Learning-Based Active Sampling and Modeling of Aquatic Environments

Julie Alhosh

School of Computer Science
McGill University
Montreal, Quebec, Canada

December 16, 2024

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Science

Abstract

This thesis presents a novel method for active sampling in aquatic environments, aimed at efficiently reconstructing water quality maps with a limited number of samples. Our approach leverages Deep Q-Networks (DQN) for intelligent path planning, combined with Gaussian processes (GPs) for posterior map reconstruction. This framework allows the autonomous system to navigate unknown regions, intelligently decide where to take samples, and generate accurate environmental models with fewer samples compared to traditional methods. We evaluated our method using both a freely available sea surface temperature dataset from the National Oceanic and Atmospheric Administration (NOAA) and real-world data collected at Lac Hertel using our autonomous surface vehicle (ASV), BlueBoat platform, equipped with a temperature sensor. Our method learns adaptive paths that enable an accurate map reconstruction with fewer samples than all baselines, boustrophedon, highest-uncertainty, and model predictive control (MPC). The results indicate that our approach not only reduces the time and energy required for mapping but also maintains high accuracy in predicting unsampled areas. Our method represents a promising step toward more efficient and accurate environmental monitoring in aquatic ecosystems, contributing to both scientific research and environmental conservation efforts.

Abrégé

Cette thèse présente une nouvelle méthode d'échantillonnage actif dans les environnements aquatiques, visant à reconstruire efficacement les cartes de qualité de l'eau avec un nombre limité d'échantillons. Notre approche s'appuie sur des «Deep Q-Networks (DQN)»pour une planification intelligente des trajectoires, combinées à des processus gaussiens (GP) pour la reconstruction de cartes a posteriori. Ce cadre permet au système autonome de naviguer dans des régions inconnues, de décider intelligemment où prendre des échantillons et de générer des modèles environnementaux précis avec moins d'échantillons que les méthodes traditionnelles. Nous avons évalué notre méthode en utilisant à la fois un ensemble de données de température de surface de la mer disponible de la «National Oceanic and Atmospheric Administration (NOAA)»et des données réelles collectées au lac Hertel à l'aide de notre véhicule de surface autonome (ASV), la plateforme BlueBoat, équipée d'un capteur de température. Notre méthode apprend des chemins adaptatifs qui permettent une reconstruction cartographique précise avec moins d'échantillons que toutes les méthodes préexistantes, le boustrophédon, l'incertitude la plus élevée et la commande prédictive (MPC). Les résultats indiquent que notre approche permet non seulement de réduire le temps et l'énergie nécessaires à la cartographie, mais aussi de conserver une grande précision dans la prédiction des zones non échantillonnées. Notre méthode représente une étape prometteuse vers une surveillance environnementale plus efficace et plus précise des écosystèmes aquatiques, contribuant à la fois à la recherche scientifique et aux efforts de conservation de l'environnement.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Prof. David Meger, for his guidance, support, and encouragement throughout the course of my research. Your insightful feedback, expertise, and patience have been invaluable to me. I am especially thankful for the opportunity to be part of the Mobile Robotics Lab (MRL), for helping me choose a meaningful research project, and for your kindness and mentorship in seeing my thesis through to completion.

I would also like to thank my friend and colleague, Jean-François Tremblay for his helpful advice, invaluable discussions, and patience. I am grateful for his willingness to always answer my (many) questions and for his continuous support. I would like to express my appreciation to my friend and colleague, Harley Wiltzer, for his advice and assistance throughout my research. I am truly grateful for his contributions.

I am very grateful for all my friends at MRL, Jean-François, Charlotte, Hanna, Wei-Di, Harley, Lucas, Scott, Ellen, Amin, Stanley, Louis, Farnoosh, and Khalil. Thank you for being great friends. I will miss our discussions (the normal and crazy ones) over lunch.

Last but certainly not least, I owe a profound debt of gratitude to my family. To my parents and sister, your unconditional love and support have been my foundation and I am incredibly thankful for your understanding and encouragement. Thank you for always being there for me.

This thesis would not have been possible without the contributions and support of each and every one of you. Thank you from the bottom of my heart.

Contents

List of Figures

List of Algorithms

List of Acronyms

ASV: **A**utonomous **S**urface **V**ehicle
AUV: **A**utonomous **U**nderwater **V**ehicle
CNN: **C**onvolutional **N**eural **N**etworks
DQN: **D**eep **Q**-**N**etworks
DRL: **D**eep **R**einforcement **L**earning
GP: **G**aussian **P**rocess
MARLAS: **M**ulti-**A**gent **R**einforcement **L**earning for **A**daptive **S**ampling
MASP: **M**ulti-robot **A**daptive **S**ampling **P**roblem
MDP: **M**arkov **D**ecision **P**rocess
ML: **M**achine **L**earning
MLP: **M**ulti-**L**ayer **P**erceptron
MPC: **M**odel **P**redictive **C**ontrol
NOAA: **N**ational **O**ceanic and **A**tmospheric **A**dministration
POMDP: **P**artially **O**bservable **M**arkov **D**ecision **P**rocess
RBF: **R**adial **B**asis **F**unction
RL: **R**einforcement **L**earning
RMSE: **R**oot **M**ean **S**quared **E**rror
SE: **S**quared **E**xponential
UAV: **U**nmanned **A**erial **V**ehicle

1

Introduction

In recent years, advancements in robotics and machine learning (ML) have opened up new possibilities for large-scale environmental monitoring. Autonomous robots are capable of gathering vast amounts of environmental data over extensive areas with significantly less manual effort compared to traditional methods. This development is particularly important in the context of ecological preservation, where real-time and high-resolution data is crucial for understanding and managing ecosystems. Robotics-based environmental monitoring offers a more efficient, scalable, and adaptable solution to challenges that would otherwise require intensive human intervention and resources.

A key application of robotics-based environmental monitoring is in aquatic environments, where understanding the spatial and temporal dynamics of water quality is essential for ecosystem management. Conventional data collection methods, such as manual sampling or deploying stationary sensors, are often limited by high costs, slow sampling rates, and incomplete coverage. For instance, measuring water temperature or pollutant levels in a lake typically requires sending field personnel to physically sample the water at various locations, a process that is both time-consuming and labor-intensive. Moreover, these methods often fail to capture the dynamic nature of water systems, which can experience changes in conditions due to weather or human impact. Autonomous robots, equipped with advanced sensors and

intelligent path planning algorithms, provide a transformative approach to this challenge by enabling continuous and adaptive environmental monitoring.

One of the most exciting aspects of autonomous environmental monitoring is the ability to perform adaptive sampling, where a robot not only collects data but also optimizes its sampling strategy in real-time. This involves the robot deciding where and when to sample based on the current knowledge of the environment and its own operational constraints. By doing so, the robot can prioritize regions of high interest (such as areas with rapid changes in temperature or pollution) and reduce the number of redundant samples taken from areas where conditions are relatively stable. This leads to more efficient use of the robot's resources (e.g., battery power, memory) while maximizing how informative the collected data is.

However, developing an efficient adaptive sampling strategy is a non-trivial task. The problem is inherently complex because the robot operates in an unknown environment, where both the state of the environment and the robot's observations evolve between deployments. As a result, traditional static sampling strategies, which predefine a set of sampling locations based on prior knowledge, are not effective. Instead, a more adaptive and intelligent approach is required, one that allows the robot to learn and adapt its sampling strategy based on live sensor measurements.

One promising solution to this problem comes from the field of reinforcement learning (RL). RL is a machine learning paradigm that focuses on how agents can learn to make decisions by interacting with an environment. The agent learns by receiving feedback from the environment in the form of rewards, which guide it toward optimal behaviors over time. In the context of adaptive sampling, the robot acts as an RL agent, where its goal is to maximize how informative the data it collects is while minimizing the number of samples required. This balance between exploration (sampling areas to gather new information) and exploitation (focusing on areas that are known to be important) is a fundamental challenge in RL and is critical for effective environmental monitoring.

In this work, we focus on the problem of efficient adaptive sampling of unknown aquatic environments using autonomous robots. The primary objective is to develop an exploration policy that enables a robot to intelligently navigate and map water qualities in unknown environements while minimizing the number of samples required. Specifically, we aim to optimize the robot's path planning and sampling strategy to collect data that is most informative for creating a spatial map of water quality, such as temperature or pollutant levels, over a given area. At each step, the information collected by the robot is used for posterior

inference, enabling the estimation of a complete map of the water quality along with its uncertainty. Multiple factors, including the estimated map and its uncertainty, are then considered to determine the next sampling location.

A concrete example of this problem arises in mapping water surface temperature in lakes. Water temperature is a key indicator of ecosystem health and can have significant effects on nutrient cycling, water quality, and aquatic species distribution [35]. Despite its importance, high-resolution temperature data in many lakes is often sparse [16] due to the limitations of manual sampling methods. This is especially problematic in regions where water temperature can fluctuate rapidly in response to external factors such as weather patterns. By deploying an autonomous robot equipped with temperature sensors and an RL-based path planner and sampling strategy, it becomes possible to monitor and map temperature distributions over time, providing valuable insights into the dynamics of the aquatic ecosystem.

This problem is further complicated by the fact that water temperature distributions can exhibit significant spatial heterogeneity. For example, areas near river inflows may be significantly cooler than open water areas, while regions exposed to sunlight may experience higher temperatures. To efficiently map such complex spatial patterns, the robot must carefully balance the need to explore unknown areas with the need to focus on areas where rapid temperature changes are likely to occur.

In traditional RL, an agent interacts with an environment defined by a state space, a set of actions, and a reward function. At each timestep, the agent observes the current state of the environment, selects an action, and receives a reward based on the outcome of that action. Over time, the agent learns a policy that maximizes the cumulative reward it can obtain. In the case of adaptive sampling, the environment is the unknown aquatic region being monitored, the states are the robot's current position and any other relevant environmental variables, and the actions correspond to the robot's movement and sampling decisions.

One of the key advantages of using RL for this task is that the robot can learn from experience. As the robot explores its environment and collects samples, it builds an approximate map of the water quality distribution, which it can use to make more informed decisions about where to sample next. This allows the robot to adapt its sampling strategy based on live sensor measurements, taking into account both the data it has already collected, the estimated map, and the uncertainty about unobserved areas. The RL framework also allows for long-term planning, enabling the robot to take actions that may not provide immediate

rewards but are expected to lead to better outcomes in the future. For example, the robot might choose to sample in a relatively uninformative region if doing so will allow it to reach a more informative region more efficiently later on. It is important to note that while previous work has focused on using RL for sampling and path planning—for instance, Manjanna et al. [23] employed RL to plan efficient paths to sample regions of interest based on an existing environmental map—to the best of our knowledge, our method is the first to use RL for adaptive sampling and path planning in an *unknown* environment, relying solely on sensor data collected during the current mission.

One of the key challenges in adaptive sampling is modeling the uncertainty of the environment, as the robot must decide where to sample based not only on the data it has already collected but also on the uncertainty associated with the unobserved regions. To address this, we employ Gaussian processes (GPs), a powerful non-parametric model that can provide a full probabilistic description of the environment. Previous work in mapping and exploration has utilized GPs to aid in navigation and localization, enabling robots to efficiently map and explore unknown environments [11, 1, 40]. They allow the robot to estimate not only the expected value of the water quality at unobserved locations but also the uncertainty associated with those estimates. This is crucial for guiding the robot's sampling strategy, as it can prioritize areas with high uncertainty where new data is likely to be most informative.

GPs provide a Bayesian framework for function estimation, meaning that they can incorporate prior knowledge about the function and update this knowledge as new data is observed. The result is a posterior distribution that provides both a point estimate of the function and a measure of uncertainty. The GP's ability to capture correlations between different points in the input space allows the robot to infer the value of the environmental variable at unobserved locations based on the samples it has already collected. This property makes GPs particularly useful for adaptive sampling tasks, as the robot can focus on areas where the uncertainty is highest, thereby improving the efficiency of the sampling process.

In this thesis, we present two contributions to the field of autonomous environmental monitoring using RL and GPs. First, we develop the first method to efficiently and adaptively sample unknown aquatic environments, leveraging RL and GPs to optimize the robot's sampling strategy based on live sensor readings. Our method is designed to minimize the number of samples required while maximizing the informativeness of the collected data, providing a high-resolution map of water quality over a given area. Second, we evaluate our method in both simulated and real-world settings, demonstrating its effectiveness in

navigating and mapping dynamic aquatic environments.

This method is not limited to mapping surface temperature; it readily supports the integration of various sensor types, making it highly versatile for monitoring different water quality indicators. For instance, different sensors could be used to monitor parameters such as dissolved oxygen, salinity, and pH, in addition to surface temperature. This flexibility means that, once equipped with the appropriate sensors, the ASV can provide critical information on a wide range of environmental conditions, offering a powerful tool for comprehensive aquatic environmental monitoring.

Moreover, the potential impact of this work extends beyond environmental monitoring. The methods and techniques developed here could be applied to a wide range of domains, from precision agriculture to disaster response, where autonomous robots are required to efficiently gather data in large, dynamic environments. By combining RL, adaptive sampling, and probabilistic modeling, we aim to provide a robust and scalable solution for real-time environmental monitoring that can adapt to the ever-changing conditions of natural ecosystems.

In the following chapters, we first review the related work, positioning our research within the broader field of using robotics and RL for environmental monitoring. Then, we introduce the essential theoretical concepts, including GPs and key RL foundations in chapter 3. In chapter 4, we clearly define the problem, describe the planning challenge, and detail our solution using RL for path planning and posterior inference for modeling the environment. Chapter 5 outlines the simulation environment and datasets—both procedurally generated and real-world data—used to train and evaluate our models. This chapter also introduces the baselines, such as Boustrophedon, Highest Uncertainty, and MPC, against which our models are compared. In chapter 6, we present findings from our experiments, highlighting the performance of our proposed method. The thesis concludes with chapter 7, reflecting on the contributions, limitations, and potential directions for further research.

2

Related Work

The use of robotics in environmental sensing has seen substantial exploration and development in recent years, with significant advancements made in a range of domains. In their comprehensive review, Dunbabin and Marques [4] and others [39, 3] discuss the progress made in environmental robotics, with a particular emphasis on marine applications where ASVs and autonomous underwater vehicles (AUVs) have been widely utilized. These systems have been employed for tasks such as habitat mapping, detecting and localizing underwater pollution sources, and monitoring environmental changes. Despite being the most advanced in terms of vehicle design and scientific application, marine based sytems face several critical challenges, such as ensuring reliability and safety and improving mission and task planning. Addressing these challenges significantly enhance scientific data collection and deepen our understanding of environmental processes.

Shkurti et al. [32] discuss the deployment of multi-robot systems in environmental monitoring which offers numerous advantages, particularly in enhancing coverage and accelerating data collection. These systems are highly beneficial in large and rapidly changing environments, where conditions can shift significantly over short periods. By employing multiple robots, researchers can optimize sampling paths and reduce prediction errors, thus improving the overall efficiency of data collection. In the paper, the authors use a multi-robot

team comprised three vehicles, an unmanned aerial vehicle (UAV), an ASV, and an AUV, allowing for efficient multi-scale imaging of selected sites. The UAV was responsible for providing up-to-date broad-scale aerial imagery, to help identifying potential sites of interest, the ASV is used for caching the waypoints it receives from the UAV and sending them to the resurfaced AUV, while the AUV performed close-up inspections of these sites, focusing on areas suspected to have high biodiversity. This robotic team is designed to assist scientists in conducting consistent and repeatable environmental monitoring experiments. Specifically, the authors tested this multi-robot team to automate the process of coral reef inspection, enabling scientists to shift from the time-consuming task of manual data collection to the high-level identification of sites that require inspection.

Hansen et al. [9] present an approach to overcome marine surveying problem over large areas. They used an ASV and a set of drifters which are inexpensive floating sensors that do not use actuators to move but instead are propelled by the ambiant flow field. These sensors are deployed to drift with ocean currents, collecting data as they move. The heterogeneous sensing system, comprised the ASV and drifters, is used to sample and estimate the flow field of an area. It is an iterative process of collecting all the flow field samples from the drifters that are deployed, pick new possible deploying locations for drifters, predict the trajecotries of the drifters when deployed at these locations based on the current estimate of the flow field, decide how useful each trajectory is in terms of its expected reward, find the best possible paths needed to reach the deployment locations, then navigate towards the deployment location that is picked based on the expected reward and the best corresponding path, and deploy the drifter.

While passive systems offer a degree of autonomy, active robotic systems like ASVs and AUVs are capable of more directed and targeted data collection, making them more suitable for tasks that require precise control over sampling locations. For instance, Binney et al. [2] explore offline path planning strategies for AUVs called gliders, which are low power AUVs that control their depth by changing their ballast and can move forward using small wings on both sides. The paper introduces a path planning strategy for underwater gliders that aimed at reducing the overall uncertainty of scalar fields that the robots are tasked with mapping using an enhanced recursive-greedy algorithm. The method generates near-optimal paths while avoiding high-traffic areas. While more efficient than exhaustive search, the algorithm struggles with large datasets and scaling, but using rough waypoint discretization helps manage areas up to a few square kilometers. To improve path accuracy, they use the Regional Ocean Modeling System (ROMS) to model ocean currents since travel time between waypoints depends not only on distance but also on ocean currents.

