THE UNIVERSITY OF
SYDNEY

Bayesian Optimisation for Planning in Dynamic Environments

Román Marchant Matus

A thesis submitted in fulfillment
of the requirements of the degree of
Doctor of Philosophy

THE UNIVERSITY OF
SYDNEY

Australian Centre for Field Robotics
School of Information Technologies
The University of Sydney

March 2016

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the University or other institute of higher learning, except where due acknowledgement has been made in the text.

Román Marchant Matus

7 March 2016

Abstract

Román Marchant Matus
The University of Sydney

Doctor of Philosophy
March 2016

Bayesian Optimisation for Planning in Dynamic Environments

This thesis addresses the problem of trajectory planning for monitoring extreme values of an environmental phenomenon that changes in space and time. The most relevant case study corresponds to environmental monitoring using an autonomous mobile robot for air, water and land pollution monitoring. Since the dynamics of the phenomenon are initially unknown, the planning algorithm needs to satisfy two objectives simultaneously: 1) Learn and predict spatial-temporal patterns and, 2) find areas of interest (e.g. high pollution), addressing the exploration-exploitation trade-off. Consequently, the thesis brings the following contributions:

Firstly, it applies and formulates *Bayesian Optimisation* (BO) to planning in robotics. By maintaining a *Gaussian Process* (GP) model of the environmental phenomenon the planning algorithms are able to learn the spatial and temporal patterns. A new family of acquisition functions which consider the position of the robot is proposed, allowing an efficient trajectory planning.

Secondly, BO is generalised for optimisation over continuous paths, not only determining *where* and *when* to sample, but also *how* to get there. Under these new circumstances, the optimisation of the acquisition function for each iteration of the BO algorithm becomes costly, thus a second layer of BO is included in order to effectively reduce the number of iterations.

Finally, this thesis presents *Sequential Bayesian Optimisation* (SBO), which is a generalisation of the plain BO algorithm with the goal of achieving non-myopic trajectory planning. SBO is formulated under a *Partially Observable Markov Decision Process* (POMDP) framework, which can find the optimal decision for a sequence of actions with their respective outcomes. An online solution of the POMDP based on *Monte Carlo Tree Search* (MCTS) allows an efficient search of the optimal action for multi-step lookahead.

The proposed planning algorithms are evaluated under different scenarios. Experiments on large scale ozone pollution monitoring and indoor light intensity monitoring are conducted for simulated and real robots. The results show the advantages of planning over continuous paths and also demonstrate the benefit of deeper search strategies using SBO.

Acknowledgements

Firstly I would like to thank my supervisor, Fabio Ramos. He has taught me in many ways how to conduct proper research. Fabio provided strong support and motivation during the duration of my candidature. Without his encouragement and knowledge it would have been really hard to reach the end of this long and exciting journey.

My colleagues from the School of IT also aided the development of this thesis. Particularly Lionel Ott participated actively in discussions and software development. We shared many ideas and collaborated during the duration of the whole candidature.

I would also like to thank my family and friends. They were constantly anxious about my research, motivating my everyday life. Particularly, I would like to thank my parents, who sacrificed their personal enjoyment for paying my high-school studies. Thank you very much! I hope you see the results of such an effort. Maria Jofre was also very supportive during the duration of my Ph.D. and helped proofreading the contents of this thesis.

Finally, I would like to thank Becas Chile and NICTA for the financial support over these four years.

To my parents, I love you.

- Negro

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

Notation

General Symbol Format

y	Unidimensional Variable
$\mathbf{x}$	Multidimensional Variable, Vector
A	Matrix

General

$p(A)$	Probability of event A
$p(A\|B)$	Probability of event A given event B
$f(\mathbf{x})$	Function over $\mathbf{x}$
I	Identity Matrix
K^{-1}	Matrix inversion
$K^{\top}$	Matrix transpose
$\|K\|$	Determinant of matrix K
x_i	i^{th} element of vector $\mathbf{x}$
$K_{(i,j)}$	Element of matrix K at row i and column j

Gaussian Processes

X	Training data locations
$X^{\star}$	Test data locations
$K(X', X'')$	Covariance matrix between X' and X''
$M(X')$	Mean vector evaluated at X'
$m(\mathbf{x})$	Mean function evaluated at $\mathbf{x}$
$k(\mathbf{x}', \mathbf{x}'')$	Covariance function evaluated between $\mathbf{x}'$ and $\mathbf{x}''$
σ_n	GP noise standard deviation
σ_f	Signal variance hyper-parameter
$\boldsymbol{\theta}$	GP hyper-parameter vector

$\boldsymbol{\theta}_m$	Mean function hyper-parameter vector
$\boldsymbol{\theta}_c$	Covariance function hyper-parameter vector
$f^\star$	Random variable for predicted value of $f(\mathbf{x}^\star)$
$\mathbb{E}[f^\star]$	Mean value of the predictive distribution over $f^\star$
$\mathbb{V}[f^\star]$	Variance of the predictive distribution over $f^\star$
$\mathbf{f}^\star$	Random variable for predicted values of $f(\mathbf{x}_i)$ with $\mathbf{x}_i \in X^\star$
$\mathbb{E}[\mathbf{f}^\star]$	Mean value of the predictive distribution
$\text{cov}(\mathbf{f}^\star)$	Covariance of the multivariate predictive distribution

Space-Time Gaussian Processes

$(\mathbf{s}, t)$	Spatial temporal location with spatial component $\mathbf{s}$ and temporal component t.
$m((\mathbf{s}, t))$	Mean function evaluated at $(\mathbf{s}, t)$
$k((\mathbf{s}, t)', (\mathbf{s}, t)'')$	Covariance function evaluated between $(\mathbf{s}, t)'$ and $(\mathbf{s}, t)''$
$f^\star$	Random variable for predicted value of $f((\mathbf{s}, t)^\star)$

Bayesian Optimisation

f	Objective function
h	Acquisition function
$b(f)$	Belief over f
$H(X)$	Entropy over X
Φ	Cumulative density function of the standard normal distribution
ϕ	Probability density function of the standard normal distribution
$\mathbf{x}^+$	Location of best sampled gathered so far

Abbreviations

PDF	Probability Density Function
GP	Gaussian Process
GPR	Gaussian Process Regression
GPM	Gaussian Process Mixture
STGP	Spatial-Temporal Gaussian Process
GTGP	Ground Truth Gaussian Process
BO	Bayesian Optimisation
SBO	Sequential Bayesian Optimisation
IG	Information Gain
UCB	Upper Confidence Bound
DUCB	Distance-based Upper Confidence Bound
MDP	Markov Decision Process
POMDP	Partially Observable Markov Decision Process
RRT	Rapidly exploring Random Tree

RRG	Rapidly exploring Random Graph
SLAM	Simultaneous Localisation and Mapping
EM	Environmental Monitoring
IEM	Intelligent Environmental Monitoring
UAV	Unmanned Aerial Vehicle
RMSE	Root Mean Squared Error
WRMSE	Weighted Root Mean Squared Error
NLPP	Negative Log Predictive Probability

Published Papers

Roman Marchant and Fabio Ramos. Bayesian Optimisation for Intelligent Environmental Monitoring. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2012

Roman Marchant and Fabio Ramos. Bayesian Optimisation for Informative Continuous Path Planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014

Jefferson Souza, Roman Marchant, Lionel Ott, Denis F. Wolf, and Fabio Ramos. Bayesian Optimisation for Active Perception and Smoot Navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014

Roman Marchant, Fabio Ramos, and Scott Sanner. Sequential Bayesian Optimisation for Spatial-Temporal Monitoring. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2014

Chapter 1

Introduction

1.1 Motivation

Environmental concerns have topped the agenda in the past decades. Problems such as water and air pollution, climate change and resource depletion are recognised by the scientific community as major challenges. Meanwhile, machine learning and robotics research have seen significant developments to take on problems with vast quantities of data, expanding their potential to address real-world problems. These advances in technology create great opportunities to tackle fundamental environmental issues [14]. The motivation behind this thesis is to allow autonomous robots to plan their movements in order to simultaneously learn and find interesting areas of a spatial-temporal phenomenon.

The most relevant case of study is *Environmental Monitoring* (EM). Climate change; air, water, land and acoustic pollution; solar power intensity; tidal wave behaviour among many others are highly complex processes that have a noticeable impact on human well-being. Scientists have shown a great deal of interest in understanding these phenomena, specially in the areas of health, mining, energy generation, agriculture, forestry and many more. However, deterministic differential equations representing the dynamics of these processes are difficult to devise, further considering that large quantities of data distributed over space and time are required. EM is typically

performed using sensor networks that collect measurements in pre-defined static locations. The possibility of having an autonomous robot to perform this task brings flexibility and reduces the number of necessary sensor nodes to cover the same area. Deploying an autonomous robot for the sampling procedure is both cost-effective and convenient as,

1. Robots can build statistical models of the environment and choose sampling locations intelligently;
2. Robots can move to more informative sensing locations adding flexibility over static sensor networks;
3. Robots can automate the sampling procedure reducing human supervision;
4. Robots can access areas that are dangerous for humans.

The use of autonomous robots for environment monitoring has expanded massively over the past decade. Hardware capabilities have increased noticeably, giving robots the power of traversing over a wide range of environments and monitoring several phenomena. However, a variety of problems arise when making use of autonomous moving robots for EM. The main challenges are how to learn an accurate spatial-temporal model while simultaneously choosing locations for finding the interesting areas of the phenomenon (e.g. where is the maximum level of pollution).

A plausible long-term aspiration is having a group of robots capable of monitoring the environment and maintain it suitable for humans. However, this goal is far from being completed and the first challenge to be solved is *Intelligent Environmental Monitoring* (IEM), i.e. a robot operates autonomously and decides where to gather observations from a natural phenomenon to best model it. For instance, an environmental monitoring challenge would be to supervise the quality of the water in a lake used as water reservoir for a big city. This task requires building a model of the concentration of pollutants over the whole lake based on previously sampled areas. Measurements can be the concentration of chemicals or other relevant variables such as temperature or

ambient light. Other examples are monitoring air pollution in cities, tracking ozone concentration, studying vegetation growth in problematic areas, monitor active bushfires and study coral reef depletion (Figure 1.1). The planning algorithms proposed in this thesis can be applied to any of these situations.

Information gathering techniques have also seen interesting developments. However, the problem of where and when to gather the most informative samples in an efficient manner is still an open question.

1.2 Overview on Informative Planning

Informative planning corresponds to the problem of decision making for acquiring useful data from a certain phenomenon. In the context of environmental monitoring, informative planning has been addressed using uncertainty reduction techniques to better predict the studied phenomenon. The most popular uncertainty-reduction informative planning research, conducted by Golovin and Krause [20], Krause and Guestrin [31], Singh et al. [66] and Singh and Krause [65], make use of submodular function optimisation theorems. Submodular functions are non decreasing functions that follow the property of diminishing returns where the increment over the function decreases as the size of the input set increases. However, in most applications, we are interested in finding areas of extreme values of a particular phenomenon, such as high pollutant concentration, low temperature, high humidity etc. Considering the actual value of the phenomenon removes the submodularity property, making existing methods not suitable for the required tasks.

The view of informative planning in this thesis considers a generic task to be achieved. The robot has to simultaneously learn from the environment and find areas of interest. Essentially, the chosen plan will help the robot to gather useful data and find the extreme values over the studied phenomenon. In order to find these extreme values the robot not only needs to reduce uncertainty but also focus on the actual predicted values of the phenomenon.

Figure 1.1 – Examples of environmental monitoring situations and their associated robotic platforms. Top row shows algae growth in a lake. Middle row shows coral reef depletion. Bottom row shows a bush fire. Source: *Australian Centre for Field Robotics* (ACFR)

The phenomena studied in this thesis may change with space and time. Therefore, the planning algorithms need to asses the quality and usefulness of information gathered at a specific location at a particular point in time. Basically, this creates two optimisation problems to be solved: The first and most important is to optimise over the spatial temporal phenomenon, i.e. find its maximum/ minimum over space and time. The second optimisation is conducted for every decision and corresponds to finding the optimal plan, i.e., which decision will provide the most useful information.

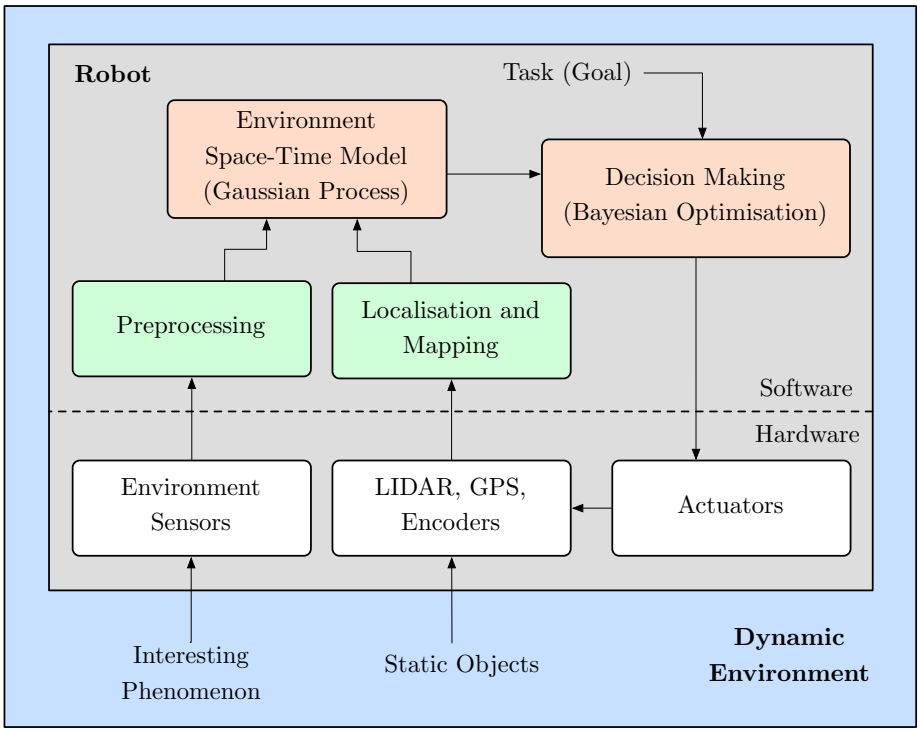


Figure 1.2 – Block Diagram

1.3 Method Overview

The proposed algorithms aim to find paths over space and time to be followed by robots in order to learn the dynamics of an unknown process and successfully complete a desired task. Figure 1.2 presents a block diagram of the proposed methodology. This diagram represents a robot immersed in a dynamic environment. The environment contains static objects that can be used to build a map and localise in it[1]. An interesting phenomenon from the environment is associated with the task the robot is assigned to, and is assumed to be known a priori, for example finding areas of higher ozone concentration or high temperature.

In Figure 1.2, the robot is represented as a grey rectangle which shows a separation between hardware and software components. In terms of hardware, we assume a fully

[1]The *Self Localisation And Mapping* (SLAM) [21] problem is considered as solved for the purposes of this thesis. The required precision in localisation depends on the scale of the environment and the speed of the robot.

functional mobile robot with the necessary actuators and sensing capabilities and do not restrict the proposed algorithms to any robotic platform in particular. The proposed planning algorithms can be used in aerial, terrestrial or aquatic situations. More important in the context of this thesis is the description of the relevant blocks in the software section of the block diagram:

Preprocessing: This first step filters unwanted data, mainly eliminating outliers and removing excessive noise from the sensing equipment. Digital filtering or anomaly detection techniques can be used to detect abnormal values.

Localisation and Mapping: Localise the robot while building a map of the surroundings.

Environment Space-Time Model: Uses the localisation to reference observations from the environment sensors and places a probability distribution over the spatial-temporal phenomenon.

Planning and Decision Making: The most important block of the system. It uses the statistics from the environment model and the known task-to-accomplish to find the optimal decision.

The two upper blocks in Figure 1.2 represent the core of the work conducted in this thesis. The first important component, Space-Time Model, builds a reliable space-time model of the environment. The second important block, Decision Making, tackles the planning component of the problem, guiding the robot towards interesting areas of the studied phenomenon.

To fully describe the method we present a real-world example: Ozone concentration monitoring. The concentration of ozone O_3 can be considered a pollutant at low altitudes. The dynamic environment represents an area of the Earth with its changing atmospheric conditions (wind, pressure, temperature, humidity) and unknown sources of contaminant at ground level. An autonomous *Unmanned Aerial Vehicle* (UAV), equipped with an ozone sensor, has the task of finding the areas where ozone pollution reaches its maximum levels. With this information, experts may be able to identify the sources of pollution and take action towards pollution reduction in the area. The

task is to find the areas of high pollution, which correspond to high concentrations of O_3. The environment model uses GPS referenced observations of ozone concentration and build an statistical model of the distribution of ozone over space and time. The path planning algorithm maximises over the possible paths the UAV can follow and finds the optimal one in order to learn the distribution and patterns of O_3 in space and time. Simultaneously, the UAV samples over areas where ozone is higher in order to find the maximum pollution levels, including when and where they occur.

1.4 Problem Statement

The problem addressed in this thesis corresponds to the following research question:

Where should a robot sense an environmental phenomenon
in order to best model its extreme values?

Properly solving this question requires addressing two main problems. The first is assuring the robot can properly model the phenomenon in the environment. In order to achieve this, it needs to identify patterns over space and time which allow it to generalise over the entire domain. The challenge for this problem is learning the model of the phenomenon using limited, noisy and sparse data.

The second part of the problem, which is the core of this thesis, is deciding where the robot should go next. These decisions have to be aligned with the goal of the monitoring task (pollution search, temperature rise surveillance, etc). Each decision corresponds to a plan, geometrically understood as discrete waypoints or even a continuous path representation. Since the environmental phenomenon is initially unknown, the complexity of the planning algorithms is high as it involves solving the exploration-exploitation trade off problem. When should the robot plan towards exploration? When should it plan towards completing the task?

1.5 Contributions

The contributions of this thesis are the following:

1. **Formulation of *Bayesian Optimisation* (BO) to planning in robotics.** BO is a widely known technique for optimising unknown, noisy and costly to evaluate functions. In this thesis, we apply BO to solve a path planning problem in the robotics context. Essentially, this extends the use of BO for using a mobile robot to optimise functions which are the realisation of an environmental phenomenon.

2. **A new family of acquisition functions for BO which consider side state.** Because in the existing BO theory there is no real agent conducting the sampling, there is no need to consider its state for evaluating candidate sampling locations. In fact, existing acquisition functions, further discussed in Chapter 2, do not take into account previously sampled locations or the sequence of them. However, since we introduced an autonomous robot as the sampling agent, we present a new family of acquisition functions, which we call *state-aware* acquisition functions. This awareness allows an acquisition function to favour sampling locations closer to the current location of the robot, favouring safer paths to avoid collisions or reduce energy consumption.

3. **Generalisation of BO for optimisation over continuous paths.** The decisions for the plain BO algorithm correspond to discrete locations over the input space. In the robotics context, it makes sense to optimise along continuous paths instead of discrete locations. Continuous paths are characterised by a set of parameters. At each step, the acquisition function is no longer evaluated with respect to a discrete location, but the value corresponds to the integral along the path candidate. The optimal set of path parameters is then found for each iteration of the BO algorithm.

4. **Layered BO for planning in spatial-temporal monitoring.** Usually, the optimisation of the acquisition function is conducted using gradient-based, local

search optimisers. However, for planning along continuous paths the search space is no longer a location in the input space of the function, but becomes a search in the parameter space of a path. Depending on the parametrisation and dynamics of the spatial-temporal phenomenon, optimising over the acquisition function can become increasingly difficult. In this thesis, we present a layered BO approach, where the acquisition function is maximised with another BO layer. This means that a *Gaussian Process* (GP) prior is also placed over the acquisition function itself and samples from it are guided using the second-layer acquisition function.

5. **Evaluation of planning algorithms in large scale experiments and on real robots.** The algorithms proposed in this thesis have been evaluated using large scale datasets and real robots. Ozone pollution is characterised and monitored using a simulated *Unmanned Air Vehicle* (UAV). Light intensity is used as an analogue to pollution to evaluate the planning algorithms using an autonomous mobile robot. The different techniques for decision-making are quantitatively and qualitatively compared under control scenarios where ground-truth is available.

6. ***Sequential Bayesian Optimisation* (SBO).** Plain BO can be readily applied to scenarios where the objective function does not vary in time and sampling locations can be chosen freely within the input domain. For the application studied in this thesis, functions vary with time and the sampling platform determines the reachable space for gathering the next samples. Combining these two issues creates an imperative need for extending the common BO algorithm to a sequential setting. SBO considers a reward for each decision in a sequence and finds the best decision in terms of the cumulative reward for the whole sequence.

7. **Develop a unifying formulation for *Partially Observed Markov Decision Process* (POMDP) and SBO.** POMDPs are a framework for non myopic decision making under uncertainty. We formulate SBO as a POMDP

model, assigning the state of the POMDP to a tuple which groups the goal function and the pose of the robot. The contribution is on identifying the components that make solving the SBO problem the same as solving its POMDP analog.

8. **A SBO solution using the POMDP formulation.** By combining *Monte Carlo Tree Search* (MCTS) and *Upper Confidence Bound for Trees* (UCT), we solve the POMDP analog of SBO for the case of continuous state and observations. This solution is convenient for systems where robots are monitoring a dynamic process, since it can be computed using limited resources and provides a multiple-step lookahead solution.

1.6 Thesis Outline

An overview of the following chapters presented in this thesis is presented below:

Chapter 2: Theoretical Background
Reviews the concepts behind regression and optimisation of noisy functions. It focuses on *Gaussian Processes* (GPs) and *Bayesian Optimisation* (BO), which are the two main theoretical components used for developing the proposed algorithms in this thesis.

Chapter 3: Planning over Waypoints
This chapter presents a planning algorithm for monitoring spatial temporal phenomena. It first shows how to build an statistical model of an space-time function using noisy samples. Then it proposes a planning algorithm based on an adapted version of BO to choose sampling locations, where the next-best location to sample corresponds to a waypoint in the environment. A new family of acquisition functions, which is state-aware, is presented. This new kind of acquisition functions is particularly interesting for tackling real-world applications of BO, where the sampling is performed by a real robotic agent. Experiments demonstrate the advantages of the planning

algorithms over naive sampling techniques such as random sampling and information gain.

