Intelligence (IJCAI)*. Beijing, China: 2013:1344–1350.
55. Brochu E, Cora VM, de Freitas N. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR*. 2010; abs/1012.2599.
56. Vanden Wyngaert S, Salcher MM, Pernthaler J, Zeder M, Posch T. Quantitative dominance of seasonally persistent filamentous cyanobacteria (*Planktothrix rubescens) in the microbial assemblages of a temperate lake*. *Limnol Ocenanogr*. 2011;56(1):97–109.

57. Garneau M-È, Posch T, Pernthaler J. Seasonal patterns of microcystin producing and non-producing *Planktothrix rubescens* genotypes in a deep prealpine lake. *Harmful Algae*. 2015;50:21–31.
58. Hollinger GA, Mitra U, Sukhatme GS. Active and adaptive dive planning for dense bathymetric mapping. In *Experimental Robotics, volume 88 of Springer Tracts in Advanced Robotics*. Springer International Publishing; 2013:803–817.
59. Hansen N, Ostermeier A. Completely derandomized self-adaptation in evolution strategies. *Evolu Computation*. 2001;9(2):159–195.
60. Binney J, Krause A, Sukhatme G. Informative Path Planning for an Autonomous Underwater Vehicle. In *IEEE International Conference on Robotics and Automation (ICRA)*. Anchorage, Alaska: 2010:4791–4796.
61. Lin S, Kernighan BW. An effective heuristic algorithm for the traveling-salesman problem. *Oper Res*. 1973;21(2):498–516.

How to cite this article: Hitz G, Galceran E, Garneau M, Pomerleau F, Siegwart R. Adaptive continuous-space informative path planning for online environmental monitoring. *J Field Robotics*. 2017;34:1427–1449. https://doi.org/10.1002/rob.21722