Adaptive sampling techniques, which allow robots to focus their data collection efforts on areas of interest where the sensed spatial field shows certain critical behaviour, have been developed to further improve the efficiency of robotic exploration. Hitz et al. [10] have demonstrated the effectiveness of adaptive sampling in large environments, where robots can dynamically adjust their sampling paths based on the data they collect in real-time. This approach minimizes both the number of samples needed and the overall prediction error, making it ideal for environments where conditions change frequently. They focus on using autonomous adaptive sampling to detect critical behaviour seen in the sensed spatial field such as exceeding a given threshold value. They achieve this by introducing a receding horizon path planner, LSE-DP algorithm, which plans efficient sampling paths non-myopically with the goal of reducing the uncertainty everywhere and more specifically around the critical behaviour. They use an ASV to apply this algorithm to monitor a toxic cyanobacteria in a lake which allowed biologists to focus on the part of the lake that exhibits the critical behaviour only.

Low et al. [20] focus on mapping environmental phenomena using multi-robot adaptive sampling to achieve wide-area coverage and hotspot sampling. The concept of hotspot sampling, where robots focus on regions with high environmental variability or relevance, is especially important in large environments and terrains containting a few small hotposts where traditional exhaustive coverage approaches become impractical. The authors introduce the Multi-robot Adaptive Sampling Problem (MASP) as a dynamic programming problem and present an adaptive multi-robot exploration strategy. They also provide theoretical analysis of MASP with varying adaptivity which shows that we can reduce the spatial mapping uncertainty when using a more adaptive strategy.

Manjanna and Dudek [22] address monitoring environmental phenomena by focusing on hotspot sampling without prior knowledge. They present an anytime planning algorithm that efficiently gathers data using non-uniform, data-driven point sampling. The objective of the paper is to balance accurate environmental field mapping with constraints like energy consumption, time, and travel distance. Instead of exhaustive coverage, their method uses a sparse sampling strategy paired with an interpolation technique to approximate the scalar field map. Their adaptive sampling algorithm follows a multi-scale path producing a map of the spatial field with variable resolution. Their method effectively captures the most valuable samples while minimizing travel and energy use, and outperforms traditional exhaustive approaches in both simulations and real-world tests using an ASV.

In some cases, a multi-scale approach is used, where robots first perform a broad survey

before honing in on specific areas of interest for more detailed sampling. Singh et al. [33] explore data-adaptive path planning strategies for mobile sensor networks, with a focus mapping environmental hotspots. The paper highlights the advantages of active learning for mobile sensor path planning, showing that it outperforms traditional sampling methods in terms of accuracy, path efficiency, and speed. They propose a multi-scale approach where an initial uniform sweep is performed to obtain a low resolutions map of the environmental phenomena followed by detailed sampling of the specific areas of interest.

In terms of exploration and mapping, RL has become an increasingly popular tool for optimizing robotic sampling strategies. Research in this area [15, 36, 14, 6] highlights the integration of RL with robotics for developing efficient exploration strategies. For example, Pan et al. [28] demonstrate the use of RL for efficient online adaptive sampling in multi-robot systems, which involves planning efficient sampling trajectories for a team of robots within a fixed endurance budget. The proposed approach, called Multi-Agent Reinforcement Learning for Adaptive Sampling (MARLAS), focuses on quasi-static environmental processes and aims to maximize sampling in high-utility regions. MARLAS learns decentralized policies, enabling robots to coordinate while avoiding overlapping trajectories.

Jamieson et al. [12] extend this work by incorporating more sophisticated exploration policies, allowing robots to balance exploration and exploitation and ensuring that robots can discover new areas of interest while efficiently covering known hotspots. In a similar vein, Nam et al. [26] employ RL to optimize exploration strategies in unknown environments, focusing on minimizing travel distance and improving overall coverage.

RL has been increasingly applied to adaptive sampling, particularly in guiding robots toward regions of higher environmental significance based on a heat map of relevance scores. In the work by Manjanna et al. [23], the authors introduce a novel approach that extends the standard RL framework by addressing the challenges posed by the complexity of state spaces in active sampling tasks. They propose several different feature aggregation methods and they show that the agent that uses the multi-resolution feature aggregation method Figure 2.1 achieves the highest rewards compared to other feature aggregation methods. The multi-resolution feature aggregation method is designed to simplify the policy search process by breaking down the environment's state space into varying levels of granularity. The core idea behind this approach is to aggregate environmental features at multiple scales. This hierarchical aggregation reduces the dimensionality of the state space, which would otherwise be too vast and complex for efficient learning. By organizing the environment into a multi-resolution hierarchy, the model can retain detailed information about the most critical

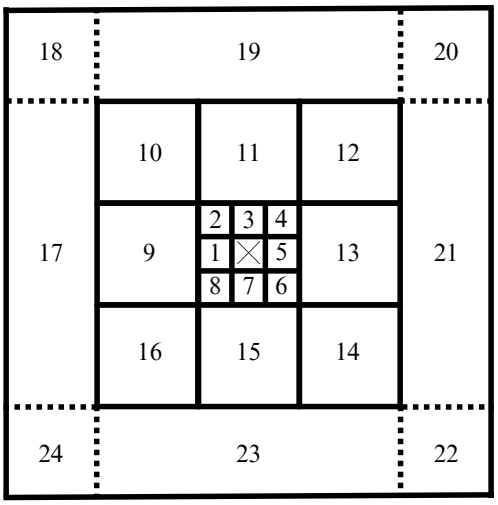


Figure 2.1: A diagram of the multi-resolution feature aggregation technique where (×) marks the current position.

areas while maintaining a broader understanding of the overall environment. This way, it intelligently balances between computational efficiency and the level of detail required for the task. For example, in their method, coarse-grained features are located far from the position of the ASV provide a general overview of the environment, helping the agent understand the larger context. Simultaneously, fine-grained features located nearby the agent's position are used to focus on the best possible immediate actions. This multi-resolution feature aggregation greatly enhances the efficiency of the policy search, making it possible to perform adaptive sampling in complex environments. The ability to switch between different levels of detail ensures that the sampling strategy is both effective in capturing critical environmental data and efficient in terms of computational resources and time. Consequently, this method is particularly well-suited for tasks such as monitoring environmental phenomena, where both accuracy and adaptability are paramount.

3

Background

In our approach, we use GPs to infer the map and RL to plan sampling locations. Both GPs and RL have proven to be invaluable tools in robotics, environmental monitoring, and autonomous systems, offering solutions for efficient modeling, decision-making, and path planning. In this chapter, we explore GPs and RL in more detail.

3.1 Gaussian Processes (GPs)

GPs are powerful non-parametric models that are widely used for regression and probabilistic modeling of complex functions [31]. A GP provides a distribution over functions and is fully defined by a mean function and a covariance (kernel) function. GPs are Bayesian in nature, offering a principled approach to uncertainty quantification by producing not only predictions (mean values) but also uncertainties associated with those predictions. In this section, we introduce the key concepts and equations of GPs, covering the prior and posterior distributions, and the role of kernel functions such as the SE and Matérn 5/2 kernels.

3.1.1 Main Definition

A GP is a collection of random variables, any finite number of which have a joint Gaussian distribution. It can be viewed as a generalization of the Gaussian distribution to infinite

dimensions. Formally, a GP is defined as:

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}')\right) \tag{3.1}$$

where $m(\mathbf{x})$ is the mean function, representing the expected value of the function at $\mathbf{x}$, $\kappa(\mathbf{x}, \mathbf{x}')$ is the kernel (covariance) function, describing the covariance between function values at inputs $\mathbf{x}$ and $\mathbf{x}'$.

The mean function is often assumed to be zero (i.e., $m(\mathbf{x}) = 0$) unless prior knowledge suggests otherwise.

A key concept in GPs is the notion of prior and posterior distributions. Before observing any data, the GP defines a *prior* distribution over possible functions. Once we observe data, we compute the *posterior* distribution over functions, which updates the prior based on the observed data.

Given a set of training points $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^d$ are input locations and $y_i \in \mathbb{R}$ are noisy observations modeled as:

$$y_i = f(\mathbf{x}_i) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2)$$

where σ_n^2 is the noise variance. The joint prior distribution over the training outputs $\mathbf{y}$ and test outputs μ_* at test locations $\mathbf{X}_*$ can be written as:

$$\begin{bmatrix} \mathbf{y} \\ \mu_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \kappa(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I & \kappa(\mathbf{X}, \mathbf{X}_*) \\ \kappa(\mathbf{X}_*, \mathbf{X}) & \kappa(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix}\right)$$

where $\kappa(\mathbf{X}, \mathbf{X})$ is the covariance matrix between the training inputs, $\kappa(\mathbf{X}, \mathbf{X}_*)$ is the covariance matrix between the training and test inputs.

The posterior distribution over test function values μ_* is obtained by conditioning the joint Gaussian prior on the observed data:

$$\mu_* \sim \mathcal{N}(\mu_*, \Sigma_*)$$

with mean and covariance given by:

$$\mu_* = \kappa(\mathbf{X}_*, \mathbf{X})[\kappa(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I]^{-1}\mathbf{y}$$

$$\Sigma_* = \kappa(\mathbf{X}_*, \mathbf{X}_*) - \kappa(\mathbf{X}_*, \mathbf{X})[\kappa(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I]^{-1}\kappa(\mathbf{X}, \mathbf{X}_*)$$

GPs offer a Bayesian framework for function estimation, allowing the incorporation of prior knowledge and updating it in a principled manner as new data is observed. Through GP inference, the posterior provides not only a pointwise mean and uncertainty but also a full description of the posterior distribution over the function and they provide information about the correlation of uncertainty across the domain. If the GP is uncertain about a particular region, it may extend this uncertainty to nearby regions, depending on the structure of the kernel.

One of the main advantages of GPs is that they provide a tractable posterior distribution over functions. Given a prior distribution and observed data, the posterior can be computed analytically, thanks to the properties of Gaussian distributions. This makes GPs computationally attractive for a wide range of problems, as exact inference can be performed without resorting to approximate methods.

Moreover, the ability to compute both the mean and uncertainty of predictions makes GPs ideal for tasks where uncertainty matters, such as active learning, and adaptive sampling [17]. By selecting points with high predictive uncertainty, GPs can efficiently guide exploration in unknown environments, ensuring that new data points are collected where they are most informative.

3.1.2 Kernels (Covariance Functions)

The kernel function $k(\mathbf{x}, \mathbf{x}')$ plays a central role in defining the shape and smoothness of the functions that a GP can model. It encodes assumptions about the function's properties, such as smoothness, periodicity, or stationarity.

Squared Exponential Kernel

The radial basis function (RBF) kernel describes a class of kernels of the form $\kappa(x, y) = f(d(x, y))$ for some decreasing function $f : \mathbb{R} \to \mathbb{R}$, where d is a metric. The SE (also known as Gaussian) kernel is an RBF kernel and is one of the most widely used kernels. It is defined as:

$$k_{\mathrm{SE}}(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\ell^2}\right) \tag{3.2}$$

where $\|x - y\|$ is the Euclidean distance between points x and y, and ℓ is the length scale, controlling the smoothness of the function.

The SE kernel assumes that function values at inputs that are close to each other are

highly correlated, resulting in smooth and continuous functions. This kernel produces very smooth functions, often too smooth for data where discontinuities or non-smooth behavior may occur.

Matérn 5/2 Kernel

The Matérn kernel is a more flexible alternative to the SE kernel. allowing for less smooth functions. The Matérn 5/2 kernel is commonly used for geospatial modeling [24], where the data may exhibit some degree of smoothness but also contain small variations that need to be captured. It is defined as:

$$\kappa(x, y) = (1 + \frac{\sqrt{5}\|x - y\|}{\ell} + \frac{5\|x - y\|^2}{3\ell^2}) \exp(-\sqrt{5}\frac{\|x - y\|}{\ell}) \tag{3.3}$$

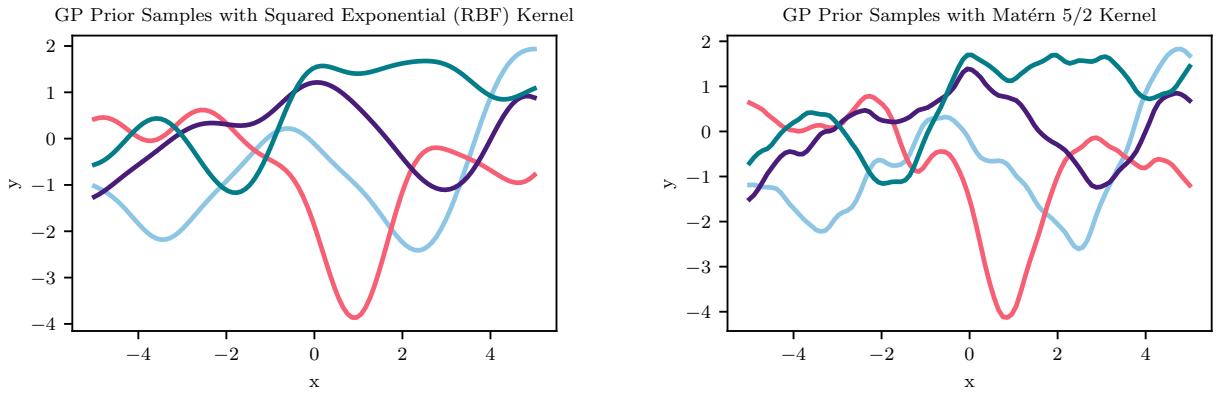
This kernel produces functions that are twice differentiable, which means that they are smoother than those generated by other Matérn kernels, such as Matérn 3/2, but less smooth than those modeled by the SE kernel [34]. The Matérn kernel can model functions with small-scale variability and is often better suited for capturing local fluctuations in data.

An Example of Prior and Posterior Sampling

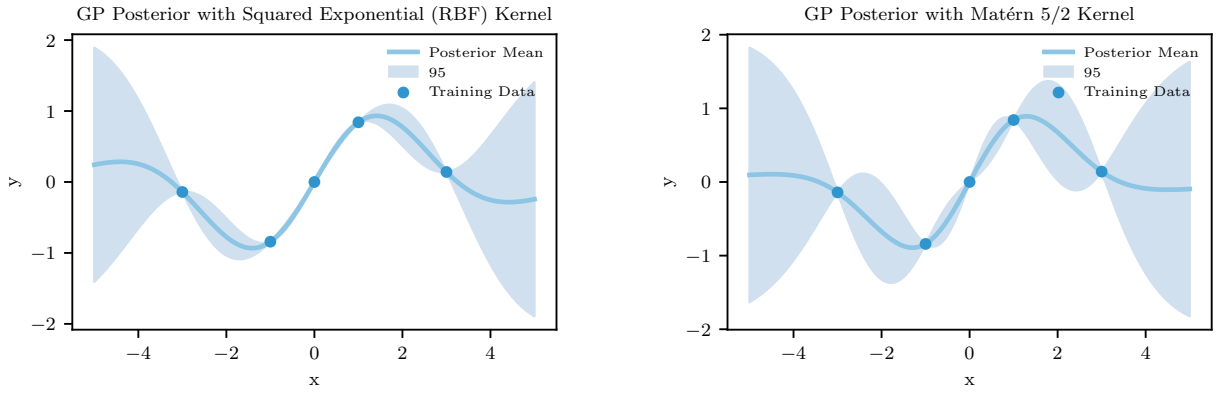
To better understand the effect of different kernel choices on GP predictions, consider a simple 1D example where we model an unknown function using different kernels. Prior to observing any data, samples drawn from the GP reflect the assumptions encoded in the kernel. For instance, an SE kernel will produce smooth, continuous functions Figure 3.1a, while a Matérn 5/2 kernel will generate functions with more variation and less smoothness Figure 3.1b. Once we observe some data, the posterior GP will adjust its predictions, providing a mean function that fits the data and the remaining uncertainty Figures 3.1c and 3.1d. These predictions are shaped heavily by the choice of kernel, demonstrating how kernels control the GP's ability to model different types of functions.

3.2 Reinforcement Learning

In this section, we provide a foundational overview of RL, focusing on key components and methods used to develop agents capable of making sequential decisions in uncertain environments. We begin with the formalization of the RL problem through MDPs, which model environments where the outcomes depend on both the current state and the agent's actions. We also discuss extensions to partially observable environments, known as POMDPs, which introduce additional uncertainty.



(a) (b)



(c) (d)

Figure 3.1: A 1-D example of prior and posterior sampling using two different kernels; SE (left column) and Matérn (right columns) kernels.

Additionally, we cover value functions and action-value functions, which quantify the expected cumulative rewards an agent can achieve. These functions are essential for evaluating and improving the agent's policy, guiding it towards maximizing long-term rewards. We delve into Q-learning, a widely-used model-free algorithm that learns an optimal action-value function for decision-making without requiring a model of the environment, as well as its deep learning extension, DQN, which apply neural networks to approximate the Q-function in complex, high-dimensional state spaces. Through these topics, we establish the theoretical and practical basis for RL methods used in adaptive decision-making tasks.