Chapter 4: Planning over Continuous Paths

A planning algorithm for optimal sampling over continuous paths is presented in Chapter 4. This chapter begins by presenting spatial-temporal GPs and the advantages of using separable covariance functions for spatial and temporal components. It generalises the discrete-planning algorithm proposed in the previous chapter to optimise over continuous paths. It also proposes a layered BO approach for solving the continuous planning problem. Experiments demonstrate the benefits of the approach, comparing continuous and discrete sampling strategies.

Chapter 5: Sequential Continuous Planning

A solution for the optimal sampling problem that considers a sequence of decisions is developed in Chapter 5. Firstly, it presents the formulation of *Sequential Bayesian Optimisation* (SBO), which takes into account the whole sequence of decisions in order to select the next sampling locations. The non-myopic version of BO, SBO, is then formulated under a *Partially Observed Markov Decision Process* (POMDP) framework. The POMDP analogue of SBO is then solved using an online POMDP solver that can accommodate continuous observations.

Chapter 6: Conclusions and Future Work

Conclusion of this thesis is presented in Chapter 6. The first part of this chapter draws some conclusions and summarises the contributions from previous chapters. It then provides directions for future work that can naturally extend the planning algorithms proposed in this thesis.

Chapter 2

Theoretical Background

2.1 Introduction

This chapter presents the necessary background in which the planning algorithms proposed in this thesis are built on. As stated in the previous chapter, the goal is to maximise an unknown objective function that changes with space and time. The main application addressed in this thesis is oriented towards optimal sensing for environmental monitoring using an autonomous robot where the realisation of the objective function corresponds to a phenomenon in the environment.

The proposed solution for autonomous optimal sensing of spatial-temporal phenomena consists of two parts. Firstly, the robot needs to model the initially unknown environmental phenomenon using noisy samples. The challenge is to learn spatial and temporal patterns that allow the robot to generalise over the un-sampled domain and predict the state of the environment in the future. In order to tackle this difficult task, Section 2.2 presents regression techniques, particularly focusing on *Gaussian Process* (GP) regression, which is a popular statistical method used in machine learning for pattern recognition.

Secondly, the robot needs to take decisions in order to learn and find the extreme values in the environmental phenomenon. These decisions correspond to a plan, es-

sentially answering the question of where to sample next. The planning algorithms proposed in this thesis are based on *Bayesian Optimisation* (BO), which is an optimisation routine for unknown and costly to evaluate functions. Section 2.3 describes the BO algorithm including details on how it chooses sampling locations.

Finally, we consider the case of long-term planning, where decisions are taken based on a sequence of decisions. Section 2.4 presents *Partially Observed Markov Decision Processes* (POMDPs), a non-myopic decision making framework. In Chapter 5 we use POMDPs combined with BO and GPs to generate multi-step lookahead plans.

2.2 Gaussian Process Regression

Regression corresponds to an area of supervised learning where the goal is to generalise an unknown function and predict its value over a continuous domain. Although there are several ways of achieving regression, a *Gaussian Process* (GP) is an elegant solution for achieving regression from noisy observations as it is a powerful, non-parametric tool for non-linear regression. They are particularly useful for the planning algorithms developed in this thesis since they provide a posterior *Probability Density Function* (PDF) over the unknown function values. GPs have been widely used for modelling spatially and temporally correlated data. Consequently, they are extensively used in all the chapters of this thesis. This section reviews GP theory for regression. For deeper insights and a more extensive theoretical description, the reader can refer to Rasmussen and Williams [58].

To model a function f, a GP places a multivariate Gaussian distribution over the space of functions mapping the input to the output. The model is completely defined by a mean function $m(\mathbf{x}|\boldsymbol{\theta}_m)$ and a covariance function $k(\mathbf{x}, \mathbf{x}'|\boldsymbol{\theta}_c)$, i.e.,

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}|\boldsymbol{\theta}_m), k(\mathbf{x}, \mathbf{x}'|\boldsymbol{\theta}_c)) \,, \tag{2.1}$$

where $\boldsymbol{\theta}_m$ and $\boldsymbol{\theta}_c$ are the mean and covariance function hyper-parameters respectively.

Using supervised learning to build a GP model involves gathering a set of observations $S = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$ from f, where $\mathbf{x}_i \in \mathbb{R}^D$ are the N sampling locations (inputs) in a D dimensional space, and $y_i \in \mathbb{R}$ are the corresponding noise outputs. These noisy samples are used to tune the hyper-parameter set $\boldsymbol{\theta} = \{\boldsymbol{\theta}_m, \boldsymbol{\theta}_c\}$ to match the behaviour of f (detailed on Section 2.2.2). The learnt GP model can be used to predict a Gaussian distribution over $f(\mathbf{x}^\star)$ at any new sampling location $\mathbf{x}^\star$, which is later referred as $f^\star$.

Samples from the real process are assumed to be noisy, i.e., $y = f(\mathbf{x}) + \epsilon$, where ϵ follows a normal distribution $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. The joint distribution of the observed target values $\mathbf{y} = \{y_i\}_{i=1}^{N}$, at locations X, and the predicted values $\mathbf{f}^\star = \{f_i^\star\}_{i=1}^{M}$, at $X^\star$, is given by

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}^\star \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X) + \sigma_n^2 I & K(X,X^\star) \\ K(X^\star,X) & K(X^\star,X^\star) \end{bmatrix}\right), \tag{2.2}$$

where $K(\cdot,\cdot)$ is the covariance matrix, defined in a component-wise fashion as

$$K(X', X'')_{(i,j)} = k(x_i', x_j'') \;, \qquad \text{with} \quad x_i' \in X' \;\text{ and }\; x_j'' \in X'' \;. \tag{2.3}$$

X, $X^\star$ are the groups of training and testing locations respectively. Given the training locations, the predictive distribution over the values at the test locations is given by

$$\mathbf{f}^\star | X, \mathbf{y}, X^\star \sim \mathcal{N}(\bar{\mathbf{f}^\star}, \mathrm{cov}(\mathbf{f}^\star)) \;, \tag{2.4}$$

with,

$$\bar{\mathbf{f}^\star} = \mathbb{E}[\mathbf{f}^\star | X, \mathbf{y}, X^\star] = K(X^\star, X)\left[K(X,X) + \sigma_n^2 I\right]^{-1}(\mathbf{y} - M(X)) \tag{2.5}$$

$$\mathrm{cov}(\mathbf{f}^\star) = K(X^\star, X^\star) - K(X^\star, X)\left[K(X,X) + \sigma_n^2 I\right]^{-1} K(X, X^\star) \;, \tag{2.6}$$

where $M(\cdot)$ is a mean vector, which contains an evaluation of the mean function in each element: $M(X)_{(i)} = m(x_i)$ with $x_i \in X$.

For the specific case when there is only one query location $\mathbf{x}^\star$, the predictive distri-

bution over $f^\star$ is given by:

$$f^\star|X, \mathbf{y}, \mathbf{x}^\star \sim \mathcal{N}(\mathbb{E}[f^\star], \mathbb{V}[f^\star]) \,, \tag{2.7}$$

with,

$$\mathbb{E}[f^\star] = K(\mathbf{x}^\star, X)K_X^{-1}(\mathbf{y} - M(X)) \tag{2.8}$$

$$\mathbb{V}[f^\star] = k(\mathbf{x}^\star, \mathbf{x}^\star) - K(\mathbf{x}^\star, X)K_X^{-1}K(X, \mathbf{x}^\star) \;\; , \tag{2.9}$$

where $K_X = K(X, X) + \sigma_n^2 I$.

The output PDF, which represents the predictions over $\mathbf{f}^\star$, depends strongly on the hyper-parameter set $\boldsymbol{\theta}$. The mean and covariance functions both affect the predicted value of f at the un-sampled locations. Generally, it is assumed that the data has a zero mean, i.e., $m(x) = 0$. This is a common choice, unless a more complex mean function, such as a polynomial, can represent the mean of the function more accurately. There are several families of covariance functions, each with their corresponding set of hyper-parameters. The covariance functions relevant for this thesis are listed in the following section.

2.2.1 Covariance Functions

The choice of the covariance function is relevant as it influences the GP's behaviour directly. Covariance functions are also known as Mercer kernels and encode the degree of correlation between two locations in the input space of a function. Given two inputs $\mathbf{x}'$ and $\mathbf{x}''$, the covariance function evaluated over both inputs is $k(\mathbf{x}', \mathbf{x}'')$ and represents the covariance between the values of f at those locations.

Covariance functions can be mainly classified into stationary and non-stationary. Stationary covariance functions may be expressed only in terms of the distance, $\mathbf{d}$, between the inputs. They are therefore invariant to the input location. To achieve a

Table 2.1 – Typical Covariance Functions.

Name	**Expression** $k(\mathbf{x}', \mathbf{x}'')$	**Hyper-Parameters**
Linear	$\sigma_f^2(\sigma_0^2 + \mathbf{x}'^T L \mathbf{x}'')$	$\boldsymbol{\theta}_c = \{\sigma_f, \sigma_0, L\}$
Matern 3	$\sigma_f^2 \left(1 + \sqrt{3\mathbf{d}}\right) \exp\left(-\sqrt{3\mathbf{d}}\right)$	$\boldsymbol{\theta}_c = \{\sigma_f, L\}$
Matern 5	$\sigma_f^2 \left(1 + \sqrt{5\mathbf{d}} + \frac{5\mathbf{d}}{3}\right) \exp\left(-\sqrt{5\mathbf{d}}\right)$	$\boldsymbol{\theta}_c = \{\sigma_f, L\}$
Polynomial	$\sigma_f^2(\sigma_0^2 + \mathbf{x}'^T L \mathbf{x}'')^p$	$\boldsymbol{\theta}_c = \{\sigma_f, \sigma_0, L, p\}$
Rational Quadratic	$\sigma_f^2 \left(1 + \frac{\mathbf{d}}{2\alpha}\right)^{-\alpha}$	$\boldsymbol{\theta}_c = \{\sigma_f, L, \alpha\}$
Squared Exponential	$\sigma_f^2 \exp\left(-\frac{\mathbf{d}}{2}\right)$	$\boldsymbol{\theta}_c = \{\sigma_f, L\}$
Periodic Exponential	$\sigma_f^2 \exp\left(-\frac{2\sin^2\left(\pi T \sqrt{\mathbf{d}}\right)}{\rho^2}\right)$	$\boldsymbol{\theta}_c = \{\sigma_f, L, T, \rho\}$

generic representation of distance in each dimension, $\mathbf{d}$ is defined as:

$$\mathbf{d} = (\mathbf{x}' - \mathbf{x}'')^\top L (\mathbf{x}' - \mathbf{x}'') \tag{2.10}$$

where L is a diagonal matrix of size D and contains a length-scale parameter on each element of its diagonal,

$$L_{(i,i)} = \frac{1}{\ell_i^2} \ . \tag{2.11}$$

There are several covariance functions studied in the literature such as Squared Exponential, Matérn, Neural Networks and many more. A list of the covariance functions relevant to this thesis is presented in Table 2.1. Each row of Table 2.1 contains the name of the covariance function, its mathematical expression and the set of its hyper-parameters $\boldsymbol{\theta}_c$.

2.2.2 Model Selection

Model selection corresponds to the problem of adapting the structure of the GP components: selection of a mean and covariance function and tuning of their hyper-parameters.

The full set of hyper-parameters of a GP is given by $\boldsymbol{\theta} = \{\boldsymbol{\theta}_m, \boldsymbol{\theta}_c, \sigma_n\}$. Training a GP corresponds to the process of finding the optimal set of parameters $\boldsymbol{\theta}^*$ that best represent f. One of the ways for finding $\boldsymbol{\theta}^*$ is by maximising a goal function (GF) which depends on the training data (samples from f) and the hyper-parameter set $\boldsymbol{\theta}$.

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \ \mathrm{GF}(\mathbf{y}, X, \boldsymbol{\theta}) \,, \tag{2.12}$$

The most popular goal function is the the log marginal likelihood (LML),

$$\mathrm{LML}(\mathbf{y}, X, \boldsymbol{\theta}) = -\frac{1}{2}\left(\mathbf{y} - M(X)\right)^T K_X^{-1}\left(\mathbf{y} - M(X)\right) - \frac{1}{2}\log|K_X| - \frac{n}{2}\log 2\pi \,, \tag{2.13}$$

where $K_X = K(X, X) + \sigma_n I$.

This procedure reflects the actual learning in the GP framework. The optimisation of the hyper-parameters adapts the structure of the GP to achieve a valid representation of the sampled data. Other techniques for training a GP include fully bayesian approaches, where a prior distribution is placed over the hyper-parameters and inference is achieved using *Markov Chain Monte Carlo* (MCMC).

2.2.3 Addressing Large Datasets

GP regression can become prohibitive for large datasets (more than 10,000 observed values). The most expensive operation is the inversion of the covariance matrix which needs to be calculated for Equation 2.5, Equation 2.6 and Equation 2.13. Matrix inversion scales as $\mathcal{O}(N^3)$, where N is the number of data-points.

The simplest approach corresponds to selecting a subset of the data-points. This selection can be made randomly or using entropy reduction techniques as shown by

Lawrence [36]. If the new set is of size $Q \ll N$, then the cost for the covariance matrix inversion would be $\mathcal{O}(Q^3)$. Even though there is an efficient reduction of computational costs, the variance computation which reflects the uncertainty, is far from the exact one. For example, uncertainty might seem high at a certain location because no data is found close to it. However, the real uncertainty should be much lower since the whole dataset contains samples which are close to that location and have been discarded.

Quiñonero-Candela and Rasmussen [57] present a review of GP approximation techniques, including *Subset of Regressors* (SoR). These techniques make use of inducing points, which are a small set of locations over the input space. The complete dataset is then projected into the inducing points, reducing the order to $\mathcal{O}(Q^2N)$ with $Q \ll N$ being the number of inducing points.

Alternatively, work by Vasudevan et al. [75] and Shen et al. [63] have shown that nearest neighbourhood selection for training can result in tractable regression. Here, they make use of KD-Trees to organise data and query the GP only with the data which is closest to the query location. Recently, Hensman et al. [23] show how stochastic variational inference can be used to approximate GP inference for millions of points.

2.3 Bayesian Optimisation

Bayesian Optimisation (BO) is an optimisation algorithm used for finding the extreme of unknown and costly-to-evaluate functions. A detailed theoretical description can be found in [3, 25, 40, 53, 61].

The goal of Bayesian optimisation is to find the maximum of an unknown function f,

$$f(\mathbf{x}) \, : \, \mathbb{R}^D \to \mathbb{R} \, , \tag{2.14}$$

i.e., to find

$$\mathbf{x}^\star = \underset{\mathbf{x} \in \mathbb{R}^D}{\arg\max}\, f(\mathbf{x})\,. \tag{2.15}$$

If a particular problem requires finding the minimum, BO can maximise the additive inverse of the original function. The use of BO is particularly suitable when f is a black-box function with no gradient information and when evaluations from it are resource consuming, hence the need for evaluating f as few times as possible. Even if f is non-convex and has multiple local optima, BO is likely to find the true global optimum. The only assumption over f is that it is Lipschitz-continuous, i.e.,

$$\exists\ C \in \mathbb{R}\ \mid\ \|f(\mathbf{x}_1) - f(\mathbf{x}_2)\| \leq C\|\mathbf{x}_1 - \mathbf{x}_2\|\ \ \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^D. \tag{2.16}$$

BO is an iterative algorithm for optimisation that makes use of Bayes' theorem at each iteration i to combine prior belief over f, $b_i(f)$, with acquired data, $X_{1:i} = \{\mathbf{x}_{1:i}, f(\mathbf{x}_{1:i})\}$, to produce a new estimation of f, called updated belief $b_{i+1}(f)$,

$$b_{i+1}(f|X) \propto p(X_{1:i}|f)b_i(f)\ . \tag{2.17}$$

In this thesis the belief over f is represented by a GP. The expected value of f has an associated variance which captures the level of uncertainty in the prediction, as described by Equation 2.4.

The key for the success of BO is the smart selection of evaluation points. At each iteration the next sampling location is the one that maximises a *expected utility*. The function that encodes the utility of sampling at one location over another is called *acquisition function*, h, and its general form will be discussed in Section 2.3.1. In brief, BO uses a quantitative measure given by the acquisition function for making informed decisions and choose the most promessing locations to sample from the unknown function over its domain.

The BO algorithm pseudo code is shown in Algorithm 1. Line 2 is the optimisation of the acquisition function. It can be noted that the problem of maximising f has now been moved to finding the maximum of h in each iteration, another non-convex

Algorithm 1 Bayesian Optimisation

Inputs: f, h, GP-prior
Outputs: $\mathbf{x}_{\max}$, $y_{\max}$

```
while i < i_max do
    x* ← arg max_x h(x)
    Gather sample y* = f(x*)
    Augment X and y with {x*, y*}
    if y* > y_max then
        y_max ← y*
        x_max ← x*
    end if
    i ← i + 1
    Update GP posterior
end while
```

optimisation. However, considering an appropriate acquisition function, optimising in the domain of the acquisition function is easier than the original problem. In fact, optimising h is a much simpler problem since it has derivative information and is fast to evaluate. This *inner optimisation* procedure can be implemented with any optimisation technique, most commonly gradient based optimisers.

A one-dimensional example of the BO algorithm using a dummy acquisition function is illustrated in Figure 2.1. It is possible to see how the mean of the GP regression model quickly converges to the unknown function and the variance is reduced near sampling locations. The shape of the acquisition function changes after each iteration[1]. Initially when the variance is high the sampled locations are automatically chosen by the algorithm to get an initial approximation of f, spreading samples across the domain. In the long term, samples concentrate near higher values of f, eventually getting a better estimate of the locations of the optimum.

2.3.1 Acquisition Functions for BO

Acquisition functions are a fundamental part of the BO algorithm since they guide the search for the optimum in every iteration. They should reflect the expected utility of

[1] A video showing all iterations is available at `http://www.it.usyd.edu.au/~rmar5258/IROS2012/1d_example.html`

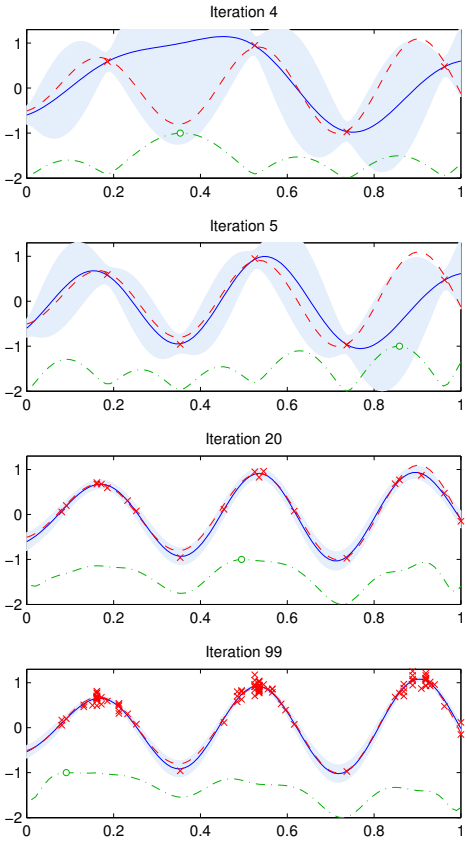


Figure 2.1 – One dimension example of active sampling based on Bayesian optimisation. The continuous blue line and shade represent the GP mean an variance respectively. The dashed red line is the unknown function, and noisy samples from this function are shown as red crosses. The dash-dot green line is the acquisition function, with a circular mark at its maximum. This function is scaled and with an offset for visualisation purposes.

sampling at each location in the domain. Therefore, higher values of the acquisition function should correspond to a greater probability of finding higher values of f. The combination of an appropriate acquisition function and a GP allows an elegant trade off between exploration and exploitation while searching for the optimum. To simplify notation in this section, the expressions for the expected value and standard deviation for $f(\mathbf{x})$ are defined as:

$$\mu(\mathbf{x}) \triangleq \mathbb{E}[f(\mathbf{x})] \tag{2.18}$$

$$\sigma(\mathbf{x}) \triangleq \sqrt{\mathbb{V}[f(\mathbf{x})]}\,, \tag{2.19}$$

where $\mathbb{E}[f(\mathbf{x})]$ and $\mathbb{V}[f(\mathbf{x})]$ are defined in Equations 2.8 and 2.9 respectively. The most popular acquisition functions are Information Gain, Probability of Improvement, Expected Improvement and Upper Confidence Bound. These are defined as follows,

Information Gain

An often used approach is to sample in areas where there is not a good understanding of the function by maximising the *Information Gain* (IG). The IG is defined as the differential entropy when adding a new point to the training set,

$$\mathrm{IG}(\mathbf{x}) \triangleq \arg\max_{\mathbf{x}} \left(H[X] - H[X \cup \mathbf{x}] \right), \tag{2.20}$$

where $H[X]$ is the entropy over the entire domain using X as training set. IG is a monotonic function that has its maximum located where the variance is highest. The IG approach would then correspond to an acquisition function that is equivalent to the variance of the prediction over the domain. This acquisition function is not suitable for most optimisation problems as it corresponds to a purely explorative approach.

Probability of Improvement

Studied by Kushner [32] and Jones [29], this acquisition function calculates the probability of improvement over the best value of the goal function, $f(\mathbf{x}^+)$, found in the current sample set X, where

$$\mathbf{x}^+ = \arg\max_{\mathbf{x}_i \in X} f(\mathbf{x}_i) \ . \tag{2.21}$$

Mathematically,

$$\mathrm{PI}(\mathbf{x}) \triangleq p(f(\mathbf{x}) \leq f(\mathbf{x}^+) + \xi) \tag{2.22}$$

$$= \Phi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+) - \xi}{\sigma(\mathbf{x})} \right) \tag{2.23}$$

where Φ is the cumulative Gaussian distribution and $\xi \leq 0$ is an exploration-exploitation trade-off parameter. Larger values of ξ determine a more explorative behaviour of the resulting algorithm, whereas $\xi \to 0$ results in pure exploitation.