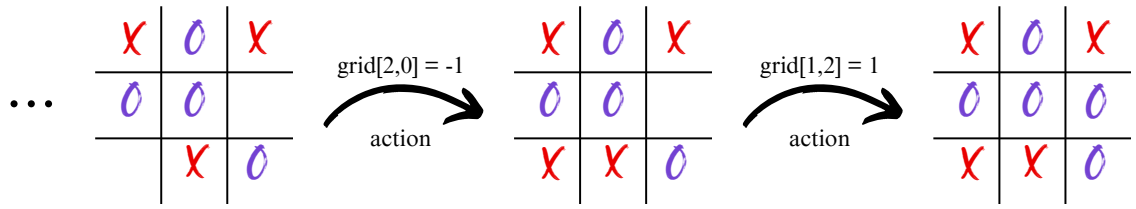
3.2.1 Formulation and Main Definitions

In RL, an agent learns to make decisions through interactions with an environment. Unlike supervised learning, where the model is trained on pre-labeled data, RL involves the agent observing the current state of the environment, deciding what action to take based on the observed state, taking said action and observing the resulting state and reward it recieves. The agent repeats that process with the goal of maximizing the cumulative reward it receives by learning an optimal policy that dictates which actions to take in given states to achieve long-term success. This setting can be described by a *Markov decision process (MDP)* [30] which is defined by a 5-tuple $(\mathcal{S}, \mathcal{A}, r, P, \gamma)$ where

- $\mathcal{S}$ is the set of possible states called the *state space*
- $\mathcal{A}$ is the set of possible actions called the *action space*
- $r : \mathcal{S} \to \mathbb{R}$ is the *reward function*
- P is the *transition probability* where $P(s' \mid s, a)$ is the probability of observaing state s' after observing state s and taking action a
- $\gamma \in [0, 1]$ is the *discount factor* used to discount the value of future rewards.

Here we present a simple example of an MDP. The following components describe and MDP for the game of "Tic-Tac-Toe" (see Section 3.2.1).

- The state space $\mathcal{S}$ is the set of all possible tuples (3×3 matrices, player turn) where the 3×3 matrices represent possible 3×3 grid configurations and the player turn can be an X, an O, or nothing if the state is terminal.
- The action space $\mathcal{A}$ is the set of all possible moves, meaning placing an X or an O(depending on the player turn) in any of the empty grid cells.



$$s_7 = \left(\begin{bmatrix} -1 & 1 & -1 \\ 1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}, -1 \right) \quad s_8 = \left(\begin{bmatrix} -1 & 1 & -1 \\ 1 & 1 & 0 \\ -1 & -1 & 1 \end{bmatrix}, 1 \right) \quad s_9 = \left(\begin{bmatrix} -1 & 1 & -1 \\ 1 & 1 & 1 \\ -1 & -1 & 1 \end{bmatrix}, 0 \right)$$

$$r(s_7) = 0 \qquad r(s_8) = 0 \qquad r(s_9) = 1$$

Figure 3.2: An example of an MDP and its components for the game of "Tic-Tac-Toe".

- The reward here is sparse and only recieved at the end of the game. The game is finite and ends after ≤ 9 moves.

$$r = \begin{cases} 1 & \text{If agent wins} \\ 0 & \text{In case of a draw} \\ -1 & \text{If agent loses} \end{cases}$$

- The transition function is deterministic. The action taken modifies the state by adding an X or an O to the 3×3 grid in the cell the player chose and switching the player turn.

- The discount factor $\gamma = 1$ since the game has a finite horizon of 9 moves and the agent only gets a reward at the end of the game.

For more complex problems, the problem can have an infinite horizon or the reward might not be sprase. Therefore, we need a general object that encompasses the reward at each step. Let G denotes the discounted cummulative reward defined as

$$G := \sum_i \gamma^i r_i = r(s_0) + \gamma r(s_1) + \gamma^2 r(s_2) + \ldots [1], \tag{3.4}$$

where $r(s_i)$ is the reward the agent receives at step i. The goal of the RL agent is to learn

[1] If $\gamma = 1$, then G turns into the cumulative reward $\sum_i r(s_i)$.

a policy $\pi : \mathcal{S} \to \mathscr{P}(\mathcal{A})$ that maximizes the expected return $\mathbb{E}[G]$. Now, we can define the *value function* V^π for a policy π, which represents how desirable a state is, by

$$V^\pi(s) = \mathbb{E}_\pi[G \mid s_0 = s] \tag{3.5}$$

. We can see that $V^\pi(s)$ satisifies the Bellman equation

$$\begin{aligned}
V^\pi(s) &= \mathbb{E}_\pi\left[\sum_i \gamma^i r(s_i) \mid s_0 = s\right] \\
&= \mathbb{E}_\pi\left[r(s) + \gamma \sum_{i>0} \gamma^{i-1} r_i \mid s_0 = s\right] \\
&= r(s) + \gamma \sum_a \pi(a \mid s) \sum_{s'} P(s' \mid s, a) \mathbb{E}_\pi\left[\sum_i \gamma^i r(s_i) \mid s_0 = s'\right] \\
&= r(s) + \gamma \sum_a \pi(a \mid s) \sum_{s'} P(s' \mid s, a) V^\pi(s').
\end{aligned}$$

Similarily, we can define the *action-value function*, also known as the *Q-function,* Q^π for a policy π by

$$Q^\pi(s, a) = \mathbb{E}_\pi[G \mid s_0 = s, a_0 = a]. \tag{3.6}$$

We can also see that the Q-function also satisfies the Bellman equation since

$$\begin{aligned}
Q^\pi(s, a) &= \mathbb{E}_\pi\left[\sum_i \gamma^i r(s_i) \mid s_0 = s, a_0 = a\right] \\
&= \mathbb{E}_\pi\left[r(s) + \gamma \sum_{i>0} \gamma^{i-1} r_i \mid s_0 = s, a_0 = a\right] \\
&= r(s) + \gamma \sum_{s'} P(s' \mid s, a) \sum_{a'} \pi(a', s') \mathbb{E}_\pi\left[\sum_i \gamma^i r(s_i) \mid s_0 = s', a_0 = a'\right] \\
&= r(s) + \gamma \sum_{s'} P(s' \mid s, a) \sum_{a'} \pi(a' \mid s') Q^\pi(s', a').
\end{aligned}$$

The Bellman Optimality Equation applies both to the value function $V^\pi(s)$ and the Q-function $Q^\pi(s, a)$. Each serves a different purpose in RL, but both express the recursive nature of optimal decision-making over time. Here is how the Bellman Optimality Equation is used for both:

1. Bellman Optimality Equation for the Value Function $V^*(s)$

   The optimal value function $V^*(s)$ represents the expected future cumulative reward

when starting from state s and following the optimal policy. It describes how desriable it is to be in a state, assuming the agent behaves optimally thereafter.

The Bellman Optimality Equation for the value function is given by:

$$V^*(s) = r(s) + \gamma \max_a \left[\sum_{s'} P(s' \mid s, a) V^*(s') \right] \tag{3.7}$$

where

- $V^*(s)$ is the optimal value of state s, which maximizes the long-term reward.
- $V^*(s')$ is the optimal value of the next state s', assuming optimal action choices going forward.

This equation expresses that the value of state s is the reward received at this state, and then choosing the best possible action which leads to a future state s', plus the discounted value of being in that state.

2. Bellman Optimality Equation for the Q-Function $Q^*(s, a)$

   The optimal Q-function $Q^*(s, a)$ extends the concept of $V^*(s)$ to also consider actions. While $V^*(s)$ gives the optimal value of a state, the optimal Q-function $Q^*(s, a)$ gives the optimal value of taking action a in state s and then following the optimal policy. It is crucial in many RL algorithms (like Q-learning §3.2.3) that learn state-action values directly.

   The Bellman Optimality Equation for the Q-function is:

$$Q^*(s, a) = r(s) + \gamma \sum_{s'} P(s' \mid s, a) \max_{a'} Q^*(s', a')$$

   where

   - $Q^*(s, a)$ is the optimal value of taking action a in state s.
   - $\max_{a'} Q^*(s', a')$ represents the maximum value obtainable from the next state s' by choosing the optimal action a'.

This version of the Bellman equation emphasizes that the value of a state-action pair depends on the future rewards obtained by transitioning to the next state s' and choosing the best possible action a' from that state.

RL is fundamentally concerned with how agents take actions in an environment to maximize cumulative rewards over time. Within this framework, *on-policy* and *off-policy* methods describe different approaches to learning from the agent's experiences. On-policy RL refers to algorithms where the policy being evaluated and improved is the same as the policy being followed to generate behavior. This means that the agent learns from actions it takes while exploring the environment based on its current policy. On-policy methods can be advantageous because they allow for stable learning, as the agent directly optimizes the policy it uses for decision-making.

In contrast, off-policy RL involves learning about one policy (the target policy) while following another policy (the behavior policy). This flexibility allows agents to learn from experiences generated by different policies, which can be particularly useful for exploration. A classic example of off-policy learning is Q-learning §3.2.3, where the agent learns the optimal action-value function irrespective of the policy it uses to explore. It updates its estimates based on the maximum action value for the next state, allowing it to converge towards the optimal policy even if the behavior policy is exploratory (such as ϵ-greedy). Off-policy methods can be more versatile, enabling the agent to improve its learning from past experiences, including those generated by other agents or even historical data.

Another critical distinction is between *model-free* and *model-based* RL. Model-free RL does not rely on a model of the environment; instead, it learns optimal policies directly from interactions with the environment. The advantage of model-free approaches is their simplicity and ease of implementation, particularly in environments where constructing a model is impractical due to complexity. However, they often require more samples to converge to optimal policies, as they learn purely from trial and error.

In contrast, model-based RL utilizes a model of the environment to predict transitions and rewards, enabling the agent to plan ahead. By simulating different action sequences using the model, the agent can evaluate potential future states and optimize its behavior accordingly. This approach can be significantly more sample-efficient since it allows for planning and learning from imagined experiences rather than relying solely on actual interactions. However, the effectiveness of model-based methods depends heavily on the accuracy of the learned model. If the model is flawed, the agent's planning can lead to suboptimal actions.

3.2.2 Partially Observable Markov Decision Process

In many real-world applications, the agent does not have access to the full state of the environment but instead must make decisions based on incomplete or noisy observations. The framwork of an MDP under imperfect information was introduced by Åström [42]. Such problems are typically described using *partially observable Markov decision processes (POMDPs)* [13] which is characterized by a tuple $\langle \mathcal{S}, \mathcal{A}, \Omega, P, \mathcal{O}, r, \gamma \rangle$, where

- $(\mathcal{S}^2, \mathcal{A}, r, P, \gamma)$ describes an MDP.
- Ω is a finite set of *observations* that the agent can receive after taking an action. Each observation provides some information about the underlying state but does not guarantee complete knowledge of it.
- $\mathcal{O} : S \times A \times \Omega \to [0, 1]$ is the *observation function* that defines the probability of receiving an observation given the current state and the action taken. Formally, $\mathcal{O}(o \mid s, a)$ is the probability of observing $o \in \Omega$ after taking action a and being in state s.

In a POMDP, the agent receives observations o_t that provide partial information about the true state s_t and must learn a policy that maximizes expected cumulative rewards based on these incomplete observations [13].

POMDPs introduce a new layer of complexity to RL since they are known to be PSPACE-complete [29] indicating that if POMDPs could be solved in polynomial time, it would imply that NP-complete problems could also be solved in polynomial time. POMDPs' inherent complexity arises from several key factors. First, POMDPs involve uncertainty in both state and observation, requiring agents to make decisions based on incomplete information. This uncertainty is represented by a belief state, which is a probability distribution over all possible states, making it necessary for the agent to update this belief as it interacts with the environment. Consequently, the belief space can grow exponentially with the number of states and observations, leading to an enormous number of potential belief states that must be considered at each decision-making step. Furthermore, computing the optimal policy in a POMDP is computationally challenging, as it requires evaluating strategies over potentially infinite time horizons while accounting for all possible sequences of observations and state transitions. This complexity is compounded by the need to consider mixed strategies, which may involve randomizing actions, further complicating the analysis of expected outcomes.

[2] The states may not be fully observable due to the partial observability of the environment.

The following is an example of a simple POMDP illustrated in Figure 3.3. Imagine a robot navigating a foggy room to find a charging station. The room has four possible locations for the robot: $\{A, B, C, D\}$, and the robot can move left or right. The charging station is in location D. However, because the room is foggy, the robot cannot see exactly which location it is in. Instead, it receives a partial, noisy observation about whether it is near a wall or in the middle of the room, and this observation may not always be accurate. It receives a correct observation with probablity $= 0.9$

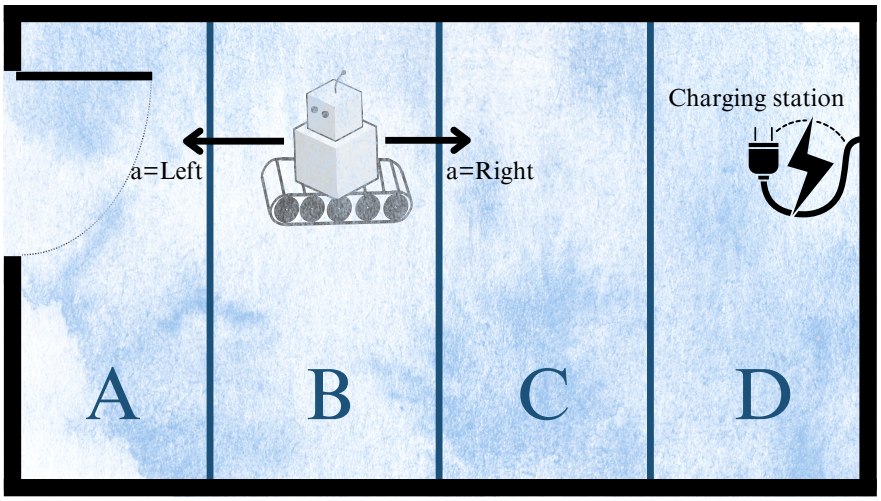


Figure 3.3: An illustration of a POMDP example. A robot navigating a foggy room to find a charging station in area D.

We define the POMDP in this example and list its components.

- $\mathcal{S} = \{A, B, C, D\}$.
- $\mathcal{A} = \{\text{Left, Right}\}$.
- $\Omega = \{0, 1\}$; 1 for "near wall" and 0 for "not near wall".
- $r(s) = \begin{cases} 1 & \text{if } s = D \\ 0 & \text{otherwise} \end{cases}$
- $P(s' \mid s, a)$ Moving left or right transitions the robot to a neighboring state.
  - $P(s' = A \mid s = B, a = \text{Left}) = 1$
  - $P(s' = B \mid s = C, a = \text{Left}) = 1$
  - $P(s' = B \mid s = A, a = \text{Right}) = 1$
  - $P(s' = C \mid s = B, a = \text{Right}) = 1$

- $P(s' = C \mid s = D, a = \text{Left}) = 1$
- $P(s' = D \mid s = C, a = \text{Right}) = 1$

- $\mathcal{O}(o \mid s)$ is the probability of receiving an observation o given the actual state s.
  - $\mathcal{O}(0 \mid s \in \{B, C\}) = 0.9$
  - $\mathcal{O}(0 \mid s \in \{A, D\}) = 0.1$
  - $\mathcal{O}(1 \mid s \in \{B, C\}) = 0.1$
  - $\mathcal{O}(1 \mid s \in \{A, D\}) = 0.9$

- $r(s) = \begin{cases} 1 & \text{if } s = D \\ 0 & \text{otherwise} \end{cases}$

Since the robot does not know exactly which state it is in because of the fog, it needs to maintain a belief state $b(s)$, which represents the probability of being in each state. After the robot takes an action and receives an observation, it updates its belief based on the observation and the action taken. This update is done using *Bayes' Rule*

$$b(s') = \frac{\mathcal{O}(o \mid s') \sum_s P(s' \mid s, a) b(s)}{\sum_{s'} \mathcal{O}(o \mid s') \sum_s P(s' \mid s, a) b(s)}, \tag{3.8}$$

which adjusts the belief based on the likelihood of the observed information given the action. The robot selects actions based on its belief. It chooses actions that maximize the expected reward based on its current belief state. Since the robot does not know exactly where it is, it will choose actions that gradually reduce uncertainty and move it toward the charging station.

In this example, suppose the robot starts with an intial belief that it is equally likely to be in any of the four states $b(s_0 = A) = b(s_0 = B) = b(s_0 = C) = b(s_0 = D) = 0.25$. After moving right $a = \text{Right}$, the robot observes "not near wall" $\mathcal{O}(o \mid s_1) = 0$. Based on this observation, the robot updates its belief according to Equation (3.8). Now, its belief is that $b(s_1 = A) = 0,\ b(s_1 = B) = b(s_1 = C) = 0.45,$ and $b(s_1 = D) = 0.05$. The process repeats, and the robot gradually refines its belief and takes actions that help it find the charging station.

3.2.3 Q-Learning

One of the primary goals in RL is to learn the optimal value function and Q-function. A key method for learning the Q-function is *Q-Learning*, Algorithm 1, a model-free off-policy algorithm designed to learn the value of actions in a given state without requiring a

model of the environment. In Q-learning, the agent learns an action-value function $Q(s, a)$, which estimates the expected future cummulative rewards of taking action a in state s and then following the optimal policy thereafter. The Q-learning update rule incorporates the maximum estimated value of the next state-action pair, enabling the agent to improve its policy based on the best possible future rewards. Mathematically, the update rule for the Q-function $Q(s, a)$ is given by

$$Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s, a)),$$

where α is the learning rate. The Q-learning algorithm iteratively updates the Q-values based on the agent's experiences, allowing it to learn an optimal policy over time. This method is particularly effective in environments with a discrete set of states and actions, making it widely applicable in various domains, including robotics and game playing.

Algorithm 1 Q-Learning

```
Pick learning rate α ∈ (0, 1]
Initialize Q(s, a), ∀(s, a) ∈ (S, A)
Initialize Q(terminal, ·) = 0
for each episode do
    Initialize state s ∈ S from environment
    for each step in episode until s is terminal do
        Pick action a ∈ A using a policy derived from Q(s, ·)
        s', r ∼ P(·, · | s, a)                    ▷ Sample next state from the environment
        Q(s, a) ← Q(s, a) + α(r + γ max_a' Q(s', a') − Q(s, a))
        s ← s'
    end for
end for
```

Q-Learning has proven to be a versatile algorithm, with applications ranging from simple grid-world environments to more complex domains. However, it faces significant scalability issues, particularly as the complexity of the state and action spaces increases. One key limitation is that the algorithm learns the value of all state-action pairs, rather than just finding the optimal policy, which leads to an increased demand for storage and computational resources. As the number of states and actions in an environment increases, the size of the Q-table, which stores the Q-values for each state-action pair, grows exponentially. This phenomenon is often referred to as the curse of dimensionality. For example, in a grid-world scenario, even a small increase in the dimensions (number of states) can lead to an enormous Q-table. If an environment has 10 states and 10 actions, the Q-table would contain $10 \times 10 = 100$ entries. However, if the state space increases to 100 states and 10 actions,

the Q-table would grow to $100 \times 10 = 1000$ entries. This exponential growth makes it impractical to store and update Q-values for larger environments, especially when dealing with high-dimensional spaces such as images or continuous states.

Building on the scalability issues of Q-learning in discrete state and action spaces, the challenges become even more pronounced when applied to environments with continuous state and action spaces. As the number of state and action variables grows, the size of the Q-table expands exponentially, making it difficult to scale efficiently. For example, when state and action variables are discretized, the required table size can become impractically large. In the case of eight state variables and two action variables, discretized into seven bins, the Q-table would need to store nearly 300 million Q-values. This makes it infeasible to gather the enormous amount of training data required to fill such a large table, and without generalization, the algorithm cannot efficiently transfer learning between similar states or actions, severely limiting its applicability in large-scale problems [7]. Additionally, using coarser discretization can lead to aliasing, where functionally different states are mapped to the same representation, causing inaccuracies in learning.

These limitations make Q-learning impractical for real-world applications with high-dimensional data, such as robotic control or video game environments. To address scalability issues, researchers often turn to function approximation methods, such as neural networks, to represent the Q-values instead of maintaining a large Q-table. However, using function approximation introduces new challenges, including stability and convergence issues, as neural networks can be sensitive to hyperparameters and require careful tuning. Moreover, the quality of the approximation can significantly affect the agent's learning performance, making it crucial to select appropriate architectures and training strategies.

3.2.4 Deep Q-Networks

Deep Q-Networks (DQN) were introduced by Mnih et al. [25], combining Q-learning with deep learning. Instead of using a Q-table to store action values, DQN uses a deep neural network to approximate the Q-function, allowing the agent to generalize across large, continuous state-action spaces. This innovation enabled RL to scale to complex environments where tabular methods would be infeasible.

In DQN, the neural network takes a state as input and outputs the Q-values for all possible actions in that state. The DQN algorithm (Algorithm 2) introduced several key innovations to stabilize training:

1. **Experience Replay:** Instead of learning from consecutive transitions, DQN stores

transitions (s, a, r, s') in a replay buffer. Mini-batches are sampled randomly from this buffer during training, breaking the correlation between sequential updates and improving data efficiency.

2. **Target Networks:** In traditional Q-learning, the target Q-value is computed using the same Q-function that is being updated, which can lead to instability. DQN mitigates this by using a target network, a separate neural network whose parameters are periodically copied from the main Q-network, to compute the target values.

Algorithm 2 DQN

Pick learning rate $\alpha \in (0, 1]$
Pick $\epsilon > 0$ such that $\epsilon << 1$
Pick target update frequency $\mathcal{C}$
Initialize a replay buffer $\mathcal{B}$ with capacity N
Initialize a neural network Q_θ with weights θ
Initialize a target neural network Q_ϕ with weights $\phi \leftarrow \theta$
for each episode **do**
  Initialize state $s \in \mathcal{S}$ from environment
  for each step in episode until s is terminal **do**
    Pick action $a \in \mathcal{A}$ using an ϵ-greedy policy based on $Q_\theta(s, \cdot)$
    $s', r \sim P(\cdot, \cdot \mid s, a)$ ▷ Sample next state from the environment
    Add (s, a, r, s') to $\mathcal{B}$
    $\theta \leftarrow \theta - \alpha \nabla_\theta (r + \gamma \arg\max_{a \in \mathcal{A}} Q_\phi(s', a) - Q_\theta(s, a))^2$
    if step is divisible by $\mathcal{C}$ **then**
      $\phi \leftarrow \theta$
    end if
    $s \leftarrow s'$
  end for
end for

However, DQN can suffer from an issue known as overestimation bias due to the nature of the max operator in the Q-learning target update. Specifically, in DQN, the algorithm estimates the maximum future reward for the next state by selecting and evaluating the same Q-values. This leads to an over-optimistic estimate because any errors or noise in the Q-value estimates can cause the algorithm to overestimate the true value of the optimal action.

4

Problem Formulation and Solution Outline

In this chapter, we formally define the adaptive sampling and mapping problem of aquatic environments and present a structured approach to solving it using RL to address the path planning problem and posterior inference to reconstruct the map. Then, we discuss using domain randomization during training to achieve a robust exploration policy. The problem involves efficiently exploring, sampling, and modeling an unknown aquatic environment while minimizing the distance travelled and the number of samples collected. By formalizing the problem as a decision-making process under uncertainty, we outline the necessary mathematical models, state representations, and optimization objectives. Through this approach, we aim to provide a robust and scalable solution to the problem of adaptive environmental monitoring.

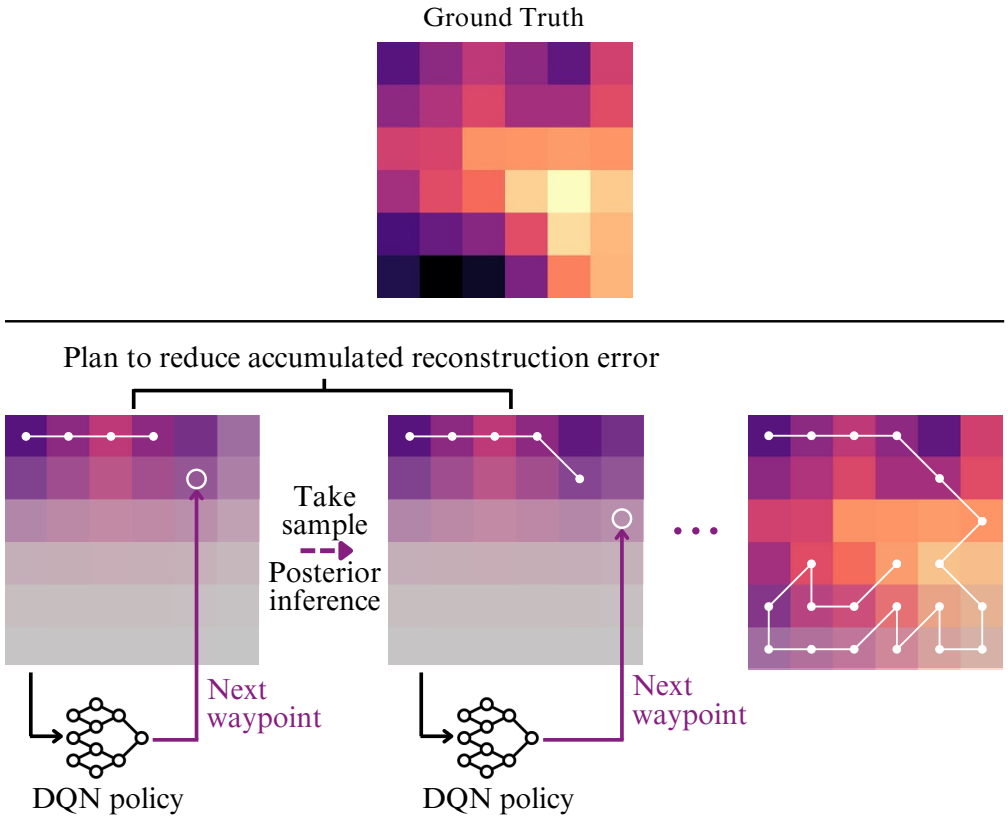


Figure 4.1: Depiction of our active sampling method. By incorporating posterior uncertainty (depicted by translucency) over the map, the agent can achieve accurate reconstruction without exhaustively covering its domain.

4.1 Problem Formulation

We consider an ASV navigating in an open body of water $\mathcal{W} \subset \mathbb{R}^2$. A *map* over $\mathcal{W}$ is an object $M \in \mathbb{R}^{\mathcal{W}}$, prescribing a scalar property (e.g., temperature, depth) to every location in the water. Our boat is assumed to be equipped with a sensor that measures $M(x)$ at positions x. In practice, however, sensors are subject to noise and cannot capture the precise value of $M(x)$. For instance, our temperature sensor has a measurement uncertainty of $\pm 0.1°\text{C}$. In this thesis, we simplify the problem by assuming that the sensor noise is minimal and negligible, allowing us to treat the sensor readings as accurate.

The overarching goal of this work is to efficiently reconstruct the environmental map M while minimizing the number of required sensor queries. To make this problem computationally tractable, we discretize the continuous water domain $\mathcal{W}$ into a finite set of N discrete

cells, which may be structured, for example, as a uniform grid. Thus, $\mathcal{W} = \{x_i\}_{i=1}^N \subset \mathbb{R}^2$, and the map M can be represented as a finite-dimensional vector in $\mathbb{R}^N$.

To evaluate the accuracy of our reconstructed map $\widehat{M}$, we use the root mean squared error (RMSE) metric, which is a standard measure of reconstruction quality. The RMSE is computed as follows:

$$c_{\mathsf{RMSE}}(\widehat{M}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\widehat{M}(x_i) - M(x_i))^2}. \tag{4.1}$$

This error metric quantifies the difference between the true map M and the estimated map $\widehat{M}$, averaged across all discretized locations. An accurate reconstruction corresponds to a low RMSE, indicating that our approach effectively captures the underlying environmental field using a limited number of sensor measurements.

In the remainder of this chapter, we describe our approach (illustrated in Figure 4.1) to approximating M with few samples by leveraging posterior inference of the map to efficiently decide which regions to sample next in response to the agent's uncertainty of the map.

4.2 The Planning Problem

Our objective is to synthesize a decision-making policy that directs the boat's movements in response to its current position in the water, as well as its degree of uncertainty about the map M. This setup is formalized as a planning problem aimed at maximizing map accuracy while minimizing the trvel time and the number of sensor measurements.

Since the agent (the boat) lacks precise information about the full extent of the underlying map M, the decision-making problem becomes one of choosing an optimal sequence of sample collection points to effectively reconstruct the map with minimal effort. Formally, this type of decision problem, balancing exploration with uncertainty and gathering sequential information to achieve a mapping goal, falls under a POMDP framework. POMDPs have been successfully applied to robotic exploration tasks in uncertain environments [18, 8, 27]. Planning in a POMDP refers to finding an optimal sequence of actions for an agent to maximize its expected rewards over time, despite having incomplete knowledge about the environment. Unlike in a fully observable environment, where the agent knows the exact current state, a POMDP setting means the agent only has partial observations that provide indirect information about the state. To make this problem computationally tractable, we

simplify it with a key approximation: we assume that sensor noise is low enough that each measurement $M(x)$ at a given position x can be treated as an exact reading. This reduction allows us to reformulate the problem as an MDP, where the agent has full observability of the environment's current state at each step. This approach of treating sensor measurements as exact readings to simplify the problem has been employed in previous studies, allowing researchers to bypass the complexity of partial observability by reformulating the problem as an MDP. This assumption enables the agent to operate with full state observability, which significantly reduces computational demands and has proven effective in similar robotic exploration tasks [37].

Under this MDP formulation, we can employ RL methods to solve the planning problem efficiently. By treating each sensor reading as accurate, the agent can accumulate data to iteratively refine its understanding of M and guide subsequent decisions based on the state of knowledge at each step. We will demonstrate in this section and §4.4 how this formulation simplifies the planning task, enabling a structured solution through RL techniques that take advantage of both state observability and goal-oriented exploration.

For navigation, we assume the boat is equipped with a robust low-level controller that manages its path from one point to another within the local vicinity. At any given location $x \in \mathcal{W}$, the boat has a set of neighboring cells, denoted by $\mathsf{N}(x)$, representing the possible next steps. Therefore, $\mathsf{N}(x)$ comprises all feasible actions from position x. This navigational setup enables the agent to systematically explore the environment in a controlled manner, assessing potential sample points along the way.

The agent's state at any time t includes both its physical location, $x_t \in \mathcal{W}$, and an updated representation of its knowledge about M, which we denote as a sufficient statistic $U_t \in (\mathbb{R} \cup \{\mathsf{unk}\})^{\mathcal{W}}$ for its posterior belief over the map based on samples collected so far. This sufficient statistic U_t holds crucial information about which parts of the map are known or unknown. Specifically, $U_t(x)$ stores the value $M(x)$ if the agent has previously sampled at position x; otherwise, it records unk, indicating that no data has yet been gathered for location x. This not only guides the agent's exploration strategy but also informs its decision-making process as it balances the benefits of exploring new regions versus re-sampling previously visited locations.

Given an initial prior distribution $p \in \mathscr{P}(\mathbb{R}^{\mathcal{W}})$ that represents our preliminary beliefs about M, the planning problem is defined by the objective of minimizing a cumulative cost function Equation (4.2). By effectively navigating the trade-off between exploration and

exploitation, the agent aims to construct a comprehensive and accurate map with minimal resource use, ultimately supporting efficient environmental mapping.

$$
\begin{aligned}
c(x_t, U_t) &= c_{\mathsf{RMSE}}\left(\mathbb{E}_{\widehat{M}\sim\rho_t}\{\widehat{M}\}\right) \\
\text{where } \rho_t &= p\left(\cdot \,\middle|\, \bigcup_{x:U_t(x)\neq\mathsf{unk}} (x, U_t(x))\right).
\end{aligned}
\tag{4.2}
$$

4.3 Posterior Inference

The cost function defined in (4.2) depends critically on the posterior distribution ρ_t over the environmental map M given the measurements collected by the ASV up to time t. To effectively model the spatial field of interest, we assume a prior distribution p for M in the form of a *Gaussian Process*. This GP framework provides a flexible, probabilistic approach to modeling spatially correlated data, as it allows for continuous spatial predictions and uncertainty quantification in areas without direct measurements.

We refer to the GP as $\mathcal{N}(\mathbf{0}, K)$ where each entry of the covariance matrix K represents the similarity or correlation between two points x_i and x_j in $\mathcal{W}$, specified by the kernel function

$$K_{i,j} = \kappa(x_i, x_j) \text{ over all } x_i, x_j \in \mathcal{W}$$

. The Matérn-5/2 kernel is used here as κ, which is a popular choice for modeling smooth spatial fields with moderate differentiability. The kernel's bandwidth parameter, $\ell > 0$, determines the spatial scale over which points are correlated; larger values of ℓ imply that the measurements at two points influence each other over a greater spatial range.

After gathering observations from previously visited locations

$$\tilde{X} = \{x \in \mathcal{W} : U_t(x) \neq \mathsf{unk}\},$$

we update the prior p to obtain the posterior ρ_t, which incorporates the observed data. The posterior mean μ_t at each unobserved location serves as an estimate for $M(x)$, while the posterior covariance Σ_t quantifies the uncertainty associated with these estimates. The GP posterior has a closed-form solution, computed by first identifying known locations, the set $\tilde{X}$ includes all points where measurements have been collected, allowing us to condition on these data points. Then, calculating the posterior mean μ_t which incorporates the data by adjusting each cell's mean prediction based on observed values and the spatial correlation

defined by K. The posterior covariance Σ_t is updated to capture the uncertainty in unsampled regions, shrinking in areas where observations are densely sampled and increasing in regions far from sampled locations.

This Gaussian posterior model, $\rho_t = \mathcal{N}(\mu_t, \Sigma_t)$ where

$$\begin{aligned} \mu_t &= AB^{-1}\vec{u}_t \quad \Sigma_t = K - AB^{-1}A^\top \\ A_{i,j} &= \kappa(x_i, \tilde{X}_j) \quad B_{i,j} = \kappa(\tilde{X}_i, \tilde{X}_j). \end{aligned} \tag{4.3}$$

enables adaptive path planning by allowing the ASV to prioritize areas with high uncertainty, thereby maximizing information gain in unobserved regions. The posterior update process, informed by the Matérn kernel, allows the planning algorithm to adaptively refine the spatial map M with each new measurement.

4.4 The Reinforcement Learning Solution

To address the problem of minimizing cumulative sampling cost efficiently, we employ RL to train a path planning and autonomous sampling policy. The primary goal is to develop a strategy that guides the agent to select paths in a way that optimizes sample collection while reducing travel and redundant measurements. Given the Markovian nature of the cost function c within the state space $\mathcal{W} \times (\mathbb{R} \cup \{\mathsf{unk}\})^{\mathcal{W}}$ and the assumption that the sensor measurements are accurate, the problem fits naturally into an MDP framework. Here, state transitions occur according to a Markovian kernel, and both position updates and data collection depend solely on the agent's current state, action, and the posterior distribution.

The transition mechanism works as follows: from any location $x_t \in \mathcal{W}$, the agent moves to a new position $x_{t+1} \in \mathsf{N}(x_t)$, which is directly determined by its chosen action. The value of U_{t+1} depends on both the agent's visitation history $\{x_i\}_{i=0}^t$. Specifically, for any location $x \in \mathcal{W}$, the map entry $U_{t+1}(x)$ will remain unchanged if $a_t \in \mathcal{A}$ if x has already been sampled, meaning

$$U_t(x) \neq \mathsf{unk}.$$

If the agent reaches a new, unvisited cell x at time $t+1$, the distribution U_{t+1} is determined by the posterior ρ_t for which U_t is a sufficient statistic.