Expected Improvement

The *Expected Improvement* (EI) [29, 50] is an acquisition function that measures the amount of improvement when choosing a new location. The improvement function

$$\mathrm{I}(\mathbf{x}) \triangleq \max\left\{0, f(\mathbf{x}) - f(\mathbf{x}^+)\right\} \tag{2.24}$$

is always positive, or zero when no improvement is obtained by evaluating at $\mathbf{x}$ i.e. $f(\mathbf{x}) \leq f(\mathbf{x}^+)$.

Assuming a GP prior for f, the expected improvement is given by

$$\begin{aligned}
\mathrm{EI}(\mathbf{x}) &\triangleq \mathbb{E}\left[\mathrm{I}(\mathbf{x})\right] \\
&= \mathbb{E}\left[\max\left\{0, f(\mathbf{x}) - f(\mathbf{x}^+)\right\}\right] \\
&= \int_{f(x^+)}^{\infty} \left(f(\mathbf{x}) - f(\mathbf{x}^+)\right) \phi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x})}{\sigma(\mathbf{x})}\right) df(\mathbf{x}) \ ,
\end{aligned} \tag{2.25}$$

where ϕ denotes the PDF of the standard normal distribution. The first term in the integrand of equation 2.25 is the amount of improvement and the second term represents the probability of that improvement. After calculating the integral of equation 2.25 using integration by parts, the expected improvement can be defined as

$$\mathrm{EI}(\mathbf{x}) = \sigma(\mathbf{x})\left[Z\Phi(Z) + \phi(Z)\right] \ , \tag{2.26}$$

where

$$Z = \frac{\mu(\mathbf{x}) - f(\mathbf{x}^+)}{\sigma(\mathbf{x})} \ , \tag{2.27}$$

and Φ denotes the *Cumulative Density Function* (CDF) of the standard normal distribution.

Similarly to the Probability of Improvement acquisition function, the EI may tend to

over exploit, by picking locations near sampled points with high value. According to Lizotte [40], this can be solved by introducing the factor ξ, resulting in,

$$Z = \frac{\mu(\mathbf{x}) - f(\mathbf{x}^+) - \xi}{\sigma(\mathbf{x})} \ . \tag{2.28}$$

Upper Confidence Bound

The UCB acquisition function, studied by Cox and John [9], considers the mean and variance at a particular location. Evaluated at $\mathbf{x}$ it takes the form:

$$\text{UCB}(\mathbf{x}) \triangleq \mu(\mathbf{x}) + \kappa \cdot \sigma(\mathbf{x}) \ . \tag{2.29}$$

The parameter κ is related to the exploration-exploitation trade off. While high values of κ lead to an explorative behaviour of the algorithm, lower values of κ favour exploitation near known sampled locations.

It is relevant to mention that none of the previously described acquisition functions consider the distance between the sample locations, which is a fundamental component of the proposed methods in the following chapter.

2.4 Partially Observable Markov Decision Processes (POMDPs)

In the following chapters, a solution for a complex BO formulation is solved using POMDPs. Therefore, the goal of this section is presenting the theory behind POMDPs. Firstly, this section defines a *Markov Decision Process* (MDP), which is the building block for POMDPs.

2.4.1 *Markov Decision Process* (MDP)

The MDP framework is a popular technique for planning and decision making. It uses a basic model from the world which is represented by the tuple $\langle S, A, T, R \rangle$, where:

- S : Set of states $\{s_1, s_2, \ldots, s_n\}$.
- A : Set of actions $\{a_1, a_2, \ldots, a_n\}$.
- $T : S \times A \times S \rightarrow [0, 1]$ is a transition function that represents the probability of transition between states s and s' when executing action a, i.e. $T(s, a, s') = p(s'|s, a)$.
- $R : S \times A \rightarrow \mathbb{R}$ is a reward function that encodes the reward of executing action a on state s, i.e. $R(s, a)$.

Decisions are encoded by the action space A. For every time-step, the dynamics of a MDP is given by the state of world s and selected action a. As a consequence, the agent receives a reward $R(s, a)$ and the world state changes to s'. A MDP is said to be Markovian since the resulting state only depends on the current state and the executed action.

A policy, π, that maps from the state space to the action space:

$$\pi : S \rightarrow A \ , \tag{2.30}$$

is used to determine what action a to execute given a particular state s. A policy can deliver different levels of reward over time. The objective is to take the best decisions, i.e. the ones that accumulate most reward. The optimal policy, $\pi^\star$, can be found by solving

$$\pi^\star = \arg\max_{\pi} \ E\left[\sum_{t=0}^{n} \gamma^t R(s, \pi(s))\right] \ , \tag{2.31}$$

where n is the planning horizon (which can be infinite) and $\gamma \in [0, 1)$ is the discount factor that encodes the present value of future reward. The state s evolves according to the transition function T.

There are several algorithms for finding the optimal policy: Value Iteration, Policy Iteration and Linear Programming. These are not particularly relevant for the contents of this chapter and further details on how to find an optimal policy for an MDP can be found in Sutton and Barto [72].

2.4.2 Partial Observability

POMDPs are a unified framework for sequential decision making under uncertainty when the state is not directly observable and the agent only has access to noisy observations. Observations depend on the state s, therefore sequences of observations can be used to infer the state. A discrete-state and discrete-action POMDP is defined in terms of a MDP (Section 2.4.1) and includes a new observation set and observation function. A POMDP is fully represented by the tuple $\langle S, A, T, R, Z, O \rangle$, where S, A, T and R are the same as for MDPs, and Z, O are given by:

- Z : Finite set of observations $\{z_1, z_2, \ldots, z_n\}$.
- $O : Z \times A \times S \to [0, 1]$ is a observation function that represents the probability of observing z if action a is executed with resulting state s, i.e. $O(z, a, s) = p(z|a, s)$.

As was mentioned earlier, the main assumption of POMDPs is that the state is only partially observable. To quantify the uncertainty in observing the true state, a probability function over the state space is defined as *belief*, $b(s)$. For discrete state spaces the belief $b(s)$ represents the probability of the state being s and satisfies

$$\forall s \in S, \quad b(s) \in [0, 1] \quad \wedge \quad \sum_{s \in S} b(s) = 1 \ . \tag{2.32}$$

The belief can be updated for every action-observation pair:

$$b_t^{a,z}(s') = \frac{O(z, a, s') \sum_{s \in S} T(s, a, s') b_{t-1}(s)}{p(z|b, a)} \ , \tag{2.33}$$

where

$$p(z|b,a) = \sum_{s\in S} b(s) \sum_{s'\in S} T(s,a,s')O(z,a,s') \ . \tag{2.34}$$

The optimal policy $\pi^\star$ for a POMDP, which maps from belief states to actions, is given by

$$\pi^\star = \arg\max_{\pi\in\Pi} E\left[\sum_{t=0}^{n} \gamma^t \sum_{s\in S} b_t(s) \sum_{a\in A} R(s,a)\pi(b_t,a)|b_0\right] \ , \tag{2.35}$$

where $\pi(b_t, a)$ represents the probability of executing a in belief b_t according to π and b_0 represents a probability distribution over the initial state, which is a uniform distribution in the cases where no prior knowledge is available.

A discrete-state POMDP can be formulated as a belief-space MDP with continuous state space, where the state is the belief. The reward becomes R_B and depends on the belief b:

$$R_B(b,a) = \sum_{s\in S} b(s)R(s,a) \tag{2.36}$$

and the transition function over beliefs, τ:

$$\tau(b,a,b') = \sum_{z\in Z} p(z|b,a)\mathbb{1}(b' = b^{a,z}) \ . \tag{2.37}$$

Note that τ differs from T, since τ is defined over the belief space. Using this analogy, the optimal policy for a discrete-state POMDP can be found using MDP solvers. The main limitation here is the sum over the observation set, which for this thesis can be considerably large or even continuous.

Continuous State Spaces

In this thesis, the state space is continuous, which brings considerable analytical complexity to the previously described theory. The continuous version of the belief update from Equation 2.33 can be reformulated as:

$$b_t(s') \propto O(z,a,s') \int b_{t-1}(s)T(s,a,s')ds. \tag{2.38}$$

Similarly, all expressions that include sums over states can be replaced with integrals.

For the case of POMDPs with continuous state, the belief space is continuous but infinite dimensional. Although existing work by Porta et al. [56] and Deisenroth et al. [12] have presented solutions for finding the optimal policy, they are limited to very simple problems.

Online vs Offline POMDP Solution

Finding the optimal policy corresponds to the full solution of a POMDP, which is also called offline solution, since it is precomputed and can determine which is the best action given any belief state. For real problems such as those addressed in this thesis, finding the exact solution can be prohibitively slow and consume excessive computer resources as shown by Ross et al. [59].

For the large POMDPs addressed in this thesis, it makes sense to find an online solution. An online solution is a better alternative since it finds a local policy for the current belief state. The search space becomes smaller since the optimal action only needs to be determined for the reachable belief states. Section 5.4 will introduce the online technique used in this thesis.

2.5 Summary

This chapter provided the necessary background used as foundation for developing the planning algorithms in this thesis. GP regression is a complex regression technique based on bayesian inference, which is used predict the value of an unknown function using noisy values from it. By learning the model hyper-parameters, a GP provides an accurate probabilistic representation of any function. GPs will be used in the following chapters to model real world environmental phenomena.

Bayesian Optimisation (BO) is used for finding the maximum of unknown functions. By maintaining a probabilistic model of the goal function, the method can quantify

uncertainty and deal with the exploration exploitation trade-off in a principle manner. BO evaluates an acquisition function at each iteration to guide the search of the optimum.

The theory of decision making under uncertainty, framed as a POMDP, has been presented. A POMDP defines world components and finds the optimal action to maximise the cumulative reward over a set of actions. POMDPs will be used in Chapter 5 to achieve non-myopic trajectory planning.

Chapter 3

Planning over Waypoints

3.1 Introduction

This chapter presents a waypoint approach towards planning for maximising an objective which is initially unknown. It shows how *Bayesian Optimisation* (BO) can be used as an optimisation routine for planning. The BO algorithm selects which locations need to be sampled at a specific time to achieve an optimal sampling of the phenomenon to be monitored. Part of this chapter has been previously published at IROS[1].

Over the last decade, a vast amount of research effort has been dedicated to field robotics. In particular, environmental monitoring using mobile robots is gaining popularity among a wide range of applications [14, 67, 69]. The motivation behind the proposed algorithms is to allow autonomous robots plan their movements in order to simultaneously learn and monitor efficiently a spatial-temporal phenomenon. Solving this problem properly requires creating a reliable spatial-temporal model of the phenomenon and finding the sample locations that maximise the robot's understanding of it.

[1]Roman Marchant and Fabio Ramos. Bayesian Optimisation for Intelligent Environmental Monitoring. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2012

The first part of this chapter shows how to create a statistical representation of a phenomenon in the environment. The second part presents a decision making algorithm that determines where to acquire next measurements in order to improve the current model.

The outcome of the planning algorithm is a series of ordered waypoints that represent a path in a discrete manner. The main contributions presented in this chapter are the extension of the BO framework to planning in the mobile robotics context and presenting a new family of acquisition functions for the BO algorithm that considers the distance between sampling locations. This last innovation is beneficial for robotic systems as it reflects the cost of moving in the environment, mainly determined by power consumption. The proposed methodology is tested in simulation and in a real environment. Compared to existing strategies, the proposed approach exhibits slightly better accuracy in terms of *Root Mean Squared Error* (RMSE) over the predicted values of the phenomena and considerably reduces the total distance travelled by the robot.

The remainder of this chapter is structured as follows. Section 3.2 details the theory for spatial-temporal models. Section 3.3 explores the use BO for determining sampling locations. Experimental setup, results and analysis are shown in Section 3.4. Finally, Section 3.6 summarises the contents of the chapter.

3.2 Spatial Temporal Modelling

Spatial-temporal modelling corresponds to the problem of inferring the state of the environment at a spatial location for a specific point in time given previously collected data.

If the spatial temporal phenomenon in the environment is considered as a function f, the full representation of f is given by

$$f(\mathbf{s}, t) \, : \, \mathbb{R}^D \times \mathbb{R} \rightarrow \mathbb{R} \, , \tag{3.1}$$

where $\mathbf{s} \in \mathbb{R}^D$ is the spatial coordinate in a D-dimensional space and t is a real number that represents time. In this chapter, the domain of f is simplified to a single input variable $\mathbf{x} = (\mathbf{s}, t) \in \mathbb{R}^{D+1}$. Therefore f is defined as

$$\begin{aligned} f(\mathbf{x}) \; : \; \mathbb{R}^{D+1} &\rightarrow \mathbb{R} \\ \mathbf{x} &\rightarrow y = f(\mathbf{x}) \; , \end{aligned} \tag{3.2}$$

where $\mathbf{x}$ is the spatial coordinate augmented with a temporal component and represents a spatial temporal location.

The function f that represents the environment is unknown and only noisy samples from it are available. A sample y from f at a spatial-temporal location $\mathbf{x}$ is given by $y = f(\mathbf{x}) + \epsilon$, where ϵ represents the noise introduced when sampling from f and follows a normal distribution, $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$, with standard deviation σ_n.

A model of f provides an estimation of $f^\star$ at a new location $\mathbf{x}^\star$. Determining $f^\star$ given a set of N noisy samples from f, $S = \{\mathbf{x}_i, y_i\}_{i=1}^N$, is a regression problem, which in principle could be tackled using any regression technique. However, not all regression techniques are suitable for the decision making algorithms presented in this thesis. These algorithms need a measure of uncertainty, which is encoded in the variance of a probability distribution. Thus, only the regression techniques that provide a posterior probability distribution over f, sometimes called predictive distribution, are useful. The posterior *Probability Density Function* (PDF) is given by

$$p(f^\star | \mathbf{x}^\star, S) \; . \tag{3.3}$$

A predictive distribution is useful because it provides not only an expectation of the predicted value, $\mathbb{E}[f^\star]$, but also delivers an idea of uncertainty, which is encoded in its variance $\mathbb{V}[f^\star]$. Both the expected predicted value and the uncertainty will be a fundamental piece in designing algorithms for monitoring initially unknown functions that vary with space and time.

The presence of noise in measurements and limitations in the maximum complexity

of the model introduce uncertainty in the prediction. Considering this, using statistical/probabilistic models seems a reasonable alternative. A very popular regression technique that provides posterior PDFs is *Gaussian Process Regression* (GPR), which was detailed in Section 2.2. We use a *Gaussian Process* (GP) as a regression tool for predicting $f^\star$, i.e. a GP is used as a statistical model or representation of an environmental phenomenon f,

$$f\left(\mathbf{x}\right) \sim \mathcal{GP}\left(m\left(\mathbf{x}|\boldsymbol{\theta}_m\right), k\left(\mathbf{x}, \mathbf{x}'|\boldsymbol{\theta}_c\right)\right) \ . \tag{3.4}$$

Here, $m\left(\mathbf{x}|\boldsymbol{\theta}_m\right)$ represents the mean function parametrised by a set $\boldsymbol{\theta}_m$ and $k\left(\mathbf{x}, \mathbf{x}'|\boldsymbol{\theta}_c\right)$ represents the covariance function parametrised by $\boldsymbol{\theta}_c$.

As it was detailed in Section 2.2, GP regression provides a posterior PDF for $f^\star = f(\mathbf{x}^\star)$. It uses a set of noisy observations from f, $S = \{\mathbf{x}_i, y_i\}_{i=1}^N$. Each sample is modelled by $y = f(\mathbf{x}) + \epsilon$, where $\epsilon \sim \mathcal{N}\left(0, \sigma_n^2\right)$ with σ_n being the noise standard deviation. This value adapts to different noise levels in the sampling process from f and is part of the hyper-parameters set $\boldsymbol{\theta}$.

The posterior PDF is gaussian $f^\star \sim \mathcal{N}\left(\mathbb{E}[f^\star], \mathbb{V}[f^\star]\right)$. The mean $\mathbb{E}[f^\star]$ and variance $\mathbb{V}[f^\star]$ are given by

$$\begin{aligned} \mathbb{E}[f^\star] &= K(\mathbf{x}^\star, X)K_X^{-1}(\mathbf{y} - M(X))\,, \\ \mathbb{V}[f^\star] &= k(\mathbf{x}^\star, \mathbf{x}^\star) - K(\mathbf{x}^\star, X)K_X^{-1}K(X, \mathbf{x}^\star)\,, \end{aligned} \tag{3.5}$$

where X is the set of training locations; $K_X = K(X, X) + \sigma_n^2 I$ with $K(\cdot, \cdot)$ being the covariance matrix, defined component wise as

$$K(X', X'')_{(i,j)} = k(x_i', x_j'')\,, \qquad \text{with} \quad x_i' \in X' \ \text{and} \ x_j'' \in X''\,; \tag{3.6}$$

and $M(X)_{(i)} = m(x_i)$ with $x_i \in X$.

The posterior PDF depends on the set of sampled locations X, the set of outcome values corresponding to each location and the selection of the mean function m and covariance function k with its respective hyperparameters $\boldsymbol{\theta}_m$ and $\boldsymbol{\theta}_c$. Without loss of

generality, it is assumed that the mean function is a constant $m(\mathbf{x}) = \eta$, i.e. $\boldsymbol{\theta}_m = \{\eta\}$. The covariance function encodes a degree of relationship between input locations. k corresponds to any covariance function or a combination of them, as described in Section 2.2. The main intuition here is that samples that are close in space and in time will expose statistic dependence. However, given the compact model for f, defined in Equation 3.2, the same covariance function is applied to all dimensions in the input, which is not ideal for capturing time or spatial specific behaviour. This limitation will be relaxed and further detailed in the following chapter that introduces more complex spatial temporal models.

3.3 Waypoint Planning

A *Gaussian Process* (GP) provides a model for the spatial temporal phenomenon. It is necessary to choose sensing locations wisely to build a high-quality model of the phenomenon at each time step. The algorithm presented in this section is a generalisation of the plain *Bayesian Optimisation* (BO) algorithm described in Section 2.3 to select locations based on the posterior PDF provided by the GP model of the environment.

Existing approaches choose the path of one or more robots based only on the existing uncertainty of the expected value over the entire domain. However, in most environmental monitoring applications, the places of interest for sampling are associated with extreme values of the sampled variable. For example, areas of high ozone concentration at lower altitude, or areas of high pollutant concentration. Therefore, this problem can be addressed as an optimisation problem where the objective function is not known a priori. However, not every optimisation strategy is useful, since the main goal is to find parts of the function that exhibit extreme values (maximum or minimum), requiring an exploration of the domain to understand what happens across the studied area. This is a non trivial problem considering that we are dealing with an unknown and complex function modelling a time dependent process.

Given all the above, the use of the BO framework fits logically because it can optimise

initially unknown noisy functions and collects more samples in the higher or lower regions of the domain. A complete and extensive theoretical treatment of the general BO algorithm can be found in Section 2.3. In order to use BO for a real-world function, there are some adaptations that need to be performed:

1. The costly-to-evaluate goal function is the dynamic phenomenon present in the environment.

2. The smart selection of evaluation locations in the BO algorithm corresponds to planning in the real-world domain.

3. Plain BO as presented in Section 2.3 assumes one can sample any point in the domain of f. However, given the restrictions of the robot, this is not possible.

4. The maximisation of the acquisition function h is translated into maximising a reward function r, which is equivalent to making decisions.

The type of planning presented in this section defines decisions as locations in space and time, i.e. for each iteration of the algorithm, a spatial–temporal goal $\mathbf{x}^\star$ is determined as the best location to sample. This is the *discrete* nature of the algorithm, where a set of discrete locations determine where the following samples will be gathered. Although this is a simple approach for planning, the following chapters propose more complex solutions over continuous paths and address the problem as a sequential decision making problem.

From the robotics point of view, it is useful to get as much information as possible from the environment while sampling at a small number of locations. If these locations are placed and ordered intelligently, the robot will save energy while constructing a reliable spatial-temporal model of the phenomenon. The reward function is defined in terms of an acquisition function h, described in Section 2.3.1, which also considers the cost of travelling in the environment, taking into account the current pose of the robot. The reward for sampling at $\mathbf{x}$ given the current pose $\mathbf{p}$ of the robot is:

$$r(\mathbf{x}|\mathbf{p}) = h(\mathbf{x}) - \gamma\, d(\mathbf{x}, \mathbf{p}) \ . \tag{3.7}$$

Algorithm 2 BO-Discrete-Planning

Inputs: f, r, GP-prior, $\mathbf{p}_{\text{init}}$
Outputs: GP-posterior, $\mathbf{p}$

```
p current pose of robot.
p ← p_init
while t < t_max do
    x* ← arg max_x r(x|p)
    while p is far from x* do
        Move towards x*
        Gather samples from f
        Update p
    end while
    Augment X and y with new samples.
    Update the GP model
end while
```

Here $d(\cdot, \cdot)$ is a distance metric between the current pose of the robot and the goal location considering only spatial components. γ is a parameter that determines the cost for travelling in the environment which depends on the platform. This improved reward function combines a regular acquisition function and a penalty based on the distance to the current location of the sensor, defining a new family of acquisition functions that are state aware. The intuition behind this new reward function is to prioritise sampling locations closer to the current pose of the robot that carries the sensing equipment. This penalty can be generalised depending on the application to consider energy consumption, smoothness of trajectory or any other function that depends on the state of the robot. Another state-aware acquisition function has been proposed recently by Contal et al. [8], however it is limited only to Mutual Information.

Algorithm 2 presents the pseudo–code for efficiently monitoring a spatial-temporal phenomenon with a robot. As it was mentioned earlier, this algorithm is an adaptation of the plain BO optimisation algorithm (Algorithm 1) introduced in Section 2.3. The main modifications are the following:

Inputs: The inputs of the plain BO algorithm are f, h and the GP-prior. For the proposed algorithm, the acquisition function h has been replaced by a reward function

r, which considers the state of the robot.

Outputs: Since the goal of the plain BO algorithm is to find the optimum, it provides as outputs the location and optimum value found. However, the goal of the proposed planning algorithm is to model f while focusing on areas of extreme values of it. The output of the algorithm is the model of f and the trajectory of the robot.

Optimisation of the Acquisition Function: Line 4 of Algorithm 2 determines the next best location to gather a sample from f by maximising the reward function r. This maximisation can be conducted with any type of inexpensive optimisation routine, including gradient based optimisers. Even if the selected location corresponds to a local optima of the acquisition function, the variance of the GP model will reduce after that observation is included, leading the optimisation outside that local optima in the next iteration. In contrast to the plain BO algorithm, this optimisation step considers the current pose $\mathbf{p}$ of the robot.

Sample Collection: In the plain BO algorithm a sample is collected at every iteration and the GP model is updated using every observation. However, in the proposed algorithm a set of samples is collected while the robot travels towards the goal location $\mathbf{x}^\star$ and the GP posterior is updated once the goal is reached. Line 5 is the beginning of a loop, which collects samples while the robot is traversing a path towards $\mathbf{x}^*$. Effectively, the sample collection for each iteration will cease when the pose of the robot is close to the goal location, where the mathematical interpretation of close will depend on the scale of the environment.

3.4 Experiments

The proposed decision making algorithm was evaluated under three different scenarios. The first one is an illustrative example of the method under a 1D goal function. The second simulates a robot monitoring ozone concentration over a real dataset obtained across the entire US territory. The last experiment shows a real-world scenario, using an autonomous mobile robot to model ambient light conditions in an indoor

environment.

3.4.1 1D Illustrative Example

In this simple case of a one dimensional function the goal is to identify the effect of using a distance aware reward function. It is expected that consecutive samples are chosen by the algorithm closer to the previous sample depending on the distance penalty.

The goal function f is unknown to the algorithm. For simplicity it does not depend on time and is given by

$$\begin{aligned} f(x) \,:\, [0,1] &\to \mathbb{R} \\ x &\to y = 0.5\left[\sin\left(17(x+0.3)\right)\cdot(x+0.3)^{0.5} - 0.7\cos\left(30x\right)\right] \,. \end{aligned} \tag{3.8}$$

A plot of f is shown in Figure 3.1. The complexity of the expression of f is chosen in such a way that presents several local maximum. In particular, it presents five locally maximum location-value pairs $\{x, y\}$: $\{0.12, 0.56\}$, $\{0.30, 0.05\}$, $\{0.53, 0.82\}$, $\{0.75, -0.12\}$, $\{0.93, 0.82\}$. The characteristics of this function allow the analysis of the effects of varying the reward function and their parameters.

For this example, the reward function is built using h = UCB as the acquisition function, previously presented in Section 2.3.1. The expression of the reward function is given by

$$r(\mathbf{x}|\mathbf{p}) = \mu(\mathbf{x}) + \kappa\sigma(\mathbf{x}) - \gamma\, d(\mathbf{x}, \mathbf{p}) \,, \tag{3.9}$$

where κ and γ are parameters that will affect the locations being chosen as new sample locations. A greater value for κ encourages exploration and a larger value of γ penalises the distance between consecutive samples.

According to Algorithm 2, all that remains is to define the GP model. The mean function is selected to be a stationary constant $m(\mathbf{x}) = \eta$, i.e. $\boldsymbol{\theta}_m = \{\eta\}$, and the selected covariance function is the 1D version of the Squared Exponential expression

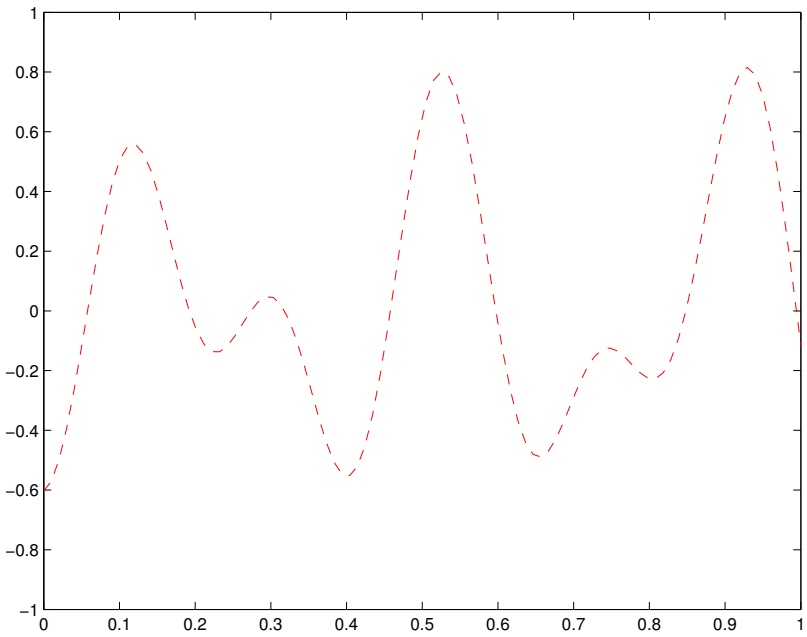


Figure 3.1 – Goal function f.

presented in Table 2.1, given by:

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{(x - x')^2}{2\sigma_l}\right) . \tag{3.10}$$

In the above, $\boldsymbol{\theta}_c = \{\sigma_f, \sigma_l\}$, where σ_f is the signal variance and σ_l is known as the length-scale hyper-parameter.

The optimisation of the hyper-parameter set $\boldsymbol{\theta}^\star = \{\eta, \sigma_f, \sigma_l, \sigma_n\}$, with σ_n being the standard deviation of the noise in observing f, is conducted using a gradient based optimiser and the LML goal function, as described in Section 2.2.2. The resulting hyper-parameters found were:

$$\boldsymbol{\theta}^\star = \{0.0945, 0.6922, 0.0789, 0.0486\} . \tag{3.11}$$

These hyper-parameters are used in the GP model of f and are kept constant. The dataset used to train the hyper-parameters corresponds to random sampling of the function. This is analogous to random movement of a robot with the goal of learning

the type of function to be optimised and select an appropriate prior. There is no decision making involved in this in this process.

Figure 3.2 presents the optimisation procedure with $\kappa = 28.6$ and $\gamma = 0.72$. It is possible to see how the mean of the GP regression model quickly converges to the unknown function and the variance is reduced near sampling locations. The behaviour of the acquisition function changes through time[2]. Initially, when the variance is high, the sampled locations are automatically chosen by the algorithm to get an initial approximation of the unknown function. In the long term, samples concentrate near higher values of the unknown function. The effect of the distance-penalty in the reward function is clear, as the next sampled location (circle) is chosen to be close to the last sampled location (vertical line).

3.4.2 Large-Scale Pollution Monitoring

The goal of this experiment is to simulate a robot sampling from a known phenomenon where ground truth is available so as to compare different approaches. In order to build the ground truth, this experiment uses part of a real-world environment dataset, made available by the United States Environmental Protection Agency[3]. This is a considerably large dataset, covering the United States territory with hourly samples dating back to 1987, including meteorological variables such as temperature, humidity, solar radiation and ozone concentration among many others.

The environmental phenomenon to be monitored in this experiment is the ozone concentration in parts per billion (ppb) with raw data provided by $N = 103$ static monitoring stations across the US territory for the 1st of August 2009. The ground truth of the phenomenon is built using a spatial- temporal GP described in Section 3.2. The GP is trained with timestamped ozone concentration data, and uses a Squared Exponential covariance function (Table 2.1), whose optimal hyper-parameters were

[2]A video showing the iterations is available at `http://www.it.usyd.edu.au/~rmar5258/IROS2012/1d_example.html`

[3]Dataset web access: `http://java.epa.gov/castnet/reportPage.do`

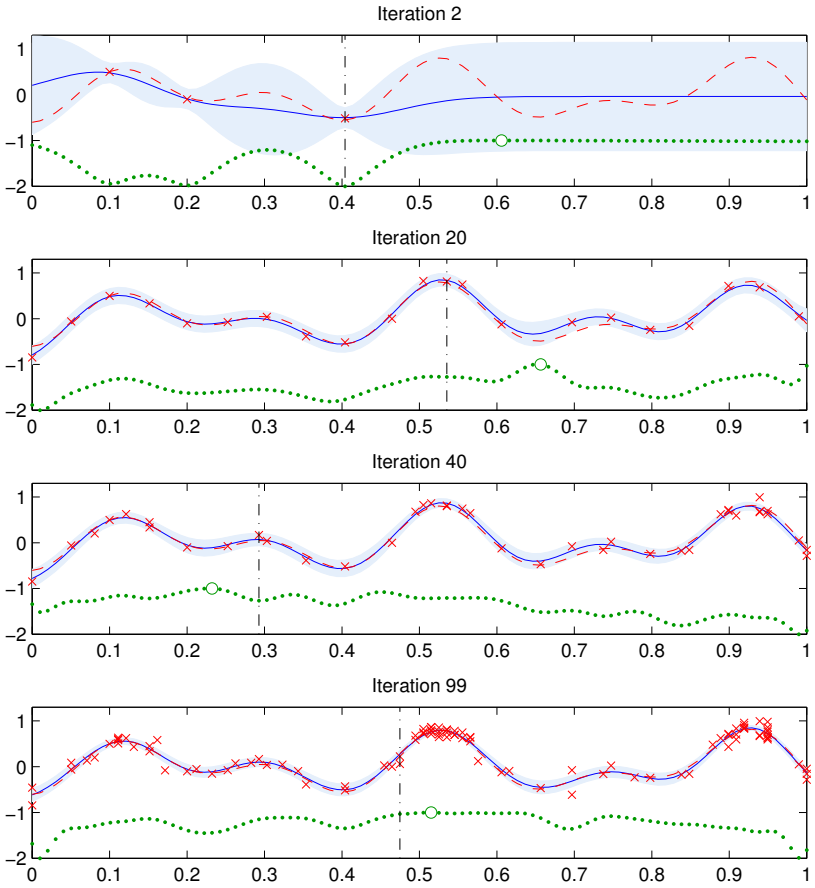
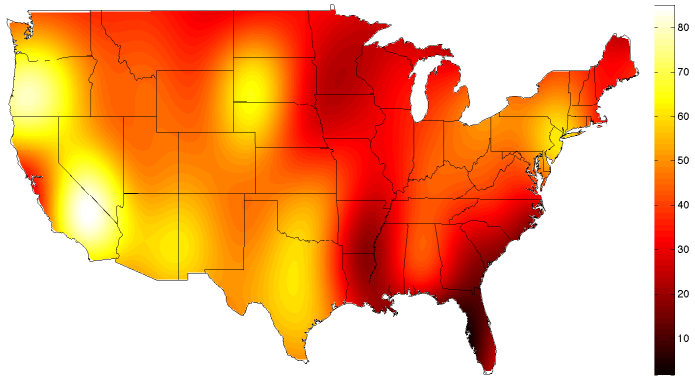


Figure 3.2 – One dimension example of active sampling based on BO. The continuous blue line and shade represent the GP mean and variance respectively. The dashed red line is the unknown function, and noisy samples from this function are shown as red crosses. The dotted green line is the reward function, with a circular mark on its maximum. This function is scaled and with an offset for visualisation purposes. The last sampling location for each iteration is shown with a vertical dash-dot line.

found by maximising the log marginal likelihood,

$$\boldsymbol{\theta}^{\star} = \{\sigma_f, \ell_{lat}, \ell_{long}, \ell_t, \sigma_n\} = \{14.02, 4.1, 4.1, 0.1336, 2.09\} \ . \tag{3.12}$$

An isotropic restriction of having the same value for both spatial length scales is enforced during the optimisation process ($\ell_{lat} = \ell_{long}$), because of the known isotropic behaviour of gas concentration in space. The mean of this GP (μ_{gt}), which follows Equation 3.5, is shown in Figure 3.3 for a specific timestamp. μ_{gt} is used as ground truth for the entire spatial temporal domain. Higher concentration of ozone



(a) Ground Truth

Figure 3.3 – Mean of the ground truth GP of ozone concentration [ppb] for $t = 0.62[day]$

is observed in the west region of the *United States of America* (USA), whereas lower concentration of ozone is found in the south east area of Florida.

The sampling performed by a mobile robot is simulated by noisy measurements from the ground- truth GP mean μ_{gt}, with $\sigma_n = 2$. The BO-Discrete-Planning algorithm (Algorithm 2) is ran over the entire temporal domain, i.e. 24 hours, for the reward function from Equation 3.9 under three different configurations that affect importantly the behaviour of the robot:

I. *Information Gain* (IG), which corresponds to a very large, but finite, value of κ, $\kappa = 10^{10}$, and $\gamma = 0$. This is a reward function oriented towards pure exploration, as all attention is focused on reducing the uncertainty in the model.

II. *Upper Confidence Bound* (UCB), with manually tuned values $\kappa = 8$ and $\gamma = 0$, considers the predicted value and the uncertainty in the model, disregarding the distance between sampling locations.

III. *Distance-based Upper Confidence Bound* (DUCB), with $\kappa = 8$ and $\gamma = 7$, is the more complete reward function, that considers the predicted value, the uncertainty associated to it and the distance between sampling locations.

The same starting point $\mathbf{x}_0 = (\text{Long}, \text{Lat}, t) = (-95, 40, 0)$ was selected for all three experiments. The simulation process considers a hypothetical *Unmanned Air Vehicle*

(UAV) that travels at a speed of $600km/h$. Given the short length scale in the time dimension, high speed helps dealing with the large distances that need to be covered by the robot in a small amount of time. This may seem unrealistic in practice. However, the results are equally valid for lower scale problems where the size of the domain and speed are reduced proportionally.

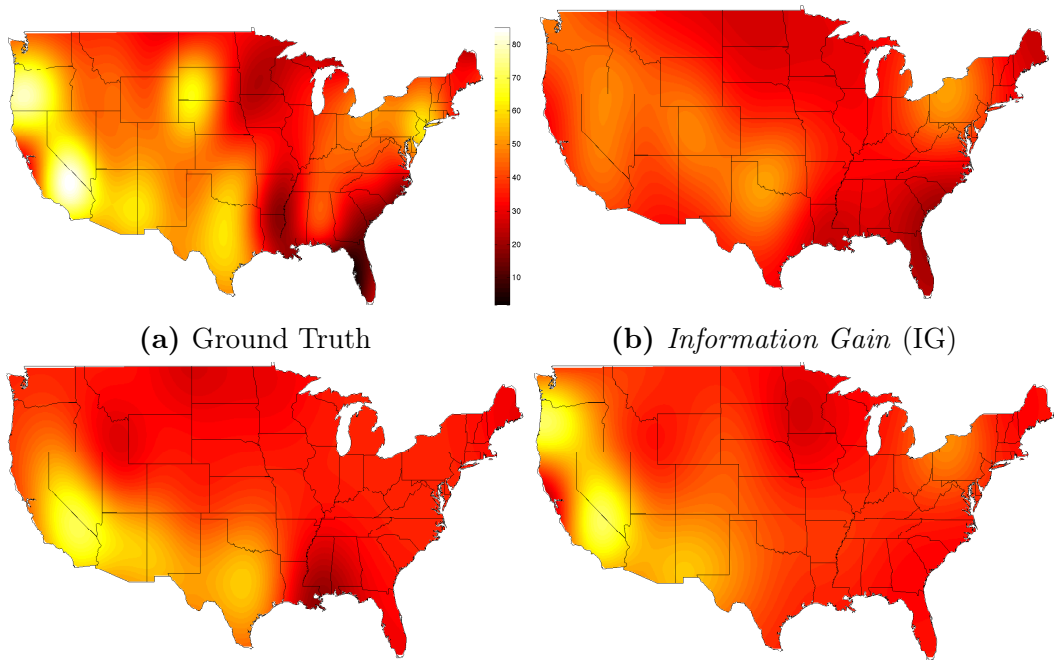
Qualitative Results

An iterative process is executed for the 24 hours of simulated data[4]. In each iteration, a robot moves towards the last selected goal. Each robot uses its own reward function for selecting a new goal after reaching the previous one. The concentration of ozone gas particles is sampled every 10 minutes while the robot moves towards the goal. This ensures that robots take a similar number of samples over the experiment, the sole difference being the places where measurements are taken, allowing a proper comparison of the three different algorithms. The estimated GP model of the phenomenon is recalculated after a new sample is acquired. Figure 3.4 shows the estimated mean of the GP model at an arbitrary time step ($t = 0.62$ days) of the simulation for the three sampling strategies. The trajectories followed by each of the robots during the 24 hours sampling window are shown in Figure 3.5.

A visual inspection of the IG results (Figure 3.4b) shows that it matches the ground truth data poorly. The IG strategy exhibits poor fit to the overall space, not showing any preference for higher or lower ozone concentrations. This is expected, given that the reward function does not take into account the predicted value of ozone itself, only the uncertainty in its predictions. The *Bayesian Optimisation* (BO) planning approach with UCB (Figure 3.4c) and with DUCB (Figure 3.4d) present a better fit to the data. Both of them model correctly the highest spike in ozone concentration in the west of the USA. However, using the DUCB seems to capture in a better way the two high concentration areas in the west of the USA.

The trajectories followed by each robot during the 24 hours of simulation are shown

[4]A video showing the ground truth, mean estimation, the acquisition function and the trajectory for each approach can be found at `http://www.it.usyd.edu.au/~rmar5258/IROS2012/results.html`



(a) Ground Truth **(b)** *Information Gain* (IG)

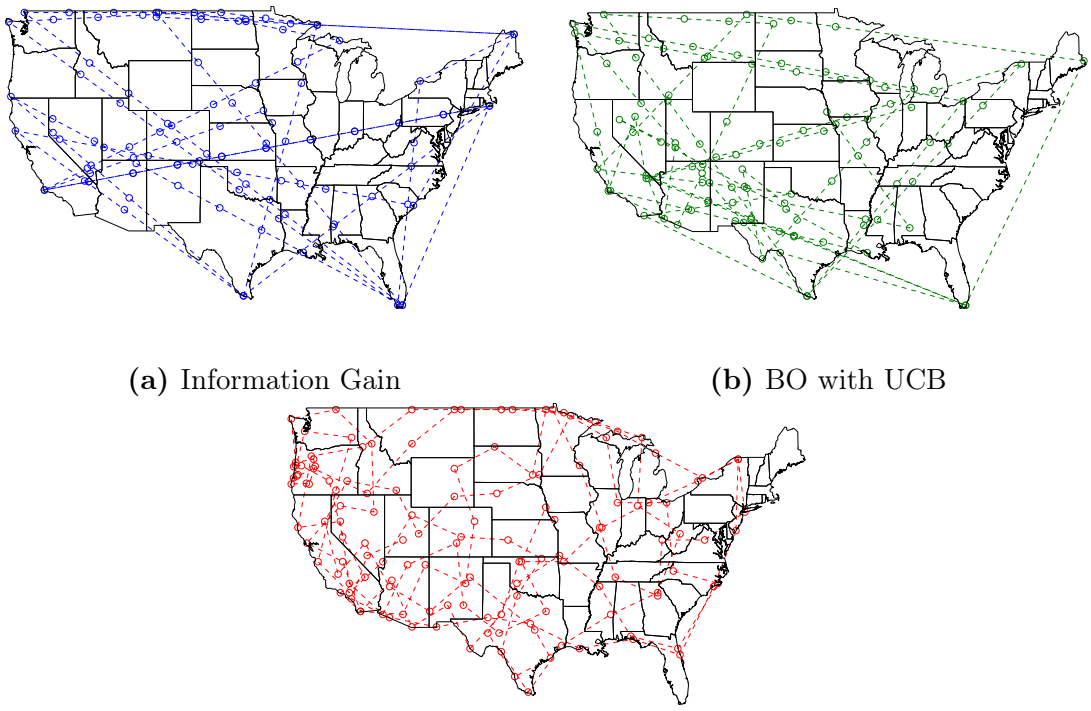
(c) *Upper Confidence Bound* (UCB) **(d)** *Distance-based Upper Confidence Bound* (DUCB)

Figure 3.4 – Estimated mean of a GP model of ozone concentration ppb. Colour-scale showed in (a) and instant of prediction ($t = 0.62$ days) is the same for all figures.

in Figure 3.5. It can be seen that the IG sampling strategy selects locations far away from each other and usually correspond to the extremes of the territory. A similar behaviour is observed for the strategy using the UCB reward function, although the sampling locations tend to concentrate in the south west of the U.S. territory. This is because the mean of the ozone concentration is higher in that area over the whole day. Finally the approach using DUCB as a reward function provides better results, where each sampling point is near to the last one. This strategy distributes measurements over the whole domain, while sampling more often in high ozone concentration areas.

Quantitative Results

The three different reward function are quantitatively compared using two different performance indicators, that make reference to the error between the ground truth and the model after each time step. The performance indicators are calculated over the whole domain, using a fine grid resolution with M samples. The first one is the



(a) Information Gain **(b)** BO with UCB

(c) BO with DUCB

Figure 3.5 – Trajectories followed by robots. Observations are only collected inside the US territory.

Root Mean Squared Error (RMSE) that reflects the error in estimation independent of the predicted value. This means that RMSE indicator gives the same importance to the error where the ozone concentration is low, than where the ozone concentration is high. Therefore RSME is not the best indicator for environmental monitoring applications but shown here as a reference. The second performance indicator is the *Weighted Root Mean Squared Error* (WRMSE),

$$\text{WRMSE} = \sqrt{\frac{\sum_{i=1}^{M} \left[\frac{(\mathbb{E}[f(\mathbf{x}_i)] - \mu_{gt}(\mathbf{x}_i))(\mu_{gt}(\mathbf{x}_i) - \min \mu_{gt}(\mathbf{x}_i))}{\max \mu_{gt}(\mathbf{x}_i) - \min \mu_{gt}(\mathbf{x}_i)} \right]}{M}}. \tag{3.13}$$

WRMSE is essentially identical to RMSE but the error is multiplied by a factor that depends on the mean of the predicted value, normalised between the minimum and the maximum over the entire domain. This performance indicator gives more importance to the error in areas with higher values of the studied phenomenon, a

Table 3.1 – Results for Simulated Experiment, Mean and Std are in ppm

Indicator	**Method**	**Mean**	**Std**	**Distance** $10^3 km$
RMSE	IG	13.78	2.31	717.38
RMSE	UCB	13.04	2.26	718.25
RMSE	DUCB	12.33	3.47	639.06
WRMSE	IG	8.21	1.78	717.38
WRMSE	UCB	7.10	1.65	718.25
WRMSE	DUCB	6.51	1.21	639.06

logical assumption if we are interested in potentially dangerous areas for humans. WRMSE is a suitable performance indicator for this particular application where ozone is a serious pollutant at ground level.