This Markovian structure enables the application of RL techniques for planning, specifically DQN, to train the agent efficiently. DQN is particularly suited to this setup, as they use experience replay and a target network to stabilize training in high-dimensional environments. In our model, both the reward and transition functions depend on U_t exclusively

through the posterior distribution ρ_t. We augment the state representation with a *visitation map* $V_t \in \{0,1\}^{\mathcal{W}}$ to indicate locations already sampled. Formally, the visitation map at each timestep t is defined as:

$$V_t(x) = \mathbb{1}[U_t(x) \neq \mathsf{unk}],$$

where $\mathbb{1}$ is an indicator function that returns 1 if $U_t(x)$ has been sampled, and 0 otherwise.

The training process focuses on learning an action-value function Q_θ, parameterized by θ, to approximate the Bellman optimality equation. At each timestep, the observed state $o_t = (x_t, \mu_t, \Sigma_t, V_t)$ is used as input for the action-value function, defined by:

$$Q_\theta(o_t, a_t) = \begin{cases} c(\mu_t) + \gamma \mathbb{E}\left\{\min_{a'} Q_{\overline{\theta}}(O_{t+1}, a')\right\} & a \in \mathsf{N}(x_t) \\ \infty & \text{otherwise,} \end{cases}$$

In this equation, γ is the discount factor which controls how much future rewards influence current decisions and the expectation is taken over $O_{t+1} := (x_{t+1}, \mu_{t+1}, \Sigma_{t+1}, V_{t+1})$ which is a Markovian transition model as discussed above. $\overline{\theta}$ denotes a delayed set of (target) parameters, as is common in DQN for ameliorating training stability.

Finally, the trained policy π_θ is derived by selecting actions that minimize the action-value function for each state:

$$\pi_\theta(x_t, \mu_t, \Sigma_t, V_t) = \operatorname*{arg\,min}_{a \in \mathsf{N}(x_t)} Q_\theta(o_t, a).$$

When the arg min is not unique, we pick the first action under some ordering; e.g. that with the lowest index in a list. This RL-based policy provides an efficient sampling strategy, balancing the trade-off between exploring new areas and revisiting uncertain locations.

4.5 Domain Randomization

To enable the autonomous agent to generalize its sampling strategy across diverse aquatic environments, we use *domain randomization* to synthesize a robust exploration policy. In robotics, our goal is to develop robust agents capable of adapting to diverse environments across different locations. Specifically, in aquatic environmental monitoring, we aim for

an agent that can autonomously sample and reconstruct a map of a water quality metric in any aquatic environment from any starting point, even in previously unseen settings, while minimizing travel distance. For instance, if tasked with mapping surface temperature in a lake, we should be able to deploy an ASV equipped with a sensor unit, initiate the mission, and expect it to efficiently navigate the environment, sampling strategically to produce an accurate surface temperature map with the shortest possible path. Training with domain randomization ensures that the agent learns to adapt its path dynamically for each new environment, making it resilient and capable of effective sampling across diverse and unknown aquatic environments.

Domain randomization works by presenting the agent with a large set of simulated maps during training. Each map represents a different instantiation of the underlying field M, effectively allowing the agent to learn patterns and strategies that are not restricted to any single map structure or set of conditions. The randomization process includes procedural generation of maps, which vary in spatial features and environmental factors, as well as integration of real-world sea surface temperature data to add realistic variability.

In procedural generation, we simulate a wide range of hypothetical surface temoerature maps to create varied scenarios that the agent could encounter in a real-world setting. For instance, we introduce changes in temperature distribution thereby encouraging the agent to adapt its sampling policy to a wide variety of layouts and data features. By simulating numerous variations, the agent develops a generalized approach that remains effective even when confronted with previously unseen maps.

We supplement this synthetic map generation with maps from a real world, specifically using historical sea surface temperature measurements. This data provides a realistic basis for training, bridging the gap between simulated and actual aquatic conditions.

In addition to increasing generalization, domain randomization also contributes to robustness in policy performance. By training over this varied map distribution, the agent becomes adept at adapting its strategy dynamically in response to environmental changes. Thus, the policy learns not only to locate high-interest sampling areas efficiently but also to respond flexibly to different maps.

Our experiments, discussed in §5.1, verify this approach. The agent trained through domain randomization demonstrates an effective sampling strategy on maps outside its training set, including a real-world test on a freshwater lake in Québec, Canada, as presented in §6. The results confirm that domain randomization enables the agent to produce consistent and

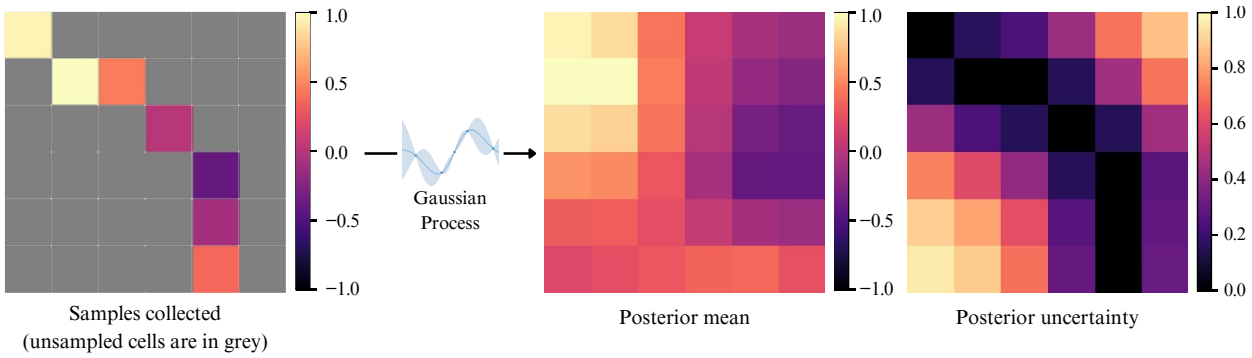


Figure 4.2: An example of using posterior inference to infer the posterior mean and the diagonal of the posterior covariance from the samples collected at time $t = 7$ with a 6×6 map discretization.

efficient sampling patterns even in complex, previously unseen environments.

4.6 Implementation details

We describe important implementation considerations for our method. Notably, using our GP posterior parameterization, the inputs of the policy and Q-function described in §4.4 (namely, $x_t, \mu_t, \Sigma_t, V_t$) can each easily be expressed as tensors. This makes them amenable to parameterization of the Q-function with a deep neural network for DQN training. To reduce memory footprint and simplify the structure of the agent state, we substitute the full covariance Σ_t of the map posterior with its diagonal, so that the input to the policy and Q-function is an element of $\mathbb{R}^{4\times N\times N}$ for $N \times N$ map discretizations, where the position x_t is expressed via one-hot encoding. The action space is encoded as $\{1, \ldots, 8\}$, describing the neighboring cells. Figure 4.2 is an illustration of using posterior inference to infer the posterior mean μ_t and the diagonal of the posterior covariance Σ_t from the samples collected. We experiment with both multi-layer perceptron (MLP) and convolutional neural networks (CNN) architectures for modeling the Q-function. To reduce the dimensionality of the state space, we also experimented with a *multi-resolution feature aggregation* technique of Manjanna and Dudek [22] for encoding the posterior, where we average posterior means and variances across larger collections of cells further away from the agents current position. A diagram of this technique is shown in Figure 4.3. The multi-resolution feature aggregation method streamlines policy search by structuring the environment's state space into multiple levels of detail. This approach reduces the state spaces dimensionality, enabling efficient learning by retaining critical details in close areas to the boat while maintaining a broader overview of further locations. This multi-level organization balances computational efficiency with necessary detail, making it ideal for adaptive sampling tasks in dynamic environments,

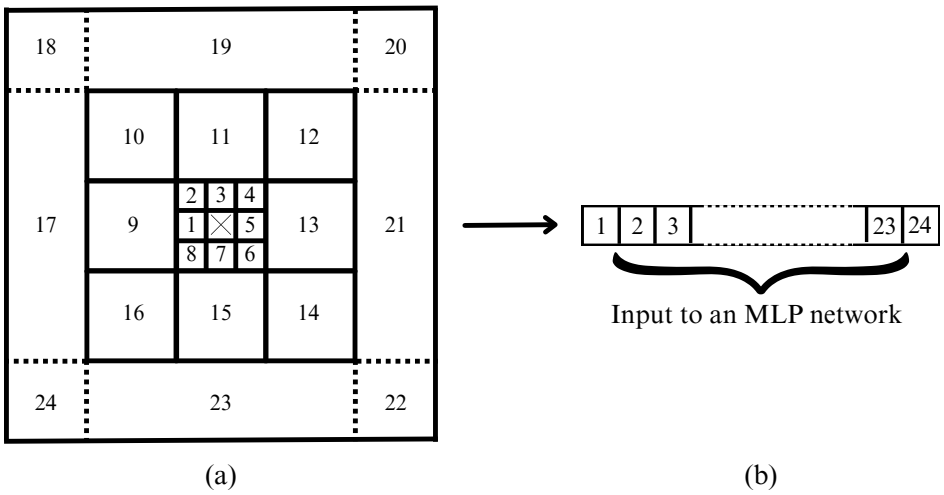


Figure 4.3: Multi-resolution feature aggregation parameterization of the state.

where both accuracy and adaptability are essential for capturing critical environmental data.

Under this parameterization of the agent's state, the state is no longer a rectangular tensor, see that in Figure 4.3 (a). The multi-resolution feature aggregation outputs a 1D array so we use an MLP on flattened observations to model the Q-function. The value of each element in the 1D array is the average of all the cells that are covered by that element. For example, consider the the boat is at the cell m, n, the element representing the 12th feature in Figure 4.3 (a) is calculated as:

$$feature_{12} = \frac{1}{9} \sum_{i=m-4}^{i=m-1} \sum_{j=n+1}^{j=n+4} M(x_{i,j}).$$

Some features might be assigned a zero if the area they cover is empty which can happend when the boat is close to the boundary of the map. We apply multi-resolution feature aggregation to each component of the observed state $o_t = (x_t, \mu_t, \Sigma_t, V_t)$, transforming each tensor into a 1D array. These arrays are then concatenated to form a single input vector for the MLP network.

In contrast, the CNN takes the full observed state tensor as input without flattening. For the MLP, each component of the observed state is first flattened, and then these flattened arrays are concatenated to serve as the input to the network.

Experimenting with MLP, CNN, and multi-resolution feature aggregation MLP allows us to explore different methods for balancing computational efficiency with effective environmental representation. The MLP serves as a straightforward approach, treating each feature independently to gauge if a simple model can effectively guide decision-making without spatial context. In contrast, the CNN captures spatial dependencies, making it ideal for scenarios where relationships between locations in the map are crucial, potentially enhancing sampling decisions through structured spatial learning. Finally, the multi-resolution feature aggregation MLP combines both local and broader environmental details by processing features at varying scales, enabling the model to focus on immediate details while maintaining an overview of the environment which is key for efficient adaptive sampling in complex settings.

5

Experimental Setup

In this chapter, we outline the setup for simulating, training, and evaluating our approach. First, we introduce the simulator environment, which emulates realistic sampling and navigation tasks. By replicating essential conditions of the target environment, the simulator provides a controlled setting to test various strategies. We then detail the datasets utilized, including both procedurally generated and real-world sources, and describe their relevance in training the agent for adaptive sampling tasks.

Our training procedure leverages these datasets to ensure the agent learns effective policies. We also discuss the evaluation methodologies and baselines applied to assess the agent's performance on both procedurally generated and real-world environments. This comprehensive approach, covering simulation, dataset design, and rigorous evaluation, establishes a reliable framework for testing and validating the effectiveness of our approach.

5.1 Training Simulator and Datasets

For the purpose of training an active sampling policy, we constructed a simulator adhering to the Gymnasium [38] interface to comply with RL libraries. Our simulator is initialized with a random map M of a property of interest in a rectangular body of water, that is discretized uniformly into a grid. As discussed in §4.2, the available actions at position x

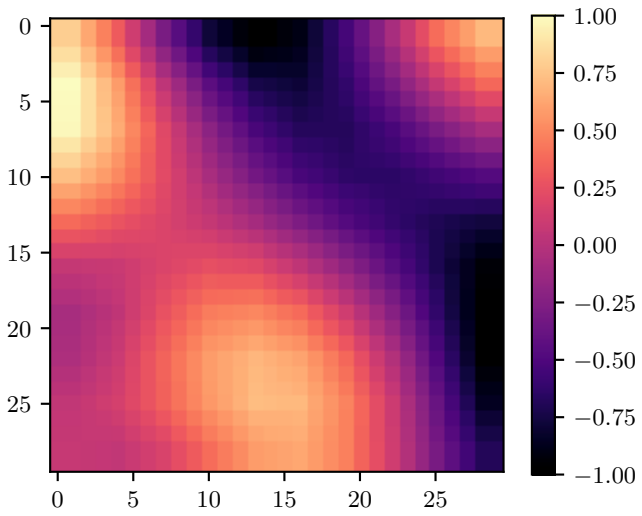


Figure 5.1: An example of a 30x30 map from the procedurally-generated dataset used for training in simulation.

are the neighboring cells $\mathsf{N}(x)$, and the cost function is given by (4.2). At every step, the agent selects an action, and observes the incurred cost, as well as the next position x and the value of $M(x)$, which the agent uses to update its *partial map* U and its posterior over maps.

At the beginning of each episode, the underlying map M of the property is randomly sampled for the purpose of domain randomization, as discussed in §4.5. In our simulation experiments, we use both procedurally-generated maps and real surface temperature maps from the Atlantic Ocean, effectively promoting generalization to a broad class of potential maps that can be seen at deployment.

5.1.1 Procedurally-Generated Maps

We randomly sample maps from a GP prior (cf. §4.3). Notably, in order to robustify our agent to misspecification of its prior over maps, we do not use the same kernel κ for generating the maps; rather, we use an RBF kernel Equation (3.2) for bandwidth parameter $\ell > 0$. Figure 5.1 depicts an example of a procedurally-generated map.

5.1.2 NOAA Surface Temperature Maps

We use the National Oceanic and Atmospheric Administration (NOAA) Daily Global 5km Satellite Sea Surface Temperature dataset [19], which provides daily sea surface temperature maps of the Atlantic Ocean bordering North and Central America as well as the Caribbean. During training, we independently sample a day and further subsample a random crop of unobstructed sea from the map on that day to instantiate our simulation. We normalize the

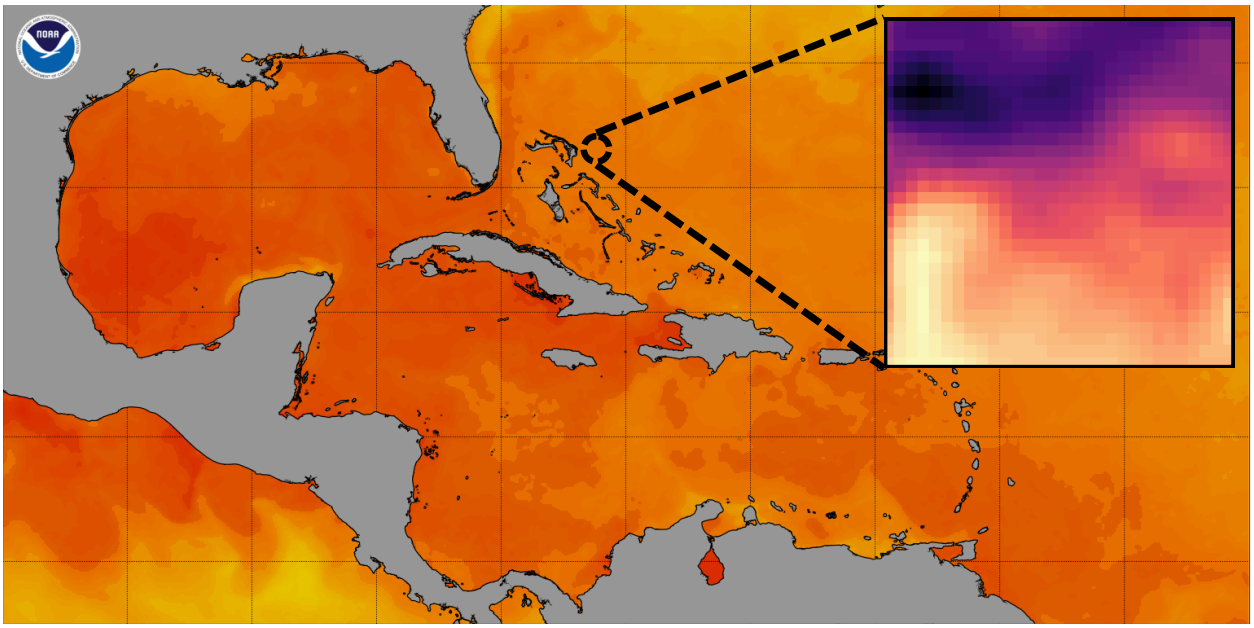


Figure 5.2: An example of the sea surface temperature (SST) map of the Caribbean area from the NOAA dataset used in simulation experiments.

temperatures to the range $[-1, 1]$. Figure 5.2 depicts a sample of a daily surface temperature map from the NOAA dataset, as well as a randomly subsampled and normalized crop. We should note that in our experiments with the NOAA dataset, we reserve a set of maps specifically for evaluation to ensure that no test maps are observed during training. Policies are trained on maps from 2000 to 2015, and evaluation is conducted on maps from 2016 to 2023. This division allows us to assess the generalization of the trained policies to unseen data.