Table 3.1 presents the results for each sampling method using the two different indicators. For each method, we calculate the mean and standard deviation of the indicator through time and the total distance the robot travelled.

Overall, it can be seen that the DUCB method has the smallest error in RMSE and in WRMSE. The standard deviation of DUCB for the RMSE indicator is higher than the other strategies because DUCB avoids sampling in areas where the ozone concentration is known to be low. Therefore, it presents small error in high ozone concentration areas and larger error in lower ozone concentration areas, increasing the standard deviation of the RMSE. The IG approach has the bigger error but smaller standard deviation because it distributes measurements uniformly over the entire domain. Regarding WRMSE, the DUCB method has the best results in terms of mean and standard deviation because the error in lower ozone concentration areas is less relevant (Equation 3.13). The distance travelled by the DUCB sampling strategy is 11% smaller than the other approaches due to the distance penalty in the acquisition function. This means that our proposed DUCB acquisition function results in lower error and at the same time reduces energy consumption.

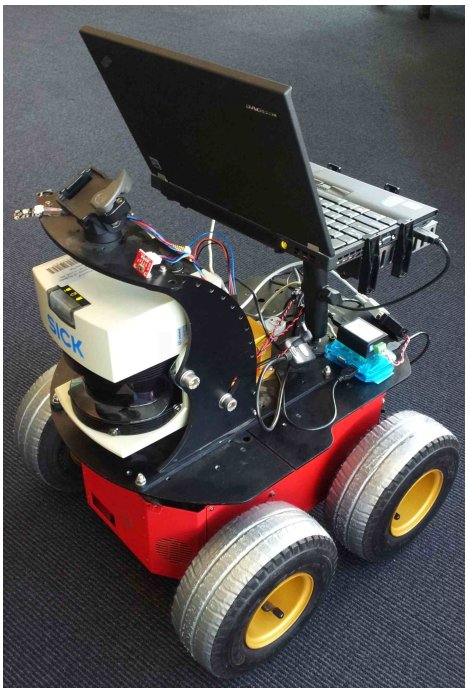


Figure 3.6 – Mobile robot used for indoor experiment.

3.4.3 Luminosity Monitoring

The last experiments consist on testing the planning algorithm in a real environment using an autonomous robot. The robot, shown in Figure 3.6, is equipped with a laser scanner, odometry, and an ambient light sensor. It uses a laptop running the *Robot Operating System* (ROS)[5] as main software infrastructure. The built-in packages in ROS deal with localisation and mapping using the laser scanner and odometry. Therefore, the robot is assumed to be properly localised during the whole measurement process, using a localisation algorithm developed by Fox [16].

The monitored phenomenon is the ambient light distribution in an indoor office environment. The intensity of the ambient light was kept constant during the execution of the experiment to allow repeatability (no variation of the process with time). Figure 3.8a shows the map built autonomously by the robot, where the shaded area represents the selected area for modelling the phenomenon. To create the ground truth of

[5] `http://www.ros.org`

the intensity of light, the robot was driven manually over the whole domain taking a considerable amount of measurements. These were used to learn a *Gaussian Process* (GP) as the ground truth (shown in Figure 3.7a), with optimal hyper-parameters

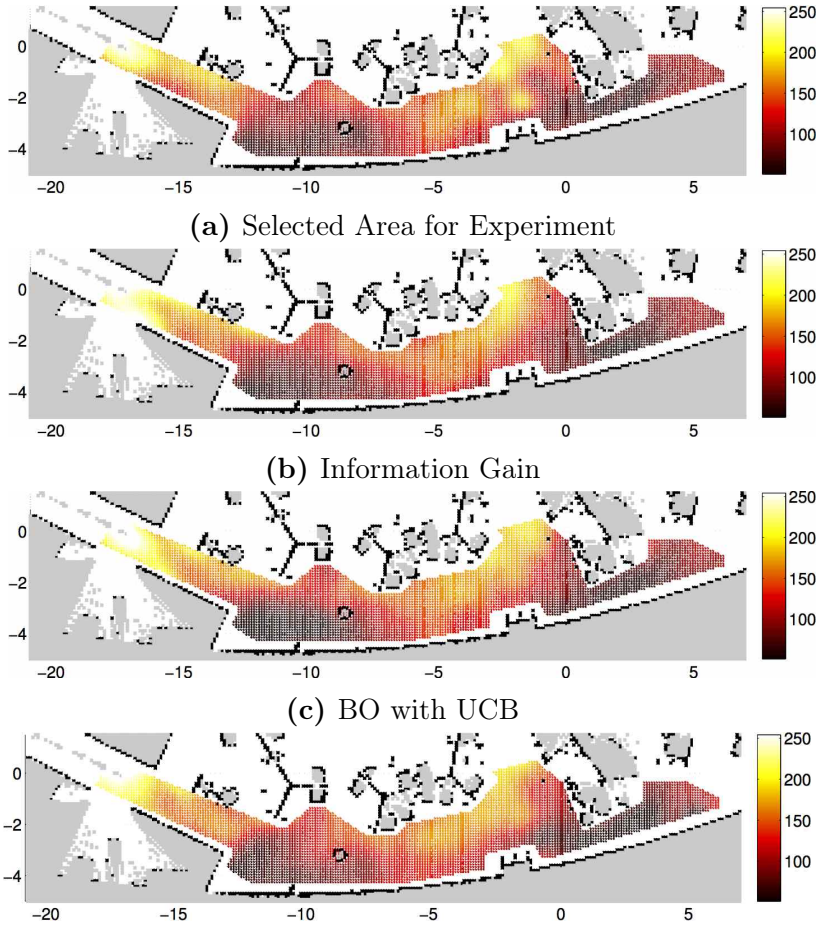
$$\boldsymbol{\theta}^{\star} = \{\sigma_f, \ell_{lat}, \ell_{long}, \sigma_n\} = \{40.01, 0.7, 0.7, 6.02\} \ . \tag{3.14}$$

The ground truth GP allows evaluating each algorithm's modelling performance.

The three different approaches in 3.4.2 are compared in the described real environment using the autonomous robot. Each sensing strategy was tested for 10 minutes, acquiring one measurement per second. The final model built by each policy is displayed in Figure 3.7 where higher light intensity is equivalent to high values in the plot (the luminosity scale and absolute values could not be provided due to the nature of the sensor). The three models preformed very similar because in this experiment the process did not vary in time, i.e., it is expected that all policies produce an accurate model if they have visited the entire domain.

Figure 3.8 shows the trajectories followed by the robot using the three sampling strategies. It can be seen that the resulting trajectories of the IG and UCB approaches are spread over the whole domain. This is because after reaching a goal, the next selected sensing location is very far from the last one. A slight difference can be observed when using the UCB strategy (Figure 3.8c), because sensing locations are more concentrated in areas with higher luminosity values. The trajectory followed by the robot using our proposed acquisition function DUCB (Figure 3.8d) is clearly more concentrated in areas of higher ambient light. In contrast, areas of lower ambient light are only visited once during the experiment, which is enough for the underlying phenomenon in this experiment. Comparing against a distance penalised information gain criterium (DIG) may be possible; however, this acquisition function is not suitable as it does not consider the mean of predictions.

Table 3.2 details the error of prediction for each sensing method, using the RMSE and WRMSE over the whole spatial domain (no mean and standard deviation of the indicators are provided as the process does not vary with time). It can be seen



(a) Selected Area for Experiment

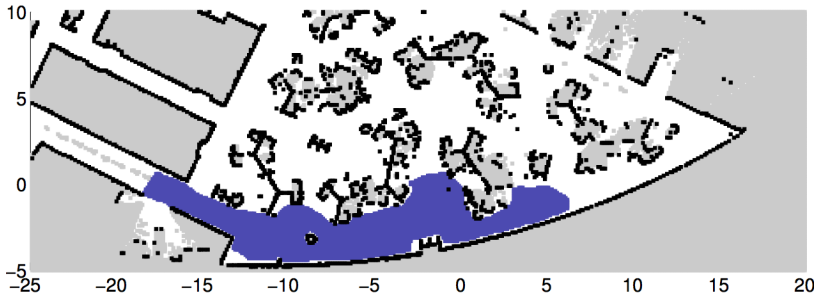
(b) Information Gain

(c) BO with UCB

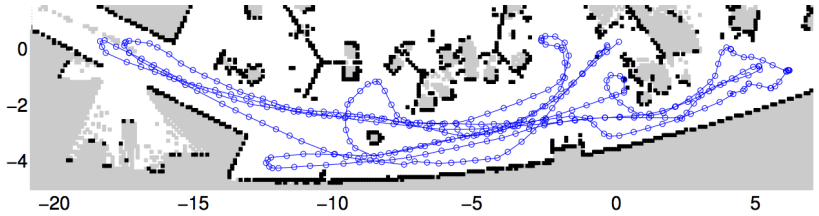
(d) BO with DUCB

Figure 3.7 – Estimated mean of a GP fitted to training data of each sensing strategy. Light intensity with no international unit scale. Axis show distance in metres.

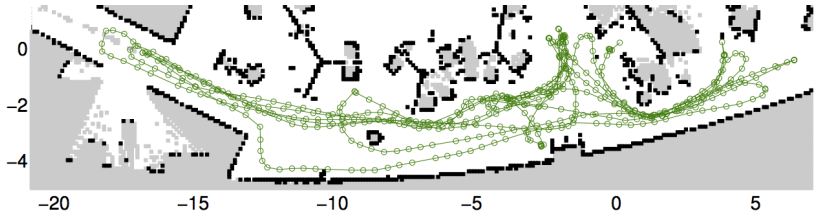
that the RMSE is similar for all three different strategies. However, our proposed strategy using distance penalty reduces the travelled distance almost to half of the other strategies, while building a similar model. Having said that, it is clear that our sensing strategy produces a more efficient sampling methodology.



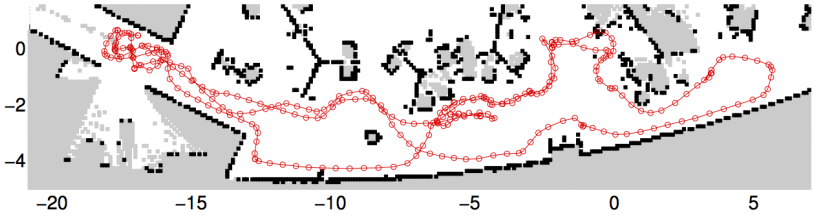
(a) Selected Area for Experiment



(b) Information Gain



(c) BO with UCB



(d) BO with DUCB

Figure 3.8 – The trajectories followed by the robot. Samples are only taken inside the allowed area defined by (a). Axis show distance in meters.

3.5 Related Work

3.5.1 Spatial-Temporal Modelling

The idea of predicting the value of a process in a confined space based on a set of examples is not new and has been widely studied. Matheron [48] pioneered this concept in geo-statistics in the 1970's. His work tackles the problem of estimating the value of

Table 3.2 – Results for Real Experiment

Indicator	**Method**	**Value**	**Distance** m
RMSE	IG	16.44	157.01
RMSE	UCB	17.16	217.45
RMSE	DUCB	16.91	98.08
WRMSE	IG	5.41	157.01
WRMSE	UCB	4.60	217.45
WRMSE	DUCB	4.70	98.086

an unknown function z in a restricted domain considering a set of known values $\hat{z}$ in specific locations. Applied to geo-statistics, the solution of this problem is known under the name of *universal kriging* and is related to intrinsic random functions. Gaetan and Guyon [17] present an updated theoretic description of universal kriging in page 43. Daley [11, chap. 2] explores the function fitting problem from the point of view of atmospheric data analysis. His research addresses a general problem called *global fitting*, were K samples are operated in pairs using basis functions and generate a Gram matrix. This matrix is then used to predict the value of the variable of interest at a particular position.

Le and Zidek [37] provide an statistical framework for modelling space-time processes. Initially, they analyse the typical approaches for modelling spatial correlated data, similarly to Matheron [48] and Gaetan and Guyon [17]. They explicitly consider relations in space-time domain and show a Bayesian kriging approach where a prior distribution is assigned to the hyper-parameters. Le and Zidek [37] present a theoretic analysis of various aspects of spatial temporal modelling, for example on how to determine suitable covariance functions, analysing separability and stationarity. Kyriakidis and Journel [33] categorise and review the existing geo-statistical space-time models.

Due to their successful results on modelling spatially correlated data in the past, kernel methods deserve particular attention. Schöolkopf and Smola [62] define kernels and present a brief analysis on Bayesian non-parametric methods for regression

and classification, particularly Gaussian Processes (GPs). In fact, Schöolkopf and Smola [62, page 480] state that it is natural to assume correlation between spatially-referenced samples, depending on their sampled location.

Furthermore, GPs have been used successfully for modelling spatially correlated data. Some examples are terrain modelling [75], wireless signal strength [15], occupancy maps [52] and gas concentration [69]. Temporal phenomena and time series have also been modelled using GPs, such as in the case of breathing patterns [2], neural firing rate [10] and atmospheric CO_2 concentration [58, 76]. GPs for temporal prediction have demonstrated capabilities for pattern discovery [76]. Existing applications of GP-based spatial-temporal models include sea surface temperature [41], precipitation [60], ozone concentration [44] and luminosity [44].

Another strategy involving GPs for learning a spatial model is presented by [74] and [69]. Tresp [74] presents the idea of combining GPs as building blocks of a more complex structure. Stachniss et al. [69] use a *Gassian Process Mixture* (GPM) that allows representing a smooth signal with impulsive peaks in particular locations, explicitly distinguishing different components. They claim that the use of this approach improves the prediction accuracy. However, it involves learning a complex mixture model from the data. Tresp [74] uses Expectation Maximisation to estimate the probability of correspondence between each data point and a GPM component, learn the hyperparameters of each GP of the mixture, and estimate a gating function for combining GP components on each new test point. Given the increasing number of parameters to be learned, GPMs may result in over-fitting.

Higdon [24] shows that space-time GP modelling can be viewed from another point of view. Although the correlation between spatial locations is usually modelled with the covariance function, Higdon suggests modelling it using the convolution between a basis function and a white noise process (latent process). He postulates that different choices of the smoothing basis function and the latent process lead to attractive approaches, such as modelling non-Gaussian distribution processes, space time dependence and non-stationary spatial correlations among others. Similarly to [74] and [69], Higdon [24] proposes adding different-scale component convolution processes to

boost the performance of the model, exploiting multi-resolution and allowing each component to model details in diverse scales. Gaetan and Guyon in [17] present a detailed theoretical approach for continuous and discrete process convolutions in the spatial modelling context.

The algorithms presented in this thesis are not strictly tied to any spatial-temporal model in particular. However, they do require that the model provides a posterior probability distribution over the environment state space.

3.5.2 Planning and Decision Making

It is generally believed that the accuracy of a model and its generalisation capability are highly dependent on how the natural phenomenon is sensed. Le and Zidek [37, page 36] state that the first step is to select the location and time to conduct measurements. Intuitively, sensing at a large number of places will increase the accuracy of the model. However, Delmelle [13, page 183] points out that although processing large amount of observations may result in a complete understanding of the phenomenon, extensive sensing is not possible due to excessive sensing cost and limited resources. Le and Zidek [37, page 35] state that measuring the complete phenomenon without error would reduce uncertainty but will consume infinite resources. For example, in an environmental monitoring application of a highly dynamic environment, the robot has a limited speed that places an upper bound on the number of sensing positions per interval of time. Therefore, wisely selecting sensing positions may improve significantly the overall performance of the model and result in higher prediction accuracy.

A usual approach for sensing is to sense where the uncertainty of the variable of interest is the highest (Javdani et al. [28]). From the information theory point of view (Mackay [42]), this corresponds to the most entropic locations. Le and Zidek [37, chap. 11] describe a strategy for placing a network of sensors based on maximising entropy. Because the entropy criteria focuses on edges of the environment, Guestrin et al. [22] suggest a new optimisation criterion based on mutual information. They address

the problem of placing several static sensors in optimal positions by maximising the mutual information between the sampling positions and the remaining un-sampled space. They prove that this is a NP-complete problem. As a consequence, they use an approximation algorithm to obtain near optimal results.

Krause and Guestrin [31] recognise mutual information as a submodular function that has interesting properties. They use a recursive greedy algorithm for walks, proposed by Chekuri and Pal [6], that chooses the next destination that maximises a submodular function. They consider path constraints for a mobile robot and generate a sequence of visiting locations for more than one robot.

Singh et al. [66] argues that the path planning problem is fundamental for environmental monitoring. They seek the optimal path that collects the most information of the phenomenon considering the displacement cost of each robot in a multi-robot system. Their approach is very similar to [31], but they added branch and bound and spatial decomposition to the path planning algorithm. The experiments conducted by Singh et al. [66] show that choosing different start locations for each robot yields the best approximation of the phenomenon as it obtains lower RMSE.

The decision making algorithms proposed in this thesis are related to the work in the active mapping community, where the goal is to efficiently build maps of initially unknown environments. The work by Kollar and Roy [30] provides useful ideas for path parametrisation. They solve the planning problem for active localisation and mapping using policy search, which shares the cumulative reward idea exploited in following chapters. Similarly, Le Ny and Pappas [38], Martinez-Cantin et al. [46] also apply uncertainty reduction towards active SLAM. The main difference between active SLAM and the problems addressed in this thesis is time variation, which introduces considerable complexity to the problem.

Stranders et al. [70] develop the first on-line, decentralised multi-agent coordination for environmental monitoring. They make use of GP regression, and sequentially choose the next sensing location of each robot that maximises the entropy of the visited positions. At every time step, the observations are sent to all robots and used to update the GP model distributively. Stransders et al. show simulated experimental

results of the same scenario studied in [22] to show the improvement obtained by mobile sensors against fixed ones. However, the work lacks a complete analysis. They cite Singh et al. [66] but fail to provide a comparison with the mutual information criterion and the pre-calculated paths.

Singh and Krause [65] extend the work developed in [31, 66]. They propose a new adaptive algorithm for solving submodular function optimisation. The new algorithm is adaptive since it considers the exploration-exploitation problem given starting and ending locations. The experiments section compares previously developed greedy algorithms and proves the expected improvement. However, they still recognise sub-optimal performance of the monitoring system.

Apart from the explicit time consideration and covariance function design, Singh et al. [67] modify the existing path planning algorithm. The novelty of their algorithm is that it considers a continuous domain and not discrete-space sensing locations as studied in [66], [31] and [65]. It uses the mutual information criteria to choose the next sensing location, and updates the covariance matrix of the process on every new sensed position.

3.6 Summary

This chapter presents a new planning strategy for monitoring time varying processes. The main contributions are:

1. Extension of the *Bayesian Optimisation* (BO) framework to choosing sensing locations in environmental monitoring applications. This allows mobile robots to take informed decisions based on the spatial-temporal GP model of the phenomena, getting as much relevant information as possible.

2. A new family of acquisition functions, called Distance based reward function. These reward function are state-aware and are energy efficient since they reduces the total distance travelled by the robot, considering distance between sample locations.

The proposed methodology was evaluated using simulation and with a real mobile robot. The results show a notable improvement in terms of error when monitoring extreme areas of time varying processes (ozone concentration experiment). A considerably important result is that our proposed distance-aware reward function helps reducing travel distances to achieve comparable results to other methods. This can be appreciated in simulated and in real experiments where the travelled distance was reduced by 40%.

The contributions made in this chapter enable an intelligent autonomous robot to efficiently monitor the state of an environmental phenomenon. We believe that using BO and designing new reward functions can help dealing with the difficulties when sampling complicated environmental processes.

Chapter 4

Planning over Continuous Paths

4.1 Introduction

This chapter presents a strategy for planning in continuous spaces for information gathering. It builds on the previous chapter and generalises the proposed planning algorithm from waypoint planning to continuous paths that can take any parametrisation. Briefly, the proposed algorithm uses two layers of *Bayesian Optimisation* (BO) to find the optimal parameters that define the best path a robot should follow in order to gather useful information from a spatial-temporal phenomenon. Part of the contents of this chapter have been previously published in ICRA[1].

The core of this chapter is to propose a solution for the spatial-temporal monitoring problem by finding informative paths in continuous domain, solving not only the question of *where* and *when* to sample, but *how* to get there. The developed decision making algorithm aims to identify areas of interest of an initially unknown environmental phenomenon, i.e. learn and monitor a spatial-temporal phenomenon. The decisions are based on an incremental spatial-temporal model of the phenomenon using a *Gaussian Process* (GP) prior, detailed in Section 2.2, which places a prior over the function space modelling a phenomenon over time. The algorithm is an extension

[1]Roman Marchant and Fabio Ramos. Bayesian Optimisation for Informative Continuous Path Planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014

of *Bayesian Optimisation* (BO), studied in Section 2.3, which can naturally deal with the exploration-exploitation trade off. Each decision is chosen by evaluating a reward function that is maximised with respect to the parameters of an unknown path. Another layer of BO is used to solve this maximisation step. This helps finding the best set of parameters that determine a continuous path where the robot travels on while taking samples. The proposed algorithm is validated on a large-scale environment for monitoring ozone concentration in the *United States of America* (USA), and on a mobile robot that monitors the dynamics of luminosity changes.

This chapter presents the following contributions:

1. Generalisation of the BO-Discrete-Planning algorithm (Algorithm 2) to optimise along continuous trajectories.
2. A layered BO approach for informative path planning in spatial–temporal environmental monitoring.

The remaining of this chapter are structured as follows. Section 4.2 presents GP regression for spatial–temporal modelling using a more complex model than in the previous chapter. Section 4.3 presents the theory for planning under continuous paths, which addresses path parametrisation and the decision making algorithm for finding the optimal path.Section 4.4 shows an experimental evaluation of the proposed algorithm, in a simulated environment and using a real robot. Finally, Section 4.6 presents a summary of the contributions and future work.

4.2 Spatial Temporal Modelling

A mathematical model of a phenomenon is very important for making correct policy decisions. Scientists have established physical and chemical laws that describe how environment phenomena behave and how relevant variables relate to each other. Although this deterministic approaches may be valid for controlled situations, they

might not be useful for most local environmental monitoring applications. For example, the spatial-temporal distribution of air- pollutant concentration in a city depends on traffic, temperature, humidity, wind speed, air pressure and building configuration among many other factors. There is no exact model that can capture the cross dependencies between all of these variables to achieve an accurate prediction of what the concentration will be in any given location. Also, air pollution does not only have spatial variations but also exhibits complex temporal patterns. Being able to model these patterns is of great importance to accurately predict the behaviour of the studied phenomenon.

In order to monitor spatial-temporal phenomena, it is necessary to learn a model of the unknown process to be monitored. A simple approach for modelling space-time dependent stochastic processes has been presented in Section 3.2, which has spatial and temporal dimensions coupled together. This model's main limitation is the capacity of capturing space and temporal specific patterns. This section presents a more complex approach for *Gaussian Process* (GP) spatial temporal modelling, where space and time components can be treated independently in the covariance function definition.

A latent noisy function $f(\mathbf{s};t)$ representing the realisation of a spatial–temporal environmental phenomenon is modelled as $y = f(\mathbf{s};t)+\epsilon$, where $\mathbf{s} \in \mathbb{R}^D$ are the coordinates in a spatial D-dimentional space, $t > 0 \in \mathbb{R}^+$ represents time and $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$ is the noise associated to each independent observation such that,

$$f(\mathbf{s},t) \,:\, \mathbb{R}^D \times \mathbb{R} \to \mathbb{R} \,. \tag{4.1}$$

A GP is a nonparametric Bayesian technique that places a prior distribution over the space of functions mapping inputs to outputs. In this chapter, the GP is fully determined by a spatial-temporal mean function $m(\mathbf{s};t)$ and a positive semi-definite spatial-temporal covariance function $k\left((\mathbf{s};t),(\mathbf{s};t)'\right)$, i.e.,

$$f(\mathbf{s};t) \sim \mathcal{GP}\left(m(\mathbf{s};t), k\left((\mathbf{s};t),(\mathbf{s};t)'\right)\right). \tag{4.2}$$

The main advantage with respect to the approach taken in Section 3.2 is that the mean and covariance functions can treat the spatial and temporal components independently. This allows individual covariance function assignment for the temporal and spatial component.

Denoting the set of the spatial temporal training sample locations from f as $X = \{(\mathbf{s};t)_i\}_{i=1}^N$ and its corresponding noisy outputs $\mathbf{y} = \{y_i\}_{i=1}^N$, the predictive distribution for a new query input $(\mathbf{s};t)^\star$ is Gaussian, $f^\star = f((\mathbf{s},t)^\star) \sim \mathcal{N}(\mathbb{E}[f^\star], \mathbb{V}[f^\star])$, where the mean and variance are given by:

$$\begin{aligned} \mathbb{E}[f^\star] = &\quad K((\mathbf{s};t)^\star, X)K_X^{-1}(\mathbf{y} - M(X)) \,, \\ \mathbb{V}[f^\star] = &\quad K((\mathbf{s};t)^\star, (\mathbf{s};t)^\star) - K((\mathbf{s};t)^\star, X)K_X^{-1}K(X, (\mathbf{s};\mathbf{t})^\star) \,. \end{aligned} \tag{4.3}$$

Here, $K(\cdot,\cdot)$ is the covariance matrix between all observations where each element (i,j) is $k((\mathbf{s};t)_i, (\mathbf{s};t)_j)$, $M(\cdot)$ is defined component wise, with $M(X)_{(i)} = m((\mathbf{s};t)_i)$ with $(\mathbf{s};t)_i \in X$ and $K_X = K(X,X) + \sigma_n I$. The main difference with the GP formulation presented in Section 2.2, is that input locations change from $\mathbf{x}$ to $(\mathbf{s};t)$.

The mean function, $m(\mathbf{s};t)$, depends on a set of hyper-parameters $\boldsymbol{\theta}_m$ and the covariance function $k\left((\mathbf{s};t), (\mathbf{s};t)'\right)$ depends on another set of hyper-parameters $\boldsymbol{\theta}_c$. A GP can adapt to any function f at two levels: The first level is choosing the expressions for m and k. The second level of adaptation is optimising the hyper-parameter set $\boldsymbol{\theta} = \{\boldsymbol{\theta}_m, \boldsymbol{\theta}_c, \sigma_n\}$, which corresponds to the learning component.

For simplicity but without loss of generality, we assume that the mean function is constant, $m(\mathbf{s};t) = \eta$, i.e. $\boldsymbol{\theta}_m = \{\eta\}$. An important decision is selecting the expression for the covariance function k, since there are several covariance functions in the literature with different characteristics. Selecting the covariance function is problem dependent, i.e. for a specific spatial-temporal phenomenon, certain expressions for k might be a better fit than others.

How to select the appropriate expression for the covariance function k? In the spatial-temporal domain, the first decision that has to be made is if space and time have independent behaviour or if they are coupled.

Independent behaviour means that the variation of f with time is independent of the spatial location, which is convenient because the full covariance function can be factorised into a spatial and temporal component. This kind of covariance functions, presented by Singh et al. [67], are defined as spatial-temporal *separable* covariance functions,

$$k_{sep}\left((\mathbf{s};t),(\mathbf{s};t)'|\boldsymbol{\theta}_c\right) = k_{\text{space}}(\mathbf{s},\mathbf{s}'|\boldsymbol{\theta}_{\text{space}})\, k_{\text{time}}(t,t'|\boldsymbol{\theta}_{\text{time}}). \tag{4.4}$$

The advantage of using separable covariance functions is that the expressions for the spatial and temporal components can be selected independently. For example, the spatial factor, k_{space}, might capture spatial dependency rate decay while the temporal factor, k_{time}, can capture a different covariance decay, not only with a different length-scale, but with a completely different covariance function expression that can include time-specific periodic behaviour.

Coupled space-time dependency means that the covariance structure for the temporal dimension depends on the spatial location or vice versa. In this case, the covariance function cannot be factorised into spatial and temporal specific components. Although this *non-separable* covariance functions can capture more complex dependencies, optimising their hyper-parameters becomes more challenging. The hyper-parameters of the covariance function are considered to be fixed over time.

The examples and experiments in this thesis use separable covariance functions by combining the covariance functions from Section 2.2.1. However, it is not restricted to any other kind of covariance function, such as non-stationary or non-separable. In addition, the planning algorithms can use any spatial-temporal regression technique, as long as it provides a posterior probability distribution.

4.3 Continuous Path Planning

This section shows how to plan for continuous paths. It builds on Section 3.3, and generalises the planning algorithm into a continuous path optimisation problem instead of waypoints. The proposed algorithm is a *Bayesian Optimisation* (BO) based

planner that uses the posterior *Probability Density Function* (PDF) provided by the spatial-temporal model to decide which is the best path to traverse in order accumulate most reward.

Firstly, we enumerate a possible path parametrisation, to later present the details of the layered BO planner that finds the near optimal path to travel.

4.3.1 Path Parametrisation

A path is a continuous curve $\mathcal{C}$ that is mathematically defined as a continuous function that maps from $u \in \mathbb{R}$ to $(\mathbf{s}, t) \in \mathbb{R}^{D+1}$.

$$\mathcal{C}(u|\boldsymbol{\beta}) \, : \, \mathbb{R} \rightarrow \mathbb{R}^{D+1} \tag{4.5}$$

where the output has $D+1$ dimensions with D dimensional spatial component $\mathbf{s}$ and time t. For example, when the dimension of the output space is two, i.e. $D = 2$, the paths are defined over a plane and $\mathcal{C}$ is parametrised by u. Different parametrisation functions can achieve the representation of a different family of curves by modifying the expression of $\mathcal{C}$. The output of $\mathcal{C}$ also depends on a set of curve parameters $\boldsymbol{\beta}$, where each parameter vector $\boldsymbol{\beta}$ represents a unique curve in the spatial-temporal output space. Border conditions may also apply when the initial or final poses are known, ultimately fixing some components of the parameter vector $\boldsymbol{\beta}$.

There are many different kinds of curve parametrisations, the most popular being piecewise polynomial parametrised by $u \in [0, 1]$ in each dimension of the output, which are called Splines. The order of the polynomials determines the type of spline. This thesis shows detailed expressions cubic splines, which are used in the experimental section. However, not that our planning algorithms are not limited to a specific parametrisation.

Cubic Splines

A cubic spline consists of third order polynomials for each spatial dimension. Considering $D = 2$, cubic splines are defined as:

$$\begin{aligned} \mathcal{C}(u|\boldsymbol{\beta}) \; : \; [0,1] &\rightarrow \mathbb{R}^3 \\ u \rightarrow s_1(u) &= a_1u^3 + b_1u^2 + c_1u + d_1 \qquad (4.6) \\ s_2(u) &= a_2u^3 + b_2u^2 + c_2u + d_2 \qquad (4.7) \\ t(u) &= \frac{1}{v_l}\int_0^u \mathrm{d}s \; , \qquad (4.8) \end{aligned}$$

where the parameter vector is $\boldsymbol{\beta} = \{a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2, v_l\}$. Here a_i, b_i, c_i and d_i control the shape of the spline while v_l is the linear speed, which determines the temporal component.

Known border conditions for the spline, such as initial position or pose, reduce the number of free parameters by fixing certain components in the parameter vector $\boldsymbol{\beta}$. Consider the following scenarios:

1. Known initial position;
   Useful when the starting location of the spline is fixed. A known initial position, $(s_1(0), s_2(0))$, translates into

   $$s_1(0) = d_1 \qquad (4.9)$$
   $$s_2(0) = d_2 \; , \qquad (4.10)$$

   leaving seven free parameters, $\boldsymbol{\beta} = \{a_1, b_1, c_1, a_2, b_2, c_2, v_l\}$.

2. Known initial pose;
   When the initial position, $(s_1(0), s_2(0))$, and the angle, α, of the spline are

known:

$$s_1(0) = d_1 \tag{4.11}$$

$$s_2(0) = d_2 \tag{4.12}$$

$$\left.\frac{\partial s_2/\partial u}{\partial s_1/\partial u}\right|_{u=0} = \frac{c_2}{c_1} = \tan\alpha \ , \tag{4.13}$$

leaving six free parameters, $\boldsymbol{\beta} = \{a_1, b_1, c_1, a_2, b_2, v_l\}$.

4.3.2 Continuous Path Planning

This section presents a planning algorithm for optimising paths to be traversed for monitoring a spatial-temporal phenomenon. The goal of the planning algorithm is to monitor efficiently a spatial-temporal phenomenon. Particularly, understanding more accurately the areas of interest. In the environmental monitoring case, this corresponds to areas of extreme values of the studied phenomenon. Thus, the problem can be framed as an optimisation problem, where the goal function is the phenomenon itself. *Bayesian Optimisation* (BO) can be used to solve the optimisation because it assumes an initially unknown goal function, quantifies uncertainty and guides the search of the optimum based its understanding of the phenomenon.

The pseudo-code of the continuous path planning algorithm (BO-Continuous-Planning) is presented in Algorithm 3. In general terms, this algorithm is essentially an adaptation of the BO algorithm, shown in Algorithm 1. It builds on Algorithm 2 proposed in Section 3.3 and generalises the approach to find continuous paths instead of discrete locations. It is an iterative algorithm, where each iteration produces a decision of where to go next. The decisions are taken based on the posterior *Probability Distribution Function* (PDF) provided by the *Gaussian Process* regression over f. For the case of continuous path planning, a decision corresponds to a set of curve parameters $\boldsymbol{\beta}$ that defines the curve a robot follows.

The BO-Continuous-Planning algorithm is an improvement over the previously proposed discrete planning algorithm (BO-Discrete-Planning) shown in Algorithm 2.

Algorithm 3 BO-Continuous-Planning

Inputs: f, h, GP-prior, $\mathbf{p}_{\text{init}}$
Outputs: GP-posterior, $\mathbf{p}$

```
p current pose of robot.
p ← p_init
while t < t_max do
    β* ← arg max_β r(C(u, β)|h)
    while p is far from C(1, β*) do
        Move along C
        Gather samples from f
        Update p
    end while
    Augment X and y with new samples.
    Update the GP model
end while
```

The main similarities and differences are the following:

Inputs: The inputs are the same as for the discrete version. f represents the phenomenon to be monitored. $h(\cdot)$ is the acquisition function that quantifies the benefit for sampling at a particular location (Section 2.3.1). The GP-prior is the statistical model used to represent f.

Outputs: Since the goal of the BO-Continuous-Planning algorithm is to monitor f, the output is a predictive model of f.

Reward Function: The reward function is associated with a decision. In Algorithm 2, the reward is evaluated at a particular location. Here, the reward function encodes the reward of travelling along a continuous path. Therefore its expression corresponds to the integral of the acquisition function $h(\cdot)$ over the path,

$$r(\mathcal{C}(u, \boldsymbol{\beta})|h) = \int_{\mathcal{C}(u,\boldsymbol{\beta})} h(v)\mathrm{d}v. \tag{4.14}$$

Sample Collection: Similar to Algorithm 2, a set of samples is collected while the robot travels along the selected curve $\mathcal{C}$ and the GP posterior is updated once the goal has been reached and a new path needs to be found.

For each iteration, the reward function is evaluated over multiple paths and depends on the acquisition function. For example, considering $h(v) = \text{UCB}$ (Equation 2.29), the reward function takes the following expression,

$$r(\mathcal{C}(u, \boldsymbol{\beta})|h) = \int_{\mathcal{C}(u,\boldsymbol{\beta})} h(v)\mathrm{d}v \ , \tag{4.15}$$

$$= \int_0^1 \text{UCB}(\mathcal{C}(u, \boldsymbol{\beta}))\|\mathcal{C}'(u, \boldsymbol{\beta})\|\mathrm{d}u \ , \tag{4.16}$$

$$= \int_0^1 \left[\mu(\mathcal{C}(u, \boldsymbol{\beta})) + \kappa\sigma(\mathcal{C}(u, \boldsymbol{\beta}))\right] \|\mathcal{C}'(u, \boldsymbol{\beta})\|\mathrm{d}u. \tag{4.17}$$

Where $\|\cdot\|$ is the magnitude of the derivative of the curve. This integral may not have an analytical solution depending on the definition of the acquisition and covariance functions. In this thesis the integral is approximated using a rectangle rule quadrature-based approximation [18], which generally results in accurate approximations for the one dimensional case (since the integral is over a 1-D variable, u).

The critical component of the Algorithm 3 is Line 4. Here, the optimal path parameter vector $\boldsymbol{\beta}^\star$ is found by maximising the reward with respect to all possible paths. The path $\mathcal{C}(u, \boldsymbol{\beta}^\star)$ is the one that cumulatively delivers the best reward by integrating over the acquisition function,

$$\boldsymbol{\beta}^\star = \arg\max_{\boldsymbol{\beta}} r(\mathcal{C}(u, \boldsymbol{\beta})|h) = \arg\max_{\boldsymbol{\beta}} \int_{\mathcal{C}(u,\boldsymbol{\beta})} h(v)\mathrm{d}v \ . \tag{4.18}$$

This maximisation step can be solved with any optimisation technique. In fact, for the plain BO algorithm and for the BO-Discrete-Planning algorithm shown in Algorithm 2 this optimisation step can be conducted using gradient-based local search optimisers. However, a global optimiser would greatly accelerate the convergence of the optimiser.

Alternatively, Equation 4.18 can be solved using another layer of BO. Since the action space $\boldsymbol{\beta}$ is at least five dimensional and the function r is highly non-convex and expensive to evaluate, BO provides a natural solution. The pseudo-code for finding the most informative path using double layered BO is presented in Algorithm 4. This means placing another GP prior over r and using a second acquisition function q to

Algorithm 4 Layered-BO-Continuous-Planning

Inputs: f, h, q, GP-prior, $\mathbf{p}_{\text{init}}$
Outputs: GP-posterior, p
1: $\mathbf{p}$ starting pose of robot.
2: $\mathbf{p} \leftarrow \mathbf{p}_{\text{init}}$
3: **while** $t < t_{\max}$ **do**
4: $\quad \boldsymbol{\beta}^\star \leftarrow \text{BO}(r(h), q)$ // Algorithm 1
5: $\quad$ **while** $\mathbf{p}$ is far from $\mathcal{C}(1, \boldsymbol{\beta}^\star)$ **do**
6: $\quad\quad$ Move along $\mathcal{C}$
7: $\quad\quad$ Gather samples from f
8: $\quad\quad$ Update $\mathbf{p}$
9: $\quad$ **end while**
10: $\quad$ Augment X and $\mathbf{y}$ with new samples.
11: $\quad$ Update GP model
12: **end while**

decide which path parameters $\boldsymbol{\beta}$ to evaluate over $r(h)$. This second layer of BO (Line 4 in Algorithm 4) follows the plain BO Algorithm 1. Note this step is fast to evaluate because it does not require the robot to move and gather training samples as it uses the existing GP model of the phenomenon.

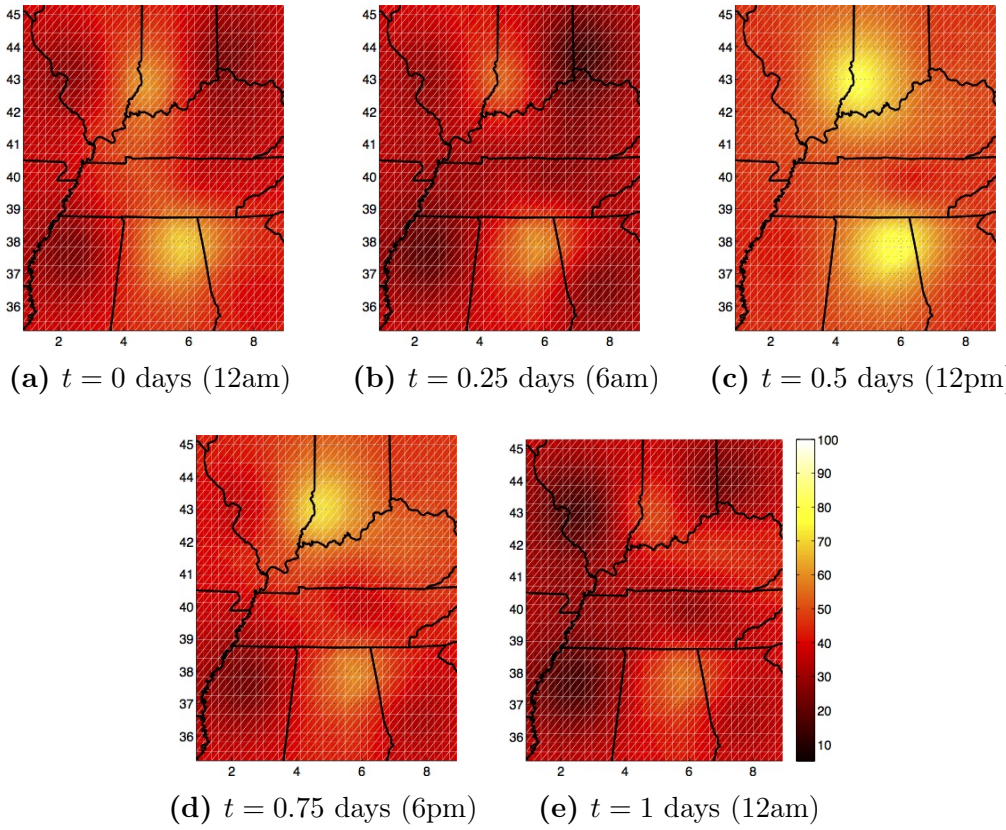
4.4 Experiments

In this section the proposed method is tested in two scenarios: a large-scale experiment for ozone monitoring in the US, and real-time monitoring of illumination with a mobile robot.

4.4.1 Large-scale Pollution Monitoring

The first experiment simulates an Unmanned Air Vehicle (UAV) monitoring ozone concentration; considered a pollutant at ground level. To simulate the environment we use real data provided by the US Environment Protection Agency[2]. A large number of ozone concentration measurements, dating back to 1987, are available with one

[2] `http://java.epa.gov/castnet/reportPage.do`



(a) $t = 0$ days (12am) **(b)** $t = 0.25$ days (6am) **(c)** $t = 0.5$ days (12pm)

(d) $t = 0.75$ days (6pm) **(e)** $t = 1$ days (12am)

Figure 4.1 – Ground-truth for ozone concentration across the USA, state limits shown in black. Area corresponds to Kentucky and Tennessee states. Axis are measured in 10^6 metres and corresponding to UTM coordinates for section 16F.

hour period for static sensing locations across the US. The UAV is forced to stay within the region specified in Figure 4.2.

The discrete data $\mathcal{S}$ from the database is used to create a simulated environment using GP regression, called *Ground Truth GP* (GTGP). A robot samples from the mean μ_{gt} of this continuous process in space and time. The GTGP uses a separable covariance function with an isotropic Matérn3 component for space and the sum of a Matérn5 and a Periodic component for time (See Table 2.1). The set of optimal hyper-parameters was found by maximising equation 2.13 using a gradient descent method. The optimal values found by the optimiser are: $\sigma_{fm3} = 1.862$, $l_{m3} = 1.195$, $\sigma_{fm5} = 0.201$, $l_{m5} = 5.84$, $\sigma_{fper} = 0.94$ and $\gamma = 5.75$. In order to deal with limited computational resources, the approximation parameter m, described in Section 2.2.3, is set to 300 for all experiments. The concentration of ozone changes periodically

Figure 4.2 – Area for the experiment (16F in UTM coordinates)

with a period of one day (known a priori), and the time is measured in days. Figure 4.1 shows a GTGP regression over space for five timestamps within one day. It can be noted that two peaks appear around mid-day with values that can reach up to 100ppb. The pattern is repeated every-day with slight variations in amplitude due to unknown environmental factors.