5.2 Training and Evaluation Procedure

Our experiment is comprised of two main phases: a domain-randomized policy training phase, and an evaluation of the resulting policy on unseen maps.

5.2.1 Policy training

We train a posterior-map-conditioned policy as described in §4.4 using DQN, using the domain randomization strategy outlined in §4.5 with random maps sampled according to §5.1. The maps are discretized to 30×30 grids. Training occurs entirely in simulation.

5.2.2 Evaluation

After extracting a policy from the training phase §5.2.1, we deploy the policy on an unseen map and evaluate the progression of its map reconstruction RMSE as a function of environment steps taken.

We perform the evaluation both on held-out maps from the NOAA dataset (cf. 5.1.2) and on a region of Lac Hertel at the Gault Nature Reserve (a lake in Québec, Canada). For the evaluation on Lac Hertel, we deployed an autonomous BlueBoat from Blue Robotics, shown in Figure 5.3. that consists of the ASV, BlueBoat equipped with a tether and connected to a sensor unit. The sensor unit contains a Raspberry Pi for communication and rosbag recording purposes, depth and temperature sensors for collecting samples. The evaluation ground truth map was generated by gathering temperature data along a comprehensive boustrophedon path, covering a 90m by 20m area. Ultimately, we show in §6 that our synthesized policy accurately reconstructs the map with far fewer samples than the exhaustive boustrophedon. We report the variance of the resulting reconstruction error across both random evaluation maps (in the case of the NOAA evaluations) and across random seeds used for synthesizing the policies.

5.3 Baselines

In this section, we describe the baseline methods that we evaluate our approach against. Since our method is the first learning-based method to our knowledge that works on unknown maps, the baseline methods do not involve any training phase.

5.3.1 Boustrophedon

This method [21] navigates the map exhaustively in a "lawmnmower" pattern (see Figure 5.4). It is a widely used baseline for coverage path planning in robotic systems. The boustrophedon pattern is especially useful in environements where obstacles are minimal as it involves moving in a systematic back-and-forth motion, covering an area by traveling in straight lines and making sharp turns at the boundaries to ensure complete coverage. This method offers a simple, yet effective, approach for ensuring comprehensive exploration of large areas in open waters. For example, in tasks like monitoring water quality in reservoirs, lakes, or oceans, boustrophedon motion ensures that the entire area is scanned with minimal overlap, making it efficient and easy to implement.

As a baseline method, boustrophedon also serves as a useful tool for evaluating the efficiency and completeness of more advanced techniques, helping to establish a standard of

Figure 5.3: A picture of our robotic setup which includes the BlueBoat and the sensor unit.

comparison for real-world aquatic exploration tasks [5].

5.3.2 Highest-Uncertainty

This method proceeds by determining a waypoint which is the cell in the map with the greatest posterior uncertainty. It then follows the shortest path to that cell using a combination of diagonal, and straight paths. When a waypoint is reached, it selects a a new waypoint and so on. Figure 5.5 is an example of a highest-uncertainty path.

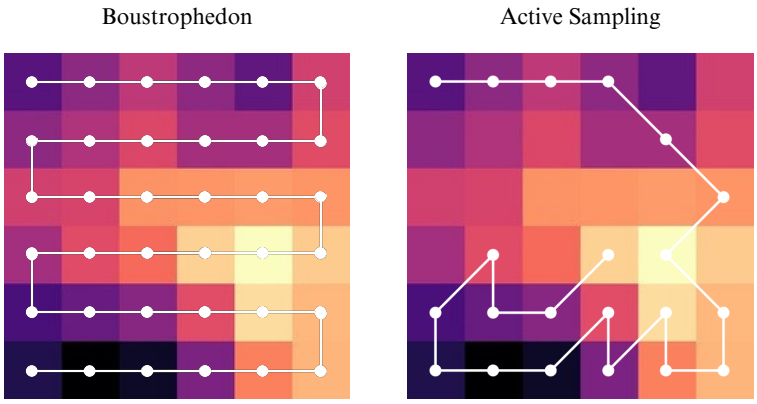


Figure 5.4: An example of a boustrophedon path and an active sampling path.

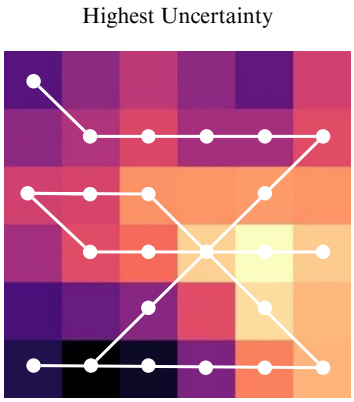


Figure 5.5: An example of a highest-uncertainty path.

The highest-uncertainty baseline aims to reduce the highest uncertainty in the map at the time of action selection. A key drawback of this method is its narrow focus on selecting only the waypoints with maximum uncertainty, without leveraging the information gathered en route to those waypoints. Moreover, by ignoring cells that have high uncertainty but not the maximum uncertainty, the method potentially misses opportunities to gather useful data while reducing travel distance.

5.3.3 Model Predictive Control

We employ *model predictive control (MPC)* as a basline for planning, leveraging its capability to optimize sequences of actions over a defined horizon. At each timestep t, MPC formulates an optimization problem that seeks to maximize the accumulated reward over a

finite sequence of actions of length $N = 30$. This optimization occurs from the current state s, taking into account the potential N future locations that could be reached through the selected actions. The reward function used in this context is defined as $r(s) = \Sigma_{x,x}$, where Σ represents the posterior covariance matrix of the system state, specifically focusing on the variance at the predicted position x of the agent at state s. This formulation directly links the control policy to the uncertainty in state estimation, emphasizing that the objective is to minimize uncertainty as we make progress toward the target.

In quantitative terms, the optimization can be expressed as:

$$\max_{a_t, a_{t+1}, \ldots, a_{t+N-1}} \sum_{k=0}^{N-1} r(s_{t+k})$$

where a_k represents the actions taken at each timestep.

It is crucial to note that MPC is unable to account for the reduction in the uncertainty over the map over the course of the action sequence. This limitation contrasts with our proposed approach, which dynamically incorporates the evolving posterior distribution to enhance decision-making across multiple timesteps.

6

Results

In this chapter, we present detailed experimental results to evaluate the effectiveness of our approach for adaptive map reconstruction. In this chapter, we provide comprehensive experimental results to assess the performance of our three models — MLP, CNN, and multi-resolution feature aggregation MLP — in adaptive map reconstruction. We begin by presenting the simulation results §6.1, including the reconstruction error per step for each of our models and the baseline methods, using evaluations on the NOAA dataset. This includes models trained on both the procedurally generated dataset and NOAA dataset, as shown in Figures 6.1 and 6.2. Additionally, we provide trajectory examples for each dataset to illustrate the adaptability of our models to different environments. In §6.2, we present evaluation results from field data collected over a 90m by 20m area of Lac Hertel, where we again compare reconstruction error per step across our models and baselines. Sample trajectories from each method and baseline are included to offer insight into their respective adaptive sampling behaviors. Finally, in §6.3, we discuss the the evaluation results' similarities and differences between datasets. We also highlight the trends across all datasets, procedurally generated, NOAA, and Lac Hertel datasets.

The results in this chapter confirm our hypothesis that the outlined methods in §4 and §5 enable accurate and data-efficient map reconstruction in aquatic environments. Specifi-

cally, §6.1 showcases our model's performance across a diverse set of procedurally-generated maps and maps from the NOAA dataset. §6.1 demonstrates both the accuracy of the reconstructed maps and the efficiency of the sampling strategy, illustrating the progressive reduction of reconstruction error over time. Additionally, in §6.2, the results from field experiments conducted in Lac Hertel provide further validation of our approach in real-world settings. We analyze the sampling paths and examine how well the models balance exploration with targeted sampling to minimize the reconstruction error. Both §6.1 and §6.2 include visualizations of the sampling paths generated by our models, offering qualitative insights into their decision-making and adaptive behavior. All reported reconstruction errors represent the mean and standard deviation over five random seeds, with variations arising from policy training in DQN and action sequence generation in the MPC-based approach, enabling a robust analysis of our models' consistency and reliability.

6.1 Simulation Results

In this section, we analyze the simulation results to evaluate the performance of our three adaptive map reconstruction models — MLP, CNN, and multi-resolution feature aggregation MLP — compared to several baseline methods — boustrophedon, highest uncertainty, and MPC. Our simulations are based on two distinct datasets: the NOAA surface temperature dataset and a procedurally generated dataset designed to diversify training environments. Through these simulations, we assess each model's reconstruction accuracy and sampling efficiency by tracking reconstruction error over time. We also include sample trajectory visualizations to provide insight into each model's approach to adaptive sampling across different environmental conditions.

In Figure 6.1, we compare the reconstruction error per step for our three adaptive sampling models against baseline approaches on the NOAA dataset. The error metric used here is the RMSE, which quantifies the difference between the reconstructed and actual environmental field values at each step of the 200 steps per episode. Lower RMSE values indicate better map reconstruction accuracy. The baseline methods shown are boustrophedon (dashed blue line), highest uncertainty (dashed orange line), and MPC (dashed green line). Each of these baselines follows different sampling strategies. The boustrophedon baseline performs exhaustive coverage by systematically scanning the environment, which initially results in high reconstruction error due to inefficient sampling but gradually improves as it completes coverage. The highest uncertainty baseline actively selects sampling points with high uncertainty, aiming to reduce error in unknown regions, which results in slightly faster initial error reduction compared to boustrophedon but its path to the regions with the high-

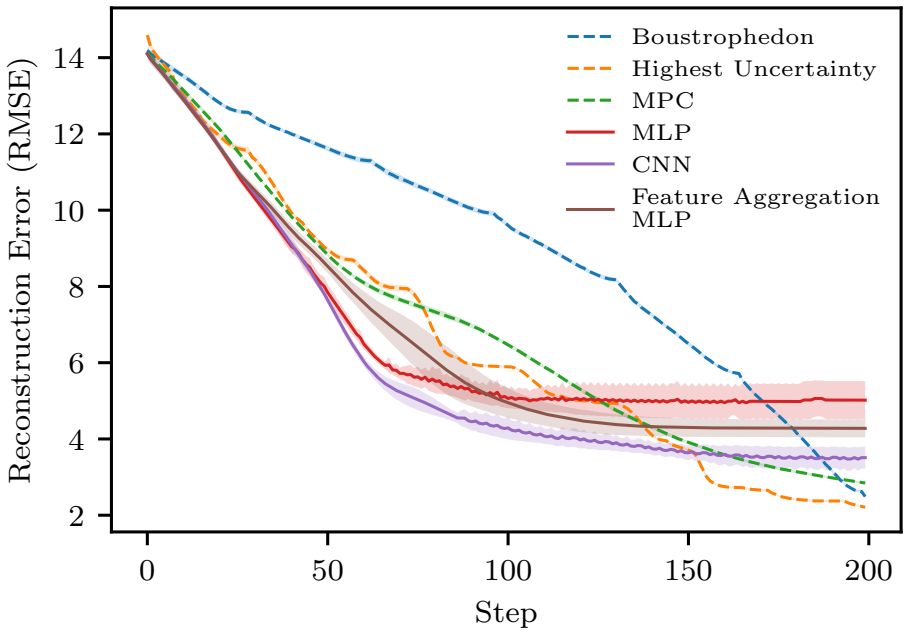


Figure 6.1: Reconstruction error per step for our method (solid lines) and baselines (dashed line) evaluated on the NOAA dataset using a model that was trained in simulation on the NOAA dataset.

est uncertainty is not optimized. This can be observed in the figure, where the step pattern illustrates that the recontruction error is lowered when a high uncertainty region is reached but the reconstruction error is moslty stable during the steps when the agent is trying to reach the target region. The MPC baseline takes a predictive approach but does not adapt as effectively in this dataset, leading to slower convergence of error compared to our models.

Among our models, the feature aggregation MLP (solid brown line) and CNN (solid purple line) demonstrate lower reconstruction error than both the baselines and the MLP model (solid red line). This improvement suggests that the feature aggregation MLP and CNN models are more effective in adaptive sampling by focusing on relevant areas with higher environmental variability. The CNN model, in particular, achieves the fastest convergence to a low RMSE, showing it can efficiently reduce error over time.

Overall, our adaptive models outperform traditional baselines, with the feature aggregation MLP and CNN demonstrating superior efficiency in reconstruction. This indicates that these models' ability to generalize spatial patterns is beneficial for environmental monitoring applications where adaptive and efficient data collection is critical.

In Figure 6.2, we observe the reconstruction error per step during the evaluations of the

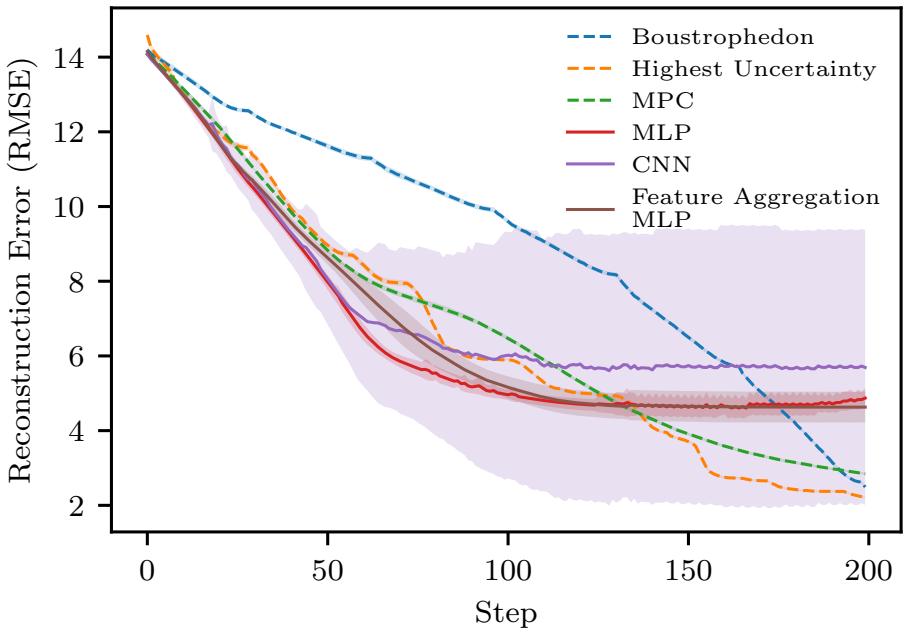
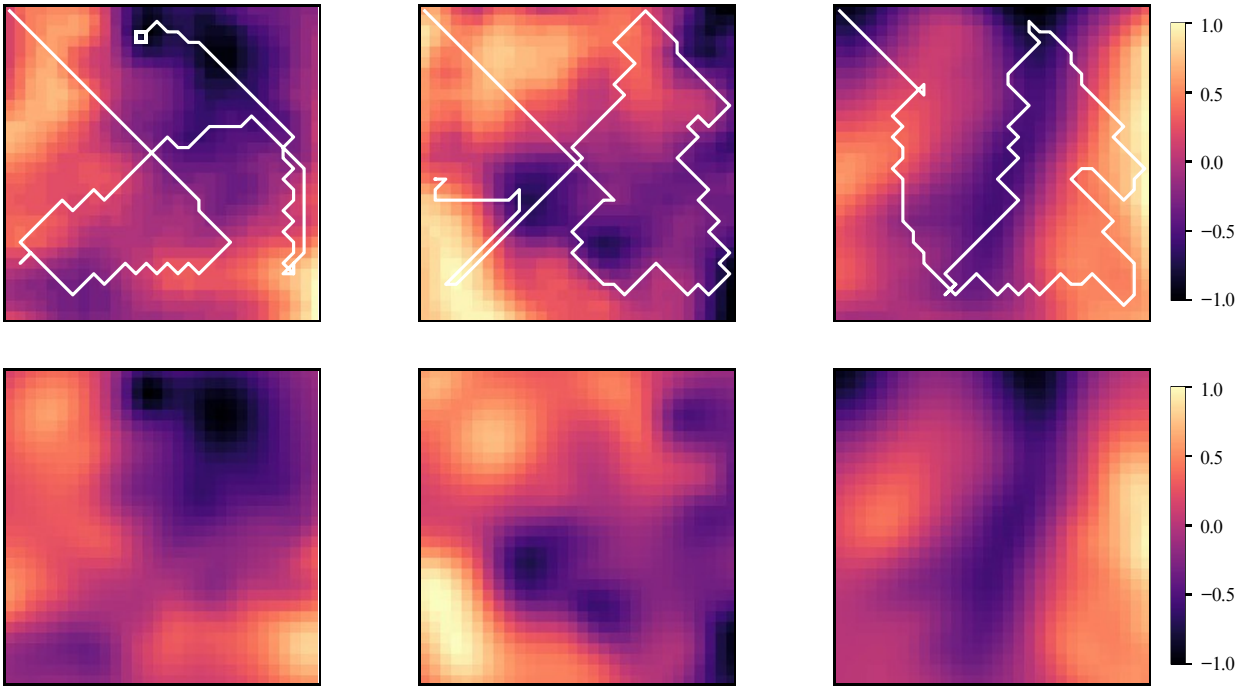


Figure 6.2: Reconstruction error per step for our method (solid lines) and baselines (dashed line) evaluated on the NOAA dataset for a model trained in simulation on procedurally-generated data.