Ideally, the robot should accurately capture changes in the areas where pollution is more densely concentrated. We compare six different techniques for planning the motion of the UAV while monitoring the environment:

1. Random Discrete Sampling (RD): Randomly pick discrete goal locations within the environment.

2. Entropy Discrete Sampling (ED): Pick discrete locations for sampling using the maximum variance (entropy) criterium [22].

3. UCB Discrete Sampling (UCBD): Pick discrete locations using UCB [43].

4. Random Continuous Sampling (RC): Select random paths using an uniform distribution over $\boldsymbol{\beta}$.

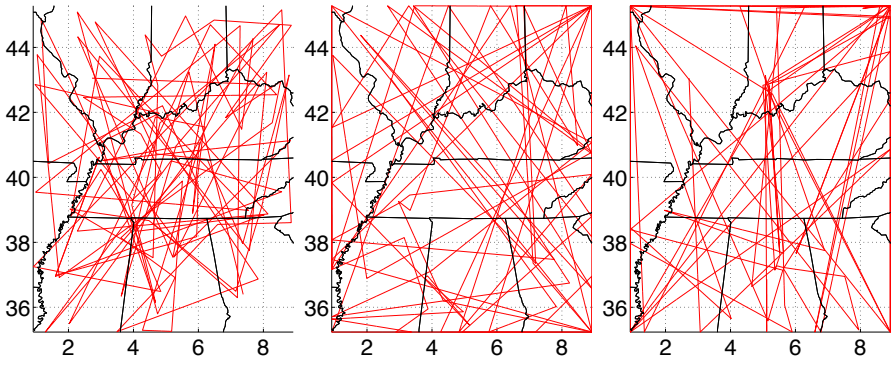
5. Entropy Continuous Sampling (EC): Find paths that maximise entropy reduction.

6. UCB Continuous Sampling (UCBC): Choose paths that maximise UCB acquisition function.

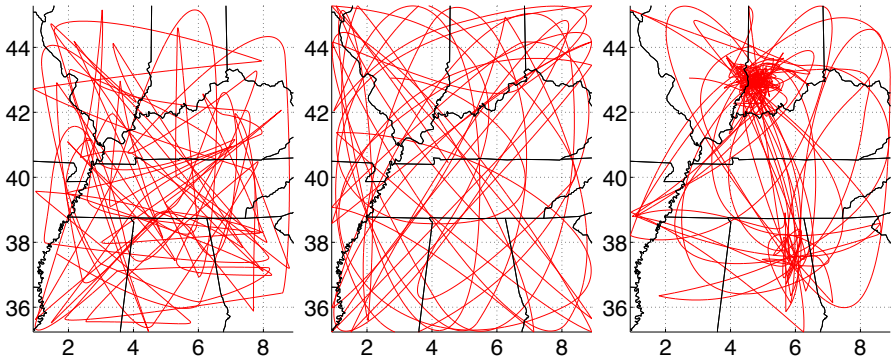
The proposed method is indicated in 6). UCBC and UCBD use the upper confidence bound acquisition function, with parameter $\kappa = 0.1$ manually tuned to balance the exploration-exploitation trade off. Although UCB was chosen due to its particular exploring behaviour [3], other acquisition functions can be used. Future work can compare different acquisition functions and learn the parameters of the acquisition function within the optimisation procedure. To include realistic sampling from the GTGP, random noise with $\sigma_n = 5$ is added to every sample independently. The experiment takes place for 30 days and assumes the vehicle moves at an average speed of $60km/h$. A sample of ozone concentration is collected every minute for all strategies, assuring that each method collects the same number of samples for predicting the values of the phenomenon. Given that all methods will acquire the same number of samples, the differences in error will only depend on the locations where the samples were acquired. The inner optimisation for maximising among paths (Line 4 of Algorithm 4) uses $q = UCB$ as acquisition function for strategies EC and UCBC. The GP model of the inner optimisation uses a Matérn3 (Table 2.1) covariance function whose hyper-parameters are optimised on each iteration using gradient decent.

Figure 4.3 shows the paths travelled by the robot for each case. A quick visual inspection shows that all methods were able to cover the region of interest and explore the entire environment. Random sampling strategies (RD and RC) do not present any interesting patterns and move chaotically across the studied area. Entropy based techniques (ED and EC) cover the region uniformly, reducing the uncertainty of the whole area. Finally, UCBD and UCBC concentrate their samples towards the areas of higher pollution.

A very important difference is the shape of paths for discrete and continuous sampling strategies. Even though κ has the same value for the acquisition function of UCBC and UCBD, the trajectories are much more concentrated over the high pollution areas for the continuous optimisation case (UCBC). The main reason is that this method



(a) RD **(b)** ED **(c)** UCBD



(d) RC **(e)** EC **(f)** UCBC

Figure 4.3 – Paths for different methods. Axis are measured in $10^6 m$.

takes into account the value of the acquisition function over the entire path that is being traversed. One way of seeing this is that if a method only takes into account a discrete goal location it will not necessarily collect useful information on its way to the target location. However, if the method does take into account the information gathered while reaching the target location, then the informativeness of gathered samples will increase noticeably.

We use four different error measures at M locations to evaluate the performance quantitatively:

1. *Root Mean Squared Error* (RMSE): Error without taking into account the value of the predicted variance.

2. *Weighted Root Mean Squared Error* (WRMSE): Places weights depending on the magnitude of the ground truth output, giving more importance to errors in higher pollution areas [43].

3. *Mean Log Loss* (MLL): Evaluates the negative log probability of the ground truth data point under the model. Takes into account not only the prediction error but also the associated uncertainty.

$$MLL = \frac{\frac{\sum_{i=1}^{M}\left(-\log p(\mu_{gt}(\mathbf{x}_i^\star)|\mathcal{S},\mathbf{x}_i^\star)\right)}{M}}{\frac{\sum_{i=1}^{M}\left(\frac{1}{2}\log\left(2\pi\sigma^2(\mathbf{x}_i^\star)\right)+\frac{(\mu_{gt}(\mathbf{x}_i^\star)-\mu(\mathbf{x}_i^\star))^2}{2\sigma^2(\mathbf{x}_i^\star)}\right)}{M}}. \tag{4.19}$$

4. *Weighted Mean Log Loss* (WMLL): Similar to WRMSE, but weighted over the mean log loss. Gives more importance to error in high pollution areas, taking into account the variance of the predictive model.

$$MLL = \frac{\sum_{i=1}^{M}\frac{\left(-\log p(\mu_{gt}(\mathbf{x}_i^\star)|\mathcal{S},\mathbf{x}_i^\star)\right)\left(\mu_{gt}(\mathbf{x}^*)-\min\mu_{gt}(\mathbf{x})\right)}{\max\mu_{gt}(\mathbf{x})-\min\mu_{gt}(\mathbf{x})}}{M} \tag{4.20}$$

with $\mathbf{x} = (\mathbf{s}; t)$ for the space-time case.

Table 4.1 shows the error for each method evaluated w.r.t. the ground truth on a grid over space and time for the entire duration of the experiment. It can be seen that the proposed method (UCBC) delivers the best performance for all indicators. UCBC favours areas of high pollutant concentration, achieving more accuracy over the areas that account for the most relevant component of error. The difference in performance between strategies is remarkable for the weighted errors WRMSE and WMLL. The main reason for this is the extra importance to model areas with high pollution (exploitative behaviour). For this experiment, UCBC also presents lower error for non weighted metrics, expected when the areas of interest account for the most important component of error. Therefore, when UCBC focuses on sampling from areas of higher concentration it will achieve lower error overall, compared to EC or RC that will model better areas that do not reduce the overall error importantly (because the output variable has lower values for non relevant areas).

Table 4.1 – Results for US Ozone Monitoring

Method	RMSE	WRMSE	LogLoss	WLogLoss
RD	7.3574	2.2324	3.4149	0.0956
ED	7.5225	2.2570	3.4332	0.0957
UCBD	7.1579	1.9764	3.3999	0.0937
RC	7.2238	2.0698	3.3951	0.0935
EC	7.1103	2.2958	3.3907	0.0956
UCBC	**6.7971**	**1.4981**	**3.3537**	**0.0863**

It is also noticeable the difference between continuous and discrete sampling strategies. An improvement is revealed for all strategies as we are optimising over continuous paths rather than choosing discrete locations.

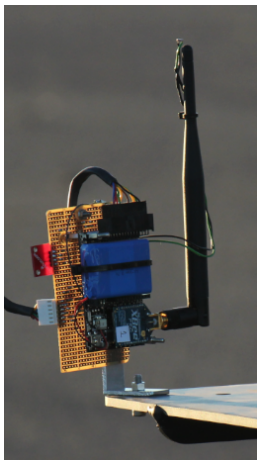
4.4.2 Luminosity Monitoring

A small, wheeled mobile robot was used to monitor dynamic illumination changes in an indoor environment. The goal of this experiment is to compare different techniques for path planning and their impact on the abilities of the robot to learn the space-time patterns of a dynamic phenomenon. The idea is to create a real-world phenomenon under a controlled environment where the dynamics can be adjusted accordingly. Two light sources with variable intensity are dimmed electronically to expose patters with a periodic component and amplitude changes through time.

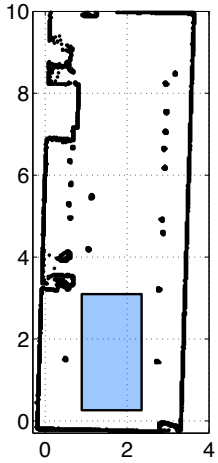
The robot is equipped with an on-board CPU running ROS[3], an environmental sensing electronic board, shown in Figure 4.4b, and a laser scanner for localisation in an previously built map. Samples from the phenomenon are gathered every one second. Ground truth is obtained by placing static sensor boards in five static locations, shown in Figure 4.4d. Figure 4.5 shows an interpolation of the measurements in these locations over space for five time stamps within a period. Figure 4.6 shows the interpolation over time for one source of light located at $(x, y) = (1.12, 2.39)$. Variations

[3]Robot Operating System `http://www.ros.org`

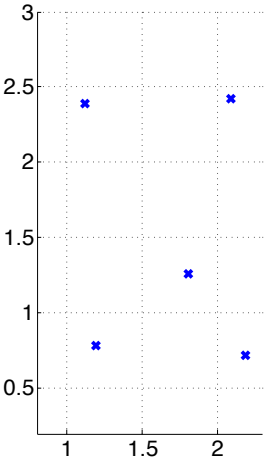
(a) Mobile robot used in the experiments.



(b) Sensing Board.



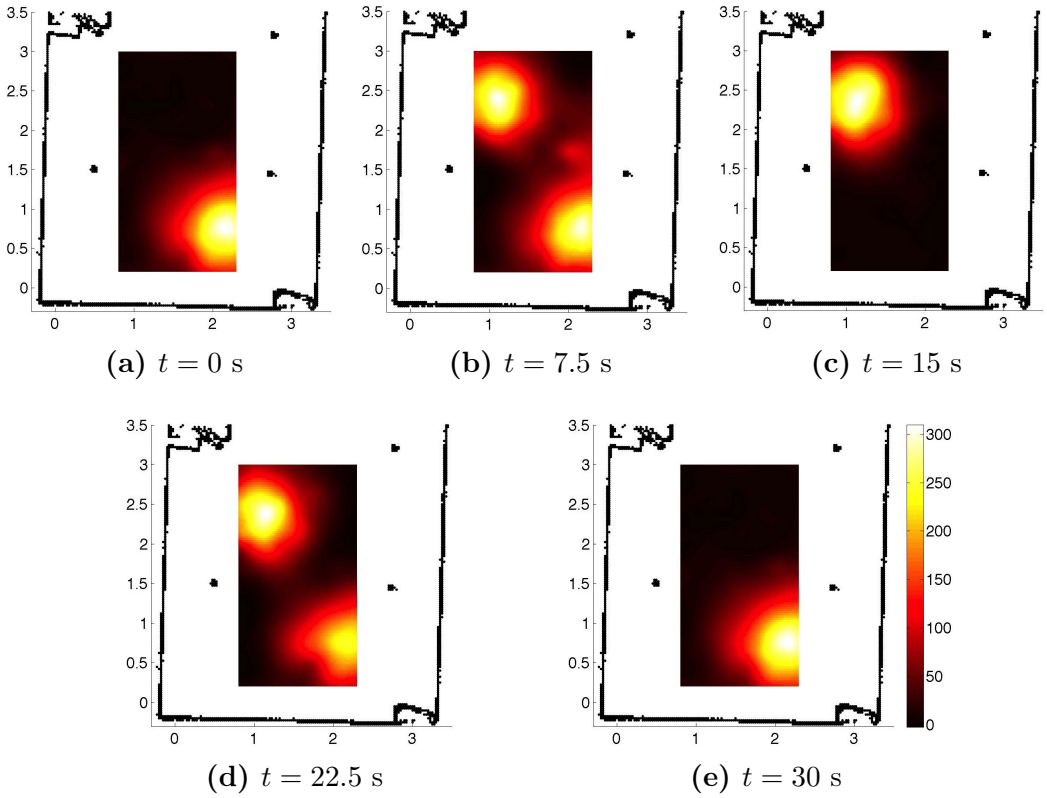
(c) Experiment area.



(d) Position of sensing nodes for ground truth.

Figure 4.4 – Sensing board, map of the area and location of ground truth measurements. Axis in metres.

in amplitude are noticeable over time and similar to many natural phenomena. Even though the light sources are easily distinguishable for a human observer, the problem is much more complex for a robot that gathers noisy samples from the unknown time-changing phenomenon. The problem becomes even more interesting when the robot needs to decide where to take next samples based on past experience and future reward.



(a) $t = 0$ s **(b)** $t = 7.5$ s **(c)** $t = 15$ s

(d) $t = 22.5$ s **(e)** $t = 30$ s

Figure 4.5 – Spatially distributed light intensity variations. Axis in metres.

Table 4.2 – Hyper-parameters For Luminosity Monitoring

σ_n	η	σ_{f1}	l_1	σ_{f2}	l_2	σ_{f3}	γ	σ_{f4}	l_4
12.0	85.0	0.59	29	11.9	8.2	11.2	298	7.6	2

The same path-planning strategies in section 4.4.1 are compared in experimental trials that last for ten minutes. The robot used the following covariance function for building the GP model of the phenomenon:

$$\begin{aligned} &k_{sep}\left((\mathbf{s};t),(\mathbf{s};t)'|\boldsymbol{\theta}\right) = k1_{mat3}(\mathbf{s},\mathbf{s}'|\boldsymbol{\theta_1})\cdot \\ &\left[(k2_{mat3}(t,t'|\boldsymbol{\theta_2}))\cdot k3_p(t,t'|\boldsymbol{\theta_3})) + k4_{mat3}(t,t'|\boldsymbol{\theta_4})\right], \end{aligned} \tag{4.21}$$

where the estimated hyper-parameters $\boldsymbol{\theta}$ are shown in Table 4.2.

Figure 4.7 shows the paths travelled by the robot using each technique. Results are similar to the experiment in the previous section. While random sampling strategies, RD and RC, derive paths mostly concentrated at the centre of the studied region,

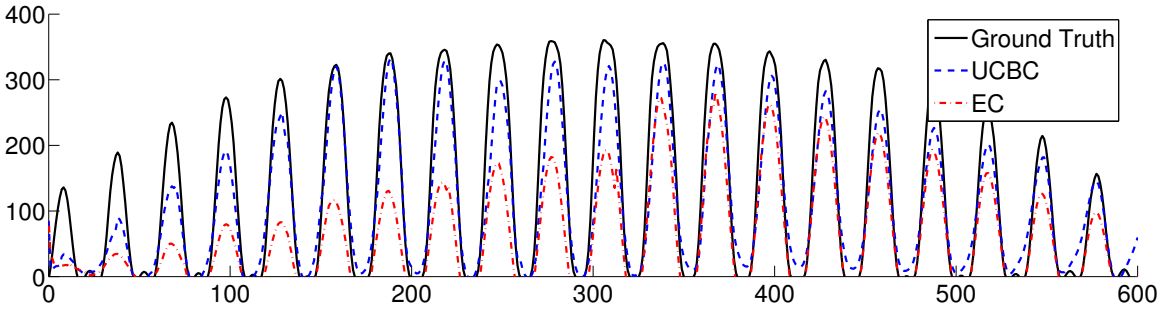


Figure 4.6 – Light intensity oscillation at location $(x, y) = (1.12, 2.39)$. Horizontal axis represents time in seconds and vertical axis represents intensity with no SI units. The mean of prediction for UCBC and EC is shown according to the legend.

paths for entropy based strategies, ED and EC, are distributed more homogeneously over space. In constrast, UCB paths focus on areas with high luminosity while at the same time exploring the environment for unknown sources of light.

Table 4.3 shows numerical results for the evaluation of the performance indicators described in section 4.4.1. Random policies perform close to entropy techniques for this obstacle-free environment. In a case of extremely low CPU availability it can be considered as a viable alternative; however, in real complex environments it is not a promising candidate. UCBC delivers the lowest error and weighted error for all the indicators. It is also shown that sampling over continuous domains results in smaller error for the case of UCB acquisition function. UCB strategies have the smallest error, demonstrating a central advantage in monitoring dynamic phenomena: monitoring areas of higher pollution more intensively results on lower overall error.

The developed method runs close to real-time in a standard CPU (i5 processor). Each iteration for finding the next optimal path takes $4.8s$ in average and can be computed before the execution of the current path is finished, avoiding unwanted pause between consecutive paths.

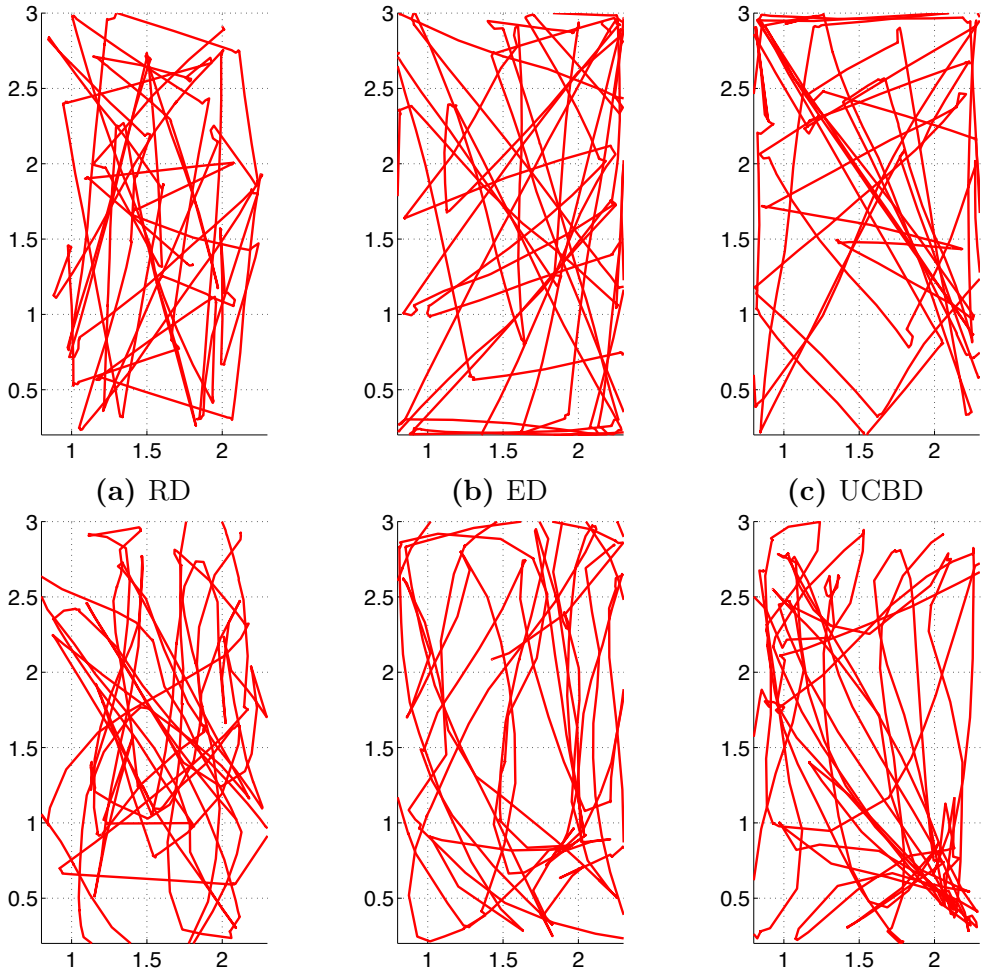


Figure 4.7 – Resulting paths for six different path planning techniques, axis are in metres.

4.5 Related Work

Decision making under uncertainty has seen significant developments as well. Information gain strategies for placing static sensors were studied in Guestrin et al. [22]. Mobile sensing agents can use active learning to choose where to sample from the environment. For this purpose, Markov Decision Processes and Reinforcement Learning approaches have been used by Bush et al. [5] and Chung et al. [7]. Bush et al. [5] uses belief-state MDPs for selecting observations that minimise uncertainty

Table 4.3 – Results for Luminosity Monitoring

Method	RMSE	WRMSE	LogLoss	WLogLoss
RD	39.598	28.061	8.949	2.327
ED	38.389	25.106	10.013	2.779
UCBD	38.210	23.715	9.685	2.672
RC	43.390	27.045	10.197	2.754
EC	48.873	38.433	11.980	3.443
UCBC	**30.121**	**21.422**	**8.098**	**2.145**

and Chung et al. [7] uses GPs for modelling the state-action value function. The main limitation of these approaches is the limited action space and tractability to deal with real scenarios. Uncertainty-driven planning was also explored by Hollinger et al. [27] using the travel salesman problem and RRTs LaValle [35] to find paths. RRTs are also explored by Lan and Schwager [34], where a record the minimum cost cycle is considered to find cyclic trajectories. Hollinger and Sukhatme [26] combines Rapidly Exploring Random Graphs (RRGs) and Branch and Bound optimisation techniques to find informative paths. An optimisation approach has been explored by Witt and Dunbabin [77], that uses simulated annealing and swarm optimisation for planning energy-optimal paths for an AUV under strong currents. The main drawback of this work is that uncertainty is not estimated by the model used to derive decisions. Another cost aware path planner was presented in Suh and Oh [71]. It assumes an already known cost map and is not useful for exploration purposes.

Recent decision making algorithms make use of submodularity properties for planning non-myopic, long-term way points for uncertainty reduction [1, 22, 31, 65]. These methods provide convergence guarantees and error bounds based on an exploration-only behaviour. While minimising the overall uncertainty of the model is important in some applications, for most of pollution monitoring tasks this is not sufficient. In such cases, it is desirable to be more accurate in areas of high pollution than in areas of low pollution. This introduces extra terms in the objective function (such as the mean of the predicted pollution concentration) making the submodularity assumption

invalid.

The proposed method has the following advantages over the previous techniques:

i) It is not a waypoint greedy solution to acquiring new observations as it takes into account measurements obtained along a path with predictions propagated over time.

ii) It considers a continuous action space.

iii) It uses both the mean and the variance to define paths and addresses the exploration-exploitation trade off in a principled Bayesian framework.

4.6 Summary

This chapter proposed a new technique for informative path planning over continuous paths for environmental monitoring. The main contribution is the derivation of a continuous action space strategy by integrating over an acquisition function in a principled Bayesian optimisation framework. This chapter presents a model for space-time phenomena using Gaussian processes which enables a robot to learn periodic patterns while preserving spatial correlations between observations. A first layer of BO is used to predict regions of high concentration. Then, a second layer of BO is used to estimate the curve parameters defining the best path to collect new observations.

The proposed method was evaluated in two experimental settings: on a large-scale autonomous monitoring problem for ozone concentration in the US, and for real-time monitoring of changes in luminosity indoors. In both cases, the mobile robot was able to learn a space-time model of the dynamic phenomena. Comparisons were performed between existing techniques for informative path planning indicating that the proposed algorithm captures more accurately the dynamics of areas where the monitored quantity has higher concentration. This ultimately results in an overall more accurate model with lower weighted error. Additionally, our technique can even

reduce the total RMSE if the areas of interest account for a significant proportion of the error, as is the case for the two studied situations.

We believe that optimising over curves for path planning can produce more informative decisions achieving longer term rewards. The method explained in this chapter can significantly improve the decision making process for efficiently monitoring a wide variety of environmental phenomena. In the next chapter we address the long-term or infinite horizon planning with continuous POMDPs and derive a joint procedure possessing the advantages of both Bayesian optimisation and POMDPs.

Chapter 5

Sequential Continuous Planning

5.1 Introduction

Thus far, the proposed algorithms for planning have focused on one step lookahead decisions. The approaches focused on myopic decisions without giving special consideration to the sampling sequence, or the costs or constraints associated with that sequence. However, in real-world sequential decision problems such as in robotics, the order in which samples are gathered is paramount, especially when the robot needs to optimise a temporally non-stationary objective function. Additionally, the state of the environment and sensing platform determine the type and cost of samples that can be gathered. To address these issues, we formulate *Sequential Bayesian Optimisation* (SBO) with side-state information within a *Partially Observed Markov Decision Process* (POMDP) framework that can accommodate discrete and continuous observations. The proposed solution involves *Monte-Carlo Tree Search* (MCTS) and *Upper Confidence bound for Trees* (UCT) for POMDPs and is extended to work with continuous state and observation domains. Through a series of experiments on monitoring a spatial-temporal process with a mobile robot, the results show that the proposed UCT-based SBO POMDP optimisation outperforms myopic and non-

myopic alternatives. This work has been previously published at UAI[1].

Bayesian Optimisation (BO); described in Section 2.3; is a global optimisation technique that has recently gained popularity in the machine learning community. BO possesses major advantages when used to find the maximum of partially observed objective functions that are costly to evaluate, lack gradient information, and can only be inferred indirectly from noisy observations. BO is robust to this setting because it builds a statistical model over the objective. More specifically, it places a *prior* over the space of functions and combines it with noisy samples to produce an incremental prediction for the unknown function. The prior usually takes the form of a *Gaussian Process* (GP), which has proved successful in modelling spatial-temporal data. The key component for the effectiveness of BO is the use of an *Acquisition Function* (AF) that guides the search for the optimum by selecting the locations where samples are gathered based on the posterior in each iteration.

BO can be readily applied to scenarios where the objective function does not vary in time and sampling locations can be chosen freely within the input domain. In real-world robotics applications, functions are likely to change with time indicating that *when* to sample is as important as *where* to sample. Another important aspect in realistic settings is that the state of the environment and sampling platform determines the reachable space for gathering the next sample. Combined, these issues create an imperative for finding optimal *sequences* of sampling locations.

The algorithms proposed in previous chapters focus on myopic decision-making by evaluating one-step lookahead for objective sampling. Non-myopic solutions have been proposed in [19, 54], but the authors acknowledge they are considerably expensive to evaluate and do not account for possible side-state presence due to external conditions. Lim et al. [39] propose solving the informative path planning problem using policy trees, however, they do not take into account time changing phenomena or continuous paths, decreasing the complexity of the problem. An optimal solution to non-myopic decision-making with side- state can be formalised in the *Partially Ob-*

[1]Roman Marchant, Fabio Ramos, and Scott Sanner. Sequential Bayesian Optimisation for Spatial-Temporal Monitoring. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2014

served Markov Decision Process (POMDP) framework. The key here is to consider the state as a tuple, consisting of the unknown function and the state of a sensing robot. However, this leaves open the question of how one can efficiently solve the resulting POMDP.

This chapter presents the following contributions:

1. Defines *Sequential Bayesian Optimisation* (SBO), which considers a sequence of generic decisions (discrete or continuous action space).
2. Formulates SBO under the POMDP framework.
3. Develops *Monte-Carlo Tree Search* (MCTS) and *Upper Confidence bound for Trees* (UCT) to solve the POMDP version of SBO.

The remaining of this chapter is structured as follows. Section 5.2 defines SBO, which is formulated under the POMDP framework in Section 5.3. Section 5.4 proposes a way to find an online solution for the multi-step lookahead SBO that aims to provide an optimal sequence of sampling locations. Section 5.5 evaluates the proposed model for spatial-temporal monitoring problems that clearly demonstrate the benefits of our UCT algorithm for non-myopic SBO optimisation.

5.2 Sequential Bayesian Optimisation

With the definitions above we can now extend BO to a sequential setting. In order to apply BO to more realistic problems we expand the existing theory to a more generic framework and include the notion of state in the definition of the problem. The main modification is that the acquisition function is replaced by a generic reward function, r. This reward depends on the state $\mathbf{x}_i$ of a mobile robotic sensor and the expected value of the objective function $f(\mathbf{x}_i)$, which using simplified notation, is noted as f_i. In the general case, because gathering each sample has an associated reward, the order in which they are gathered has a direct influence over the total accumulated

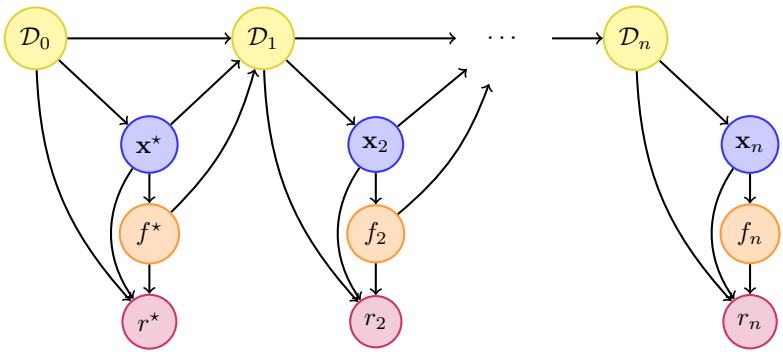


Figure 5.1 – Bayesian network representation for SBO.

reward for a specific lookahead. We call this kind of optimisation technique *Sequential Bayesian Optimisation* (SBO).

Sampling locations and their associated observations are grouped in $\mathcal{D}$, which contains all the history of samples $\{x_i, f_i\}$ and is built incrementally as shown in Figure 5.1. Using a similar treatment to plain BO, the myopic expectation of the reward r (ER), can be obtained by marginalising out all unknown outcomes,

$$\begin{aligned} \mathrm{ER}(\mathbf{x}^\star|\mathcal{D}_0) &= \mathbb{E}_{f^\star}\left[r(\mathbf{x}^\star, f^\star|\mathcal{D}_0)\right] && (5.1)\\ &= \int r(\mathbf{x}^\star, f^\star|\mathcal{D}_0)p(f^\star|\mathbf{x}^\star, \mathcal{D}_0)\mathrm{d}f^\star \ . && (5.2)\end{aligned}$$

This expression represents the first section of the Bayesian network shown in Figure 5.1. It represents the expected reward only based on the immediate reward $r^\star$. The n-step lookahead expression considers the whole reward sequence, from $r^\star$ to r_n and is given by

$$\begin{aligned} &\mathrm{ER}_n(\mathbf{x}^\star|\mathcal{D}_0) \\ &= \int \cdots \int \left(r(\mathbf{x}^\star, f^\star|\mathcal{D}_0) + \sum_{i=2}^{n} \left(r(\mathbf{x}_i, f_i|\mathcal{D}_{i-1})\right)\right) \\ &\qquad p(f^\star|\mathbf{x}^\star, \mathcal{D}_0) \times \prod_{i=2}^{n} p(f_i|\mathbf{x}_i)p(\mathbf{x}_i|\mathcal{D}_{i-1}) \\ &\qquad\qquad \mathrm{d}f^\star \mathrm{d}f_2 \cdots \mathrm{d}f_n \mathrm{d}\mathbf{x}_2 \cdots \mathrm{d}\mathbf{x}_n \ , \end{aligned} \tag{5.3}$$

where we are marginalising out all future outcomes $(f^\star, f_2 \ldots f_n)$ and locations $(\mathbf{x}_2 \ldots \mathbf{x}_n)$.

This expression has been derived by Osborne et al. [54], however, it is modified here to consider the whole sequence of locations for the reward calculation, not just the expected improvement for the last observation. It is important to note that within the BO algorithm, ER can be seen as the acquisition function $h(\mathbf{x})$ for selecting sampling locations. ER needs to be maximised w.r.t. $\mathbf{x}^\star$ in each iteration of the algorithm.

In real robotic deployments, decisions $\{\mathbf{x}_i\}$ can be represented as continuous paths followed by the robot. We represent these paths as parametrised curves, $\mathcal{C}$, over the input space, with each curve characterised by a set of parameters $\boldsymbol{\beta}$ (See Section 4.3.1). The following expression shows the expected reward for traversing a path with parameters $\boldsymbol{\beta}^\star$, and looking ahead for n steps, i.e. considering n paths in the future and integrating all possible rewards,

$$\begin{aligned}
&\mathrm{ER}_n(\boldsymbol{\beta}^\star, \mathcal{D}_0) \\
&= \int_{f^\star}\int_{f_2}\cdots\int_{f_n}\int_{\boldsymbol{\beta}_2}\cdots\int_{\boldsymbol{\beta}_n} \\
&\qquad \left(r(\mathcal{C}_{\boldsymbol{\beta}^\star}|\mathcal{D}_{N-1}) + \sum_{i=2}^{n} r(\mathcal{C}_{\boldsymbol{\beta}_i}|\mathcal{D}_{i-1})\right) \\
&\qquad p(f^\star|\boldsymbol{\beta}^\star, \mathcal{D}_{N-1}) \prod_{i=2}^{n} p(f_i|\boldsymbol{\beta}_i, \mathcal{D}_{i-1}) p(\boldsymbol{\beta}_i|\mathcal{D}_{i-1}) \\
&\qquad\qquad \mathrm{d}f^\star \mathrm{d}f_2 \cdots \mathrm{d}f_n \mathrm{d}\boldsymbol{\beta}_2 \cdots \mathrm{d}\boldsymbol{\beta}_n \ .
\end{aligned} \tag{5.4}$$

In this expression we are marginalising out all possible objective function expectations and paths for n steps. Unfortunately, given the infinite number of possible paths, this integral does not have an analytical solution and can only be approximated. In the following section we illustrate how SBO can be represented in a POMDP formulation and solved using online decision making POMDP solvers.

5.3 SBO as Online POMDPs

Our SBO formulation is *state-aware*, i.e. it considers the state of a mobile robot for decision making. This problem can be formulated as a POMDP problem in a similar manner as described by Toussaint [73] for regular BO. The main idea is to include

the objective function, which is partially observable, together with the state of the robot, into the state definition. We assume the robot's pose is fully observable and part of the state as side information $\mathbf{p}$. The decision of where to sample f is encoded by the action space, which is limited by the possible actions that can be performed by the robot. In the discrete case, an action is represented by moving to a specific cell. For the continuous case an action means travelling along a continuous path. More formally, the elements of the POMDP definition for side-state SBO are:

- S : The state, which is a tuple $\{f, \mathbf{p}\}$, where f is a latent (not directly observable) function defined over space and time representing the unknown spatial-temporal process. Additionally, we include the state of the sensing robot, $\mathbf{p}$, which is fully observable, as the side information.

- A : The parametrised action space $a(\boldsymbol{\beta})$. The actions can be described as move according to parameters $\boldsymbol{\beta}$ and gather samples from f in the process. In the continuous case, $\boldsymbol{\beta}$ are the parameters of a continuous curve defined over the domain of f.

- T : The transition function which is defined over the entire state $\{f, \mathbf{p}\}$. $T(\{f, \mathbf{p}\}, a(\boldsymbol{\beta}), \{f', \mathbf{p}'\})$ is the transition probability of resulting in state $\{f', \mathbf{p}'\}$ given that action $a(\boldsymbol{\beta})$ was taken at state $\{f, \mathbf{p}\}$. Assuming that the robot does not affect or change the objective function, the joint transition probability can be decomposed into the product of two independent transition functions:

$$\begin{aligned} &T(\{f, \mathbf{p}\}, a(\boldsymbol{\beta}), \{f', \mathbf{p}'\}) \\ &\quad = T_f(f, a(\boldsymbol{\beta}), f') T_{\mathbf{p}}(\mathbf{p}, a(\boldsymbol{\beta}), \mathbf{p}') \ . \end{aligned} \tag{5.5}$$

  Since f is not affected by the actions in A, the transition function T_f is the identity,

$$T_f(f, a(\boldsymbol{\beta}), f') = \mathbb{1}(f' - f) \ , \tag{5.6}$$

  i.e. equal to one only when $f' = f$. The transition function $T_{\mathbf{p}}$ depends on the definition of the action space, and can often be modelled deterministically since

robots can navigate with accurate positioning and path following controllers in many large-scale outdoor monitoring applications. When the action space is defined as a location, the action parameters $\boldsymbol{\beta}$ represent a location, and $T_{\mathbf{p}}$ can be calculated using,

$$T_{\mathbf{p}}(\mathbf{p}, a(\boldsymbol{\beta}), \mathbf{p}') = \delta(\mathbf{p}' - \boldsymbol{\beta}) \ . \tag{5.7}$$

- R : If the objective function f is sampled at a discrete location $\boldsymbol{\beta}$ then the *expected* reward in an SBO POMDP belief state is the reward value w.r.t. beliefs $b(f)$ minus any application-specific action $cost(\mathbf{p}, \boldsymbol{\beta})$ associated with movement according to $\boldsymbol{\beta}$:

$$\mathrm{ER}(\{f, \mathbf{p}\}, a(\boldsymbol{\beta})) = \mathbb{E}_{b(f)}[r(\boldsymbol{\beta})] + cost(\mathbf{p}, a(\boldsymbol{\beta})) \ , \tag{5.8}$$

where r is the generic reward which corresponds to the acquisition function in plain BO. When the action space is parametrised by curves, the reward associated with an action is given by the sum of the rewards along the curve $\mathcal{C}$:

$$\mathrm{ER}(\{f, \mathbf{p}\}, \mathcal{C}(\boldsymbol{\beta})) = \sum_{\mathbf{x}_i \in \mathcal{C}(\boldsymbol{\beta})} \mathrm{ER}(\{f, \mathbf{p}\}, \mathbf{x}_i) \ , \tag{5.9}$$

where the sum can be replaced by an integral when the sensing device allows continuous sampling along the curve.

- Z : In SBO, objective observations $z \in \mathbb{R}$ are simply noisy observations of $f(\boldsymbol{\beta})$ as defined next.

- O : The observation function is defined according to the action space parametrisation. When the action space is defined as a sampling location $\boldsymbol{\beta}$, f can be evaluated directly on $\boldsymbol{\beta}$,

$$O(z, a(\boldsymbol{\beta}), \{f, \mathbf{p}\}) = p(z|f(\boldsymbol{\beta})) \ . \tag{5.10}$$

We observe that for GPs, we can generate z by sampling from a GP posterior for f at location $\boldsymbol{\beta}$. When the action space is a curve $\mathcal{C}$, f is evaluated at a

number of sample locations within $\mathcal{C}$. The observation function for this set of observations $\{z_i\}$ is

$$O(\{z_i\}, \mathcal{C}(\boldsymbol{\beta}), \{f, \mathbf{p}\}) = \prod_{\mathbf{x}_i \in \mathcal{C}(\boldsymbol{\beta})} p(z_i | f(\mathbf{x}_i)) \ . \tag{5.11}$$

The belief is then the probability distribution over the space of functions f and updated as described in Equation 2.38. If the model for f is a GP, the belief update for an action-observation pair can be computed directly. The action component can be ignored for purposes of updating a belief in f, since as stated earlier, the robot's physical state does not affect or change the objective function; it only restricts the observations that can be made regarding f. Therefore, the belief update over f is simply computed by adding new location-observation pairs to the GP training data set.

While Martinez-Cantin et al. [47] use BO to find the policy parameters, we present a methodology to solve this POMDP by sampling a subset of action primitives that the robot can execute in the environment. Action primitives and maximum likelihood observation selection are the key points to approximate Equation 5.4.

5.4 MCTS and UCT for solving SBO

Monte Carlo Tree Search (MCTS) is a popular technique for solving large POMDPs [4, 64]. This method can turn a exhaustive search in decision trees into an efficient approximation using Monte-Carlo samples from the tree. MCTS efficiently searches reachable beliefs from a given initial belief state and is useful for real-time online planning.

Silver and Veness [64] have shown how MCTS can reach impressive scalability through the use of *Upper Confidence Bound for Trees* (UCT), which they call POMCP. In this work we conserve their idea of efficient tree search. However, we consider the case where the belief update is a GP update for f and use the maximum likelihood

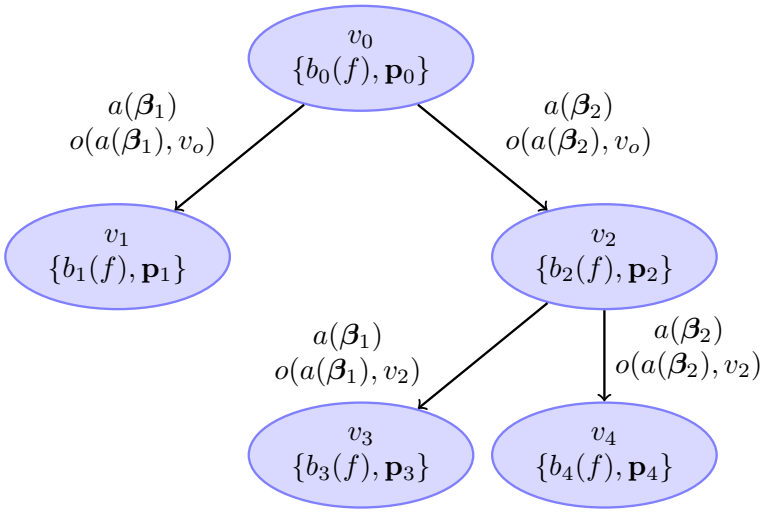


Figure 5.2 – Example of a tree with depth 2, partially expanded from a set of two action primitives.

observation, as in Platt et al. [55] and Martinez-Cantin et al. [47]. The maximum-likelihood observation assumption helps reducing the branching factor of the tree, which would grow uncontrollably when sampling observations.

For the SBO problem, each node in the tree consists of a belief representation for f and a side-state $\mathbf{p}$. We define the ith node by v_i. For each action-observation pair, the belief representation $b(f)$ and side state $\mathbf{p}$ are updated easily since $b(f)$ is a GP prior and side-state transitions are deterministic and observable. Every new action-observation simulation creates a new node with the updated belief and side-state.

The action space is discretised into a fixed set of action primitives. The tree is built incrementally starting with an initial node v_0. Figure 5.2 shows an example of a small tree that has been expanded partially with two action primitives. Each ellipse represents a node, that consists of a belief over f, $b(f)$, and side-state $\mathbf{p}$. A node is expanded by simulating the outcomes of executing an action. The outcomes (noisy observations of f) are the maximum- likelihood observations. The branching factor of the tree will be the number of action primitives. When a node is expanded, a new node is created using the updated belief and new side state.

The first step in each iteration is to find a leaf node candidate for expansion/evalu-

Algorithm 5 Monte Carlo Tree Search for SBO

```
function a* = MCTS(b(f), p, depth_max)
    v_0 = NewNode(b(f), p, reward_min)
    i ← 0
    while i < {Max MCTS iterations} do
        v_l ← TreePolicy(v_0)
        r ← DefaultPolicy(v_l)
        Backup(v_l, r)
    end while
    return a* = BestChild(v_0)
end function
function v_l = TreePolicy(a)
    v ← v_0
    while Depth(v) ≤ depth_max do
        if v has untried actions then
            Choose a from untried actions
            r ← Simulate a                          ▷ Simulate Reward
            Update b(f) and p.
            return v_l =NewNode(b(f), p, r)
        else
            v = BestChild(v)
        end if
    end while
    return v
end function
```

ation, which is done inside of the function TreePolicy. This search is guided by the function BestChild, which uses the statistics stored for each node (accumulated reward and number of visits) to select the most *promising* child. Starting from the chosen leaf node, a random action selection is conducted until the maximum depth is reached, executed within DefaultPolicy. The total accumulated reward is then backed up in function BackUp, that updates the statistics on all the nodes visited during the current iteration. Each iteration of the search algorithm simulates a sequence of up to n actions, where n is the maximum depth. When the iteration loop is completed, the best action is determined by picking the best child from the parent node v_0. Algorithm 5 shows the full procedure for building a tree and