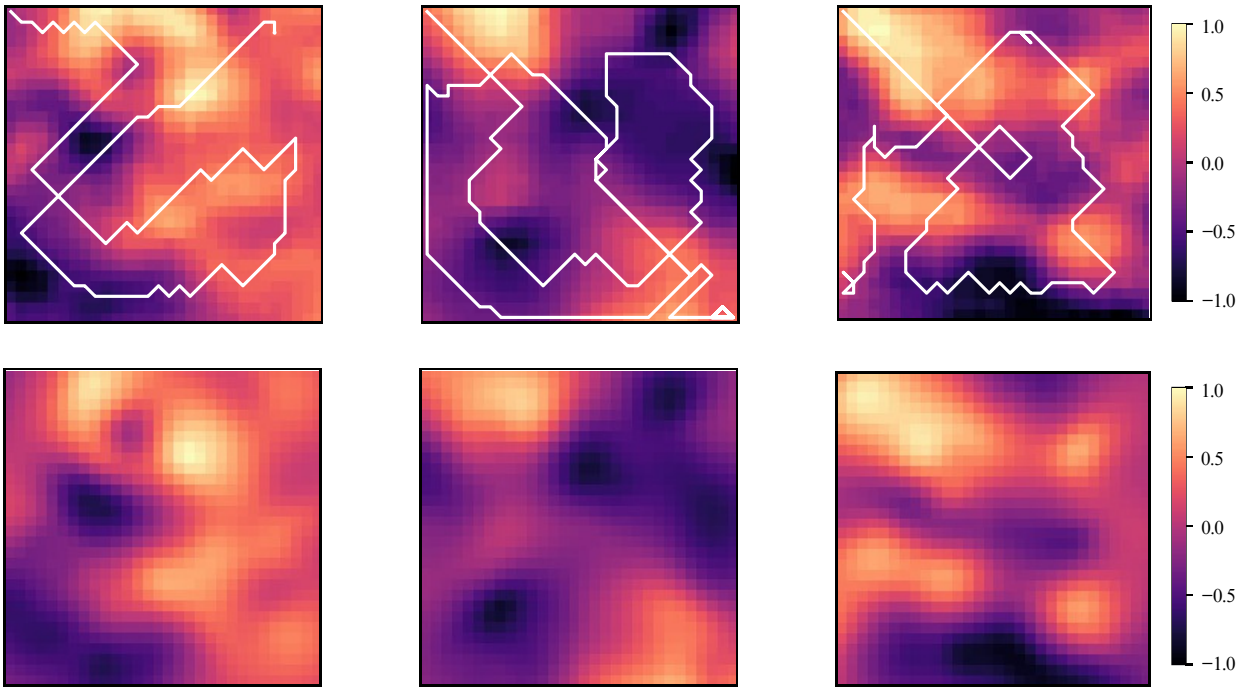
NOAA dataset for models trained on procedurally generated data. This figure illustrates the performance of our three models compared to baseline methods using dashed lines for the baselines and solid lines for our models.

Overall, our models demonstrate a faster gradual reduction in reconstruction error over time, converging toward low error values by step 150 out of the 200 evaluation steps per episode. Notably, the CNN and feature aggregation MLP models exhibit consistent improvement, reaching lower error levels than the baselines. The discussion regarding the baselines results above applies here which emphasizes the advantage of using learned models which can adaptively adjust their sampling patterns to more accurately reconstruct the environment.

Adding to this analysis, visualizations of the sampling trajectories in Figures 6.3a and 6.3b further highlight the strategic adaptation achieved by the DQN-based models. Unlike the rigid, predetermined sampling patterns of baseline methods, our models dynamically adjust their paths in response to the information gathered from each sample. This adaptive behavior allows them to effectively balance coverage between unexplored regions and areas with high uncertainty, enabling more efficient data collection.



(a) Evaluation trajectories on 3 different sea surface temperature maps from the NOAA dataset.



(b) Evaluation trajectories on 3 different sea surface temperature maps from the procedurally generated dataset.

Figure 6.3: Evaluation trajectories on sea surface temperature maps. In each of the subfigures, the upper row is the ground truth of the sea surface temperature maps. The trajectory of the DQN agent is shown on the ground truth maps. The lower row is the DQN agent's final estimate of the maps.

In the final map estimates, displayed in the lower row of each subfigure, we see close alignment with the actual sea surface temperature maps (shown in the upper rows), even in regions that were not directly sampled. This level of alignment suggests that the DQN agents successfully generalize achieving accurate reconstructions with relatively few samples.

Across all trajectory examples, we observe a common initial sampling pattern where the models begin with a diagonal path, covering roughly a quarter to half of the full diagonal. This early strategy provides broad initial coverage, after which the DQN agents diverge into more adaptive loops that target specific areas based on the posterior mean estimates. This shared diagonal starting approach suggests an efficient, systematic initiation across models. However, as sampling progresses, only the DQN agents trained with adaptive policies continue to strike an effective balance between exploration and data efficiency. In contrast, The baseline methods — boustrophedon and highest uncertainty — follow a rigid path regargless of the environment while the MPC baseline follows paths that differ from one environment to the other due to the randomness when selecting action sequences. This shows that the baseline methods tend to follow less flexible paths, reflecting their limited adaptability in this context.

6.2 Field Test Results

For our field tests, we deployed an ASV to gather surface temperature data from Lac Hertel, located within the Gault Nature Reserve. This survey leveraged the ASV's precise waypoint-following controller to cover the area effectively, resulting in comprehensive surface temperature maps, as shown in Figures 6.5 and 6.6.

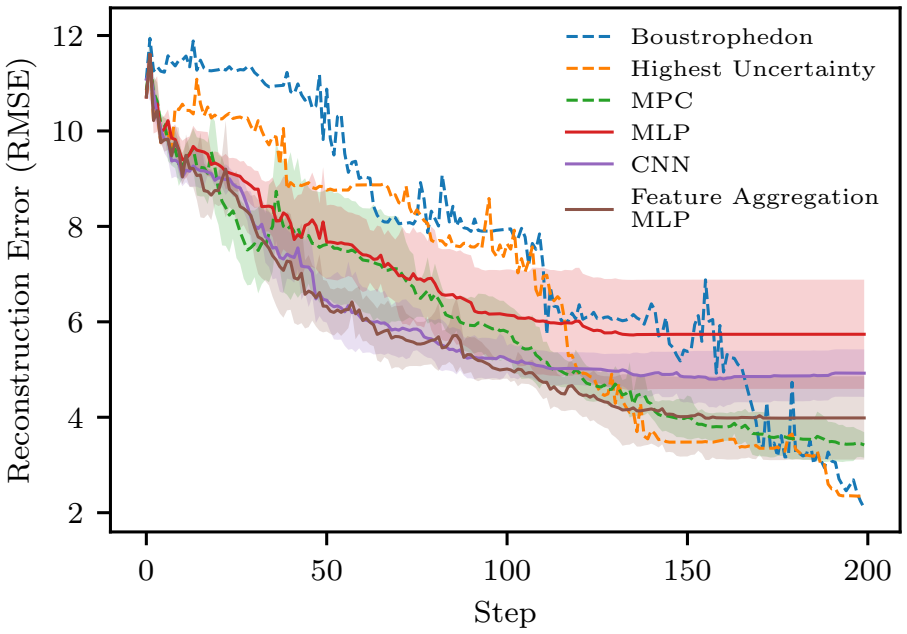


Figure 6.4: Reconstruction error per step for our method (solid lines) and baselines (dashed line) evaluated on Lac Hertel data.

In these tests, we evaluated our DQN-based agents, which were trained on historical NOAA dataset maps, alongside various baseline methods on the Lac Hertel temperature map. Figure 6.4 shows the reconstruction error per step for various models and baseline methods. Similar to the previous figures, it provides insights into the performance of our models (solid lines) compared to the baselines (dashed lines) over the course of the sampling episode.

Initially, the reconstruction errors for all methods start relatively high, reflecting the limited information available at the beginning of sampling. As the sampling progresses, we observe a consistent decrease in error across all methods, with notable differences in the rate and extent of error reduction. The baselines, such as the boustrophedon and highest uncertainty approaches, show a more variable decline, with a pattern of repeatedly decreasing reconstruction error for a few episodes then the reconstruction error plateaus for a few episodes and so one. This pattern can be explained by their rigid path, which only provides informative samples that significantly lower the reconstruction error, after every row in boustrophedon or after reaching intermediate high-uncertainty target locations in the highest uncertainty method. This pattern shows that these methods lack a consistent adaptive approach to minimizing error efficiently, as they do not account for evolving environmental

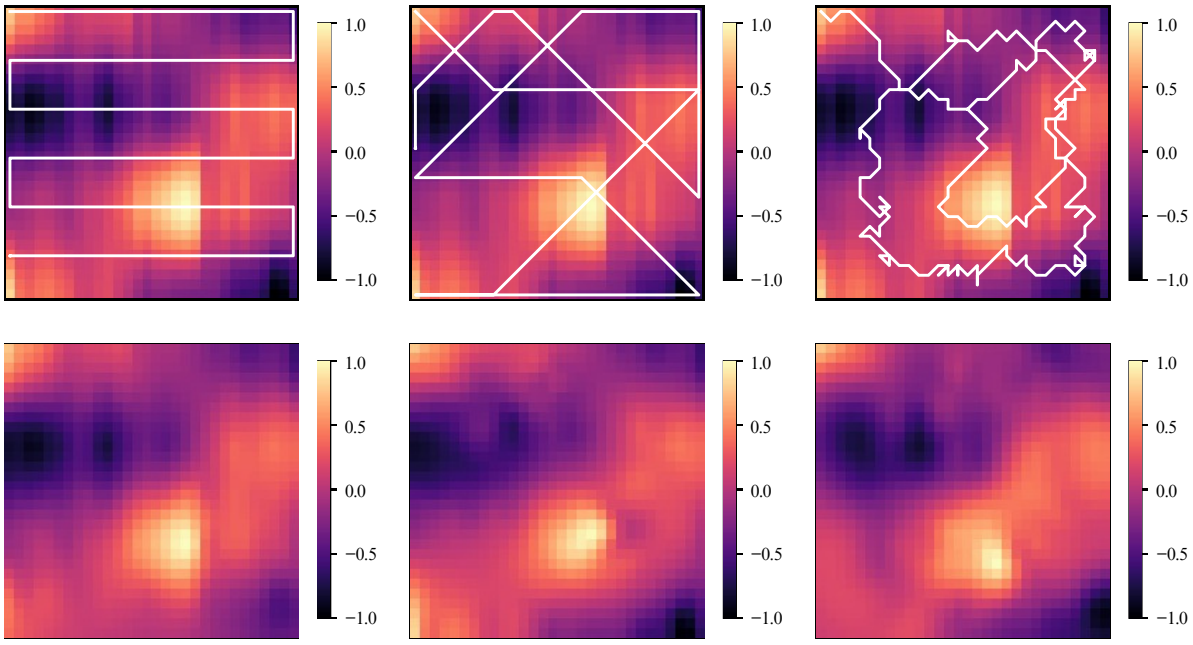
information as effectively as the learned models.

Among our models, the feature aggregation MLP consistently achieves lower reconstruction error compared to the MLP and CNN models, demonstrating the advantage of leveraging multi-resolution feature aggregation in this task, especially that the models were not trained on data similar to the temperature map from Lac Hertel. By contrast, the CNN model, while initially performing similarly to the feature aggregation MLP, eventually stabilizes at a slightly higher error level. This suggests that the CNN model may be less efficient in generalizing to the structure of the Lac Hertel temperature map compared to the multi-resolution feature aggregation model.

This figure highlights the effectiveness of our adaptive sampling models in reducing reconstruction error at a fast rate, with the feature aggregation MLP exhibiting the best performance. This result highlights the power of multi-resolution feature aggregation for training models that generalize to previously unseen maps while still maintaining a small network architecture. Overall, this outcome underscores the importance of incorporating adaptive learning and feature aggregation techniques to improve map reconstruction accuracy, especially in environments with complex or variable characteristics like Lac Hertel.

In the Lac Hertel setting, we show a visualization of sampling trajectories of the baselines in Figure 6.5 and our models in Figure 6.6 to further highlight the adaptive sampling strategy employed by the DQN-based agents. Unlike the baselines, the DQN agents exhibit a balanced approach by prioritizing key regions without becoming overly focused on any single area. This adaptive selection process allows the DQN agents to achieve accurate map reconstructions without requiring exhaustive coverage, thereby enhancing both sampling efficiency and reconstruction fidelity.

The trajectories of the DQN agents reflect this efficient approach by showing targeted exploration that effectively balances exploration and exploitation, optimizing both spatial coverage and data collection. For comparison, in Figure 6.5, we observe the rigid paths followed by the boustrophedon and highest-uncertainty methods, which lack adaptive adjustments and remain consistent across different maps. Although the MPC agent introduces some variability in its paths due to randomized action sequence selection, it still does not achieve the level of strategic adaptation demonstrated by the DQN agents. This visualization emphasizes the superiority of the DQN-based methods in creating accurate and efficient map reconstructions in complex environments like Lac Hertel.



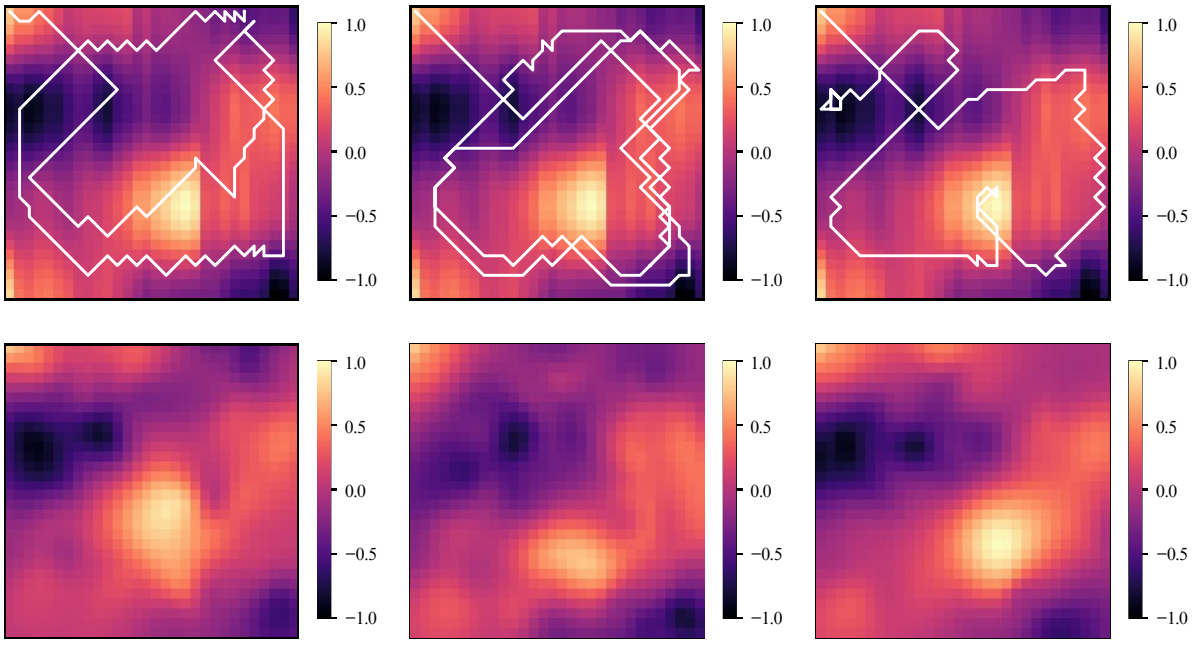
(a) Boustrophedon (b) Highest-Uncertainty (c) MPC

Figure 6.5: Evaluation trajectories of the baslines on the temperature map at Lac Hertel. In each of the subfigures, the upper plot is the ground truth of the sea surface temperature maps. The trajectory of the agent is shown on the ground truth maps. The lower plot is the agent's final map estimate.

6.3 Trends Across Datasets

The reconstruction error trends across the procedurally generated, NOAA, and Lac Hertel datasets provide several important insights into the relative performance of each sampling method. Across all reconstruction error plots (Figures 6.1, 6.2 and 6.4), we observe that the boustrophedon approach, which follows a fixed lawn-mower pattern, consistently shows the slowest reduction in error. This systematic method does not account for the information value of each sampled point, as it relies on a predefined path rather than prioritizing high-information areas. Consequently, boustrophedon's lack of adaptive sampling results in a slower convergence rate.

In contrast, the MPC and highest uncertainty baselines achieve faster error reduction by incorporating adaptive sampling strategies that focus on high-uncertainty areas. By dynamically prioritizing these regions, they improve reconstruction accuracy more quickly than the non-adaptive boustrophedon method. However, while both of these baselines incorporate an element of adaptive behavior, they do not match the level of efficiency demonstrated by the DQN-based agents.



(a) MLP (b) CNN (c) Feature Aggregation MLP

Figure 6.6: Evaluation trajectories of our method on the temperature map at Lac Hertel. In each of the subfigures, the upper plot is the ground truth of the sea surface temperature maps. The trajectory of the agent is shown on the ground truth maps. The lower plot is the agent's final map estimate.

The DQN agents significantly outperform these baselines, particularly in the early sampling stages. They achieve lower reconstruction errors within the first 100—150 steps, showcasing the effectiveness of our models in obtaining accurate reconstructions with limited data. This adaptive behavior is a key strength of our approach, as it enables the DQN agents to dynamically identify and prioritize high-utility sampling locations, rather than relying on preset trajectories.

Among the DQN models, the MLP-based agent consistently exhibits faster error reduction with limited samples compared to the baselines, demonstrating efficiency across both synthetic and real-world datasets. However, distinct performance differences emerge between the CNN and multi-resolution feature aggregation MLP models depending on the dataset. On the NOAA dataset, the CNN model performs better, likely due to its capacity to capture spatial dependencies effectively in structured, real-world environments. Conversely, on procedurally generated data, all three models achieve comparable results, reflecting the generalized adaptability of our approach across different environments.

On the Lac Hertel dataset, the multi-resolution feature aggregation MLP outperforms

the CNN-based agent, suggesting an advantage in adapting to complex, heterogeneous environments. This model's ability to aggregate features across multiple spatial scales allows it to capture intricate, multi-scale environmental patterns that are prevalent in previously unseen real-world conditions. By leveraging this feature aggregation, the feature aggregation MLP-based model dynamically adjusts its sampling strategy in response to the environmental heterogeneity, thereby enhancing map reconstruction accuracy relative to the CNN-based approach.

In summary, these results demonstrate that our adaptive sampling approach effectively reduces reconstruction error with limited data. This adaptive strategy allows our agents to outperform traditional baselines by focusing sampling efforts on high-utility regions, leading to efficient and accurate environmental reconstructions with a much smaller number of samples across a range of datasets.

7

Discussion, Conclusion, and Future Work

In this work, we introduced a novel method for active sampling in aquatic environments that integrates DQN for path planning and GPs for reconstructing maps of a water quality. By leveraging the power of RL, our method dynamically adjusts the sampling strategy based on the current state of knowledge, thereby enabling more efficient data collection. We demonstrated the effectiveness of this approach by applying it to a real-world dataset, the NOAA sea surface temperature dataset, and validating it with data collected from Lac Hertel, Quebec. We showed that our method can efficiently reconstruct environmental maps, outperforming several traditional techniques.

One of the key strengths of our method is its ability to balance exploration and exploitation effectively. This is a hallmark of RL algorithms, and in the context of adaptive sampling, it means the agent is able to explore areas where data is sparse while also focusing on regions that are likely to yield the most valuable information. The integration of the GP model into the sampling process ensures that uncertainty in the unobserved areas is systematically reduced.

The results from our simulations and real-world validations support the effectiveness of this approach. In particular, the comparison with traditional sampling methods such as boustrophedon, MPC, and highest-uncertainty revealed significant efficiency improvements. In our experiments, our method achieved the same reconstruction accuracy at step 50 that the boustrophedon method required 150 steps to reach.

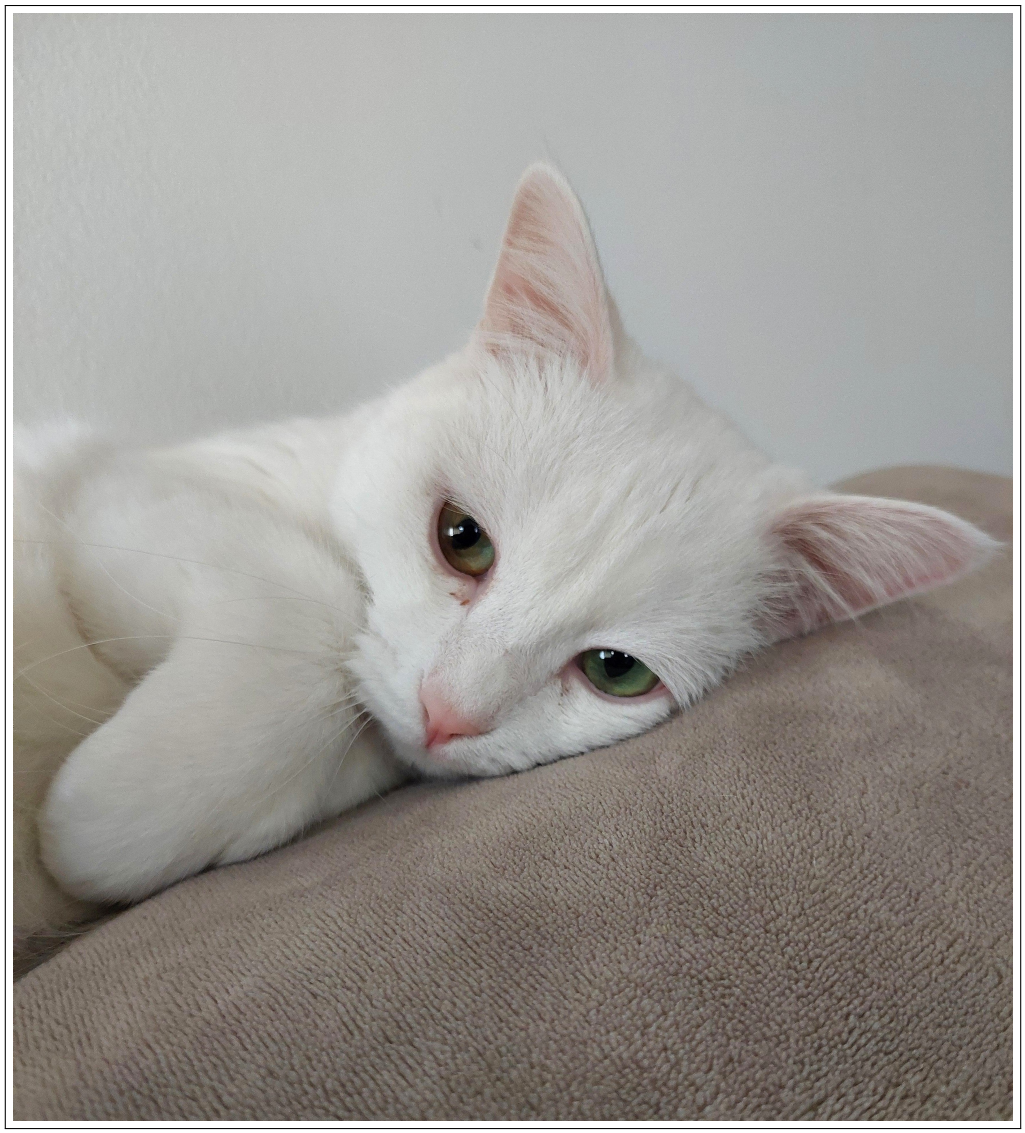
These results highlight the ability of our method to significantly reduce the time and resources required for high-resolution environmental monitoring, which is particularly important in applications where resources such as battery life and operational time are limited.

The strengths of our method are evident in its ability to efficiently reconstruct spatial maps of environmental variables while maintaining high accuracy with fewer samples. The combination of RL for path planning and GPs for map reconstruction and uncertainty quantification offers a robust framework for adaptive sampling.

One of the limitations of our method lies in the generalization of our method to different environmental variables. While we demonstrated the effectiveness of the approach for mapping sea surface temperature, other environmental variables, such as dissolved oxygen levels or pollutant concentrations, may exhibit different spatial dynamics and require adjustments to the model. In particular, some variables may be more affected by external factors such as wind, tides, or human activity, which would need to be incorporated into the agent's decision-making process.

Looking ahead, there are several promising avenues for extending this work. One exciting direction is to apply the method to 3D mapping of environmental variables, such as dissolved oxygen levels, which play a crucial role in determining the health of aquatic ecosystems [41]. Mapping dissolved oxygen levels in three dimensions is more challenging than mapping surface temperature, as it requires the robot to navigate and sample at different depths. This would involve modifying the RL agent to operate in a 3D environment, as well as extending the GP model to account for the vertical structure of the water column.

In conclusion, our work has demonstrated the potential of reinforcement learning and Gaussian Processes for improving the efficiency and accuracy of environmental monitoring. By combining intelligent path planning with uncertainty quantification, our method offers a powerful tool for reconstructing spatial maps of environmental variables, with applications ranging from water quality monitoring to ecosystem management. The results presented here lay the groundwork for future advancements in adaptive sampling, opening the door to new possibilities for understanding and preserving our natural ecosystems.



Lilly

Bibliography

[1] Mahmoud Ali, Hassan Jardali, Nicholas Roy, and Lantao Liu. Autonomous navigation, mapping and exploration with gaussian processes. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu, editors, *Robotics: Science and Systems*, 2023. ISBN 978-0-9923747-9-2. URL http://dblp.uni-trier.de/db/conf/rss/rss2023.html#AliJRL23.

[2] Jonathan Binney, Andreas Krause, and Gaurav Sukhatme. Informative path planning for an autonomous underwater vehicle. In *IEEE International Conference on Robotics and Automation, ICRA 2010*, pages 4791–4796, 05 2010. doi: 10.1109/ROBOT.2010.5509714.

[3] Miguel Duarte, Jorge Gomes, Vasco Costa, Tiago Rodrigues, Fernando Silva, Víctor Lobo, Mario Monteiro Marques, Sancho Moura Oliveira, and Anders Lyhne Christensen. Application of swarm robotics systems to marine environmental monitoring. In *OCEANS 2016 - Shanghai*, pages 1–8, 2016. doi: 10.1109/OCEANSAP.2016.7485429.

[4] Matthew Dunbabin and Lino Marques. Robots for environmental monitoring: Significant advancements and applications. *IEEE Robotics & Automation Magazine*, 19(1): 24–39, 2012. doi: 10.1109/MRA.2011.2181683.

[5] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258–1276, 2013. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2013.09.004.

[6] Luíza Caetano Garaffa, Maik Basso, Andréa Aparecida Konzen, and Edison Pignaton de Freitas. Reinforcement learning for mobile robotics exploration: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 34(8):3796–3810, 2023. doi: 10.1109/TNNLS.2021.3124466.

[7] Chris Gaskett, David Wettergreen, and Alexander Zelinsky. Q-learning in continuous state and action spaces. In Norman Foo, editor, *Advanced Topics in Artificial Intelligence*, pages 417–428, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. ISBN 978-3-540-46695-6.

[8] Devin K. Grady, Mark Moll, and Lydia E. Kavraki. Extending the applicability of pomdp solutions to robotic tasks. *IEEE Transactions on Robotics*, 31(4):948–961, 2015. doi: 10.1109/TRO.2015.2441511.

[9] Johanna Hansen, Sandeep Manjanna, Alberto Quattrini Li, Ioannis Rekleitis, and Gregory Dudek. Autonomous marine sampling enhanced by strategically deployed drifters in marine flow fields. In *OCEANS 2018 MTS/IEEE Charleston*, pages 1–7, 2018. doi: 10.1109/OCEANS.2018.8604873.

[10] Gregory Hitz, Alkis Gotovos, François Pomerleau, Marie-Eve Garneau, Cédric Pradalier, Andreas Krause, and Roland Y. Siegwart. Fully autonomous focused exploration for robotic environmental monitoring. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2658–2664, 2014. doi: 10.1109/ICRA.2014.6907240.

[11] Maani Jadidi, Jaime Valls Miro, and Gamini Dissanayake. Gaussian process autonomous mapping and exploration for range sensing mobile robots. *Autonomous Robots*, 42, 02 2018. doi: 10.1007/s10514-017-9668-3.

[12] Stewart Jamieson, Jonathan P. How, and Yogesh Girdhar. Finding the optimal exploration-exploitation trade-off online through bayesian risk estimation and minimization. *Artificial Intelligence*, 330:104096, 2024. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2024.104096.

[13] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, 1998. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(98)00023-X.

[14] Thomas Kollar and Nicholas Roy. Using reinforcement learning to improve exploration trajectories for error minimization. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 3338–3343, 2006. doi: 10.1109/ROBOT.2006.1642211.

[15] Thomas Kollar and Nicholas Roy. Trajectory optimization using reinforcement learning for map exploration. *The International Journal of Robotics Research*, 27(2):175–196, 2008. doi: 10.1177/0278364907087426.

[16] Maartje C. Korver, Bernhard Lehner, Jeffrey A. Cardille, and Laura Carrea. Surface water temperature observations and ice phenology estimations for 1.4 million lakes globally. *Remote Sensing of Environment*, 308:114164, 2024. ISSN 0034-4257. doi: https://doi.org/10.1016/j.rse.2024.114164.

[17] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–284, 02 2008. doi: 10.1145/1390681.1390689.

[18] Mikko Lauri and Risto Ritala. Planning for robotic exploration based on forward simulation. *Robotics and Autonomous Systems*, 83:15–31, 2016. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2016.06.008. URL https://www.sciencedirect.com/science/article/pii/S0921889015301779.

[19] Gang Liu, Scott F. Heron, C. Mark Eakin, Jacqueline L. De La Cour, Erick F. Geiger, Kyle V. Tirak, William J. Skirving, and Alan E. Strong. NOAA Coral Reef Watch (CRW) Daily Global 5-km (0.05 degree) Satellite Coral Bleaching Heat Stress Monitoring Product Suite. NOAA National Centers for Environmental Information. Dataset, 2018. URL https://doi.org/10.25921/6jgr-pt28. CoralTemp.

[20] Kian Hsiang Low, John Dolan, and Pradeep Khosla. Adaptive multirobot wide-area exploration and mapping. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, volume 1, pages 23–30, 01 2008. doi: 10.1145/1402383.1402392.

[21] V.J. Lumelsky, S. Mukhopadhyay, and K. Sun. Dynamic path planning in sensor-based terrain acquisition. *IEEE Transactions on Robotics and Automation*, 6(4):462–472, 1990. doi: 10.1109/70.59357.

[22] Sandeep Manjanna and Gregory Dudek. Data-driven selective sampling for marine vehicles using multi-scale paths. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6111–6117, 2017. doi: 10.1109/IROS.2017.8206511.

[23] Sandeep Manjanna, Herke Van Hoof, and Gregory Dudek. Policy search on aggregated state space for active sampling. In Jing Xiao, Torsten Kröger, and Oussama Khatib, editors, *Proceedings of the 2018 International Symposium on Experimental Robotics*, pages 211–221, Cham, 2020. Springer International Publishing.

[24] Budiman Minasny and Alex B McBratney. The matérn function as a general model for soil variograms. *Geoderma*, 128(3-4):192–207, 2005.

[25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. doi: 10.1038/nature14236.

[26] Seunghyeop Nam, Tuan Anh Nguyen, Eunmi Choi, and Dugki Min. Shangus: Deep reinforcement learning meets heuristic optimization for speedy frontier-based exploration of autonomous vehicles in unknown spaces, 2024. URL https://arxiv.org/abs/2407.18892.

[27] Farzad Niroui, Ben Sprenger, and Goldie Nejat. Robot exploration in unknown cluttered environments when dealing with uncertainty. pages 224–229, 10 2017. doi: 10.1109/IRIS.2017.8250126.

[28] Lishuo Pan, Sandeep Manjanna, and M. Ani Hsieh. Marlas: Multi agent reinforcement learning for cooperated adaptive sampling. In Julien Bourgeois, Jamie Paik, Benoît Piranda, Justin Werfel, Sabine Hauert, Alyssa Pierson, Heiko Hamann, Tin Lun Lam, Fumitoshi Matsuno, Negar Mehr, and Abdallah Makhoul, editors, *Distributed Autonomous Robotic Systems*, pages 347–362, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-51497-5.

[29] Christos H Papadimitriou and John N Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

[30] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, August 2014. ISBN 978-1-118-62587-3.

[31] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 11 2005. ISBN 9780262256834. doi: 10.7551/mitpress/3206.001.0001.

[32] Florian Shkurti, Anqi Xu, Malika Meghjani, Juan Camilo Gamboa Higuera, Yogesh Girdhar, Philippe Giguere, Bir Bikram Dey, Jimmy Li, Arnold Kalmbach, Chris Prahacs, Katrine Turgeon, Ioannis Rekleitis, and Gregory Dudek. Multi-Domain Monitoring of Marine Environments Using a Heterogeneous Robot Team. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1747–1753, Algarve, Portugal, October 2012.

[33] Aarti Singh, Robert Nowak, and Parameswaran Ramanathan. Active learning for adaptive mobile sensing networks. volume 2006, pages 60–68, 01 2006. doi: 10.1109/IPSN.2006.244057.

[34] Michael L. Stein. *Interpolation of Spatial Data*. Springer Series in Statistics. Springer New York, New York, NY, 1999.

[35] Jennifer Sunday, Amanda Bates, and Nicholas Dulvy. Thermal tolerance and the global redistribution of animals. *Nature Climate Change*, 2:686–690, 09 2012. doi: 10.1038/nclimate1539.

[36] Lei Tai and Ming Liu. Mobile robots exploration through cnn-based reinforcement learning. *Robotics and Biomimetics*, 3, 12 2016. doi: 10.1186/s40638-016-0055-x.

[37] Selim Temizer, Mykel Kochenderfer, Leslie Kaelbling, Tomas Lozano-Perez, and James Kuchar. Collision avoidance for unmanned aircraft using markov decision processes*. 08 2010. ISBN 978-1-60086-962-4. doi: 10.2514/6.2010-8040.

[38] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

[39] Marco Trincavelli, Matteo Reggente, Silvia Coradeschi, Amy Loutfi, Hiroshi Ishida, and Achim J. Lilienthal. Towards environmental monitoring with mobile robots. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2210–2215, 2008. doi: 10.1109/IROS.2008.4650755.

[40] Kwangjin Yang, Seng Keat Gan, and Salah Sukkarieh. A gaussian process-based rrt planner for the exploration of an unknown and cluttered environment with a uav. *Advanced Robotics*, 27:431–443, 04 2013. doi: 10.1080/01691864.2013.756386.

[41] Na Zhao, Zemeng Fan, and Miaomiao Zhao. A New Approach for Estimating Dissolved Oxygen Based on a High-Accuracy Surface Modeling Method. *Sensors (Basel, Switzerland)*, 21(12):3954, June 2021. doi: 10.3390/s21123954.

[42] Karl Johan Åström. Optimal control of markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1):174–205, 1965. ISSN 0022-247X. doi: https://doi.org/10.1016/0022-247X(65)90154-X. URL https://www.sciencedirect.com/science/article/pii/0022247X6590154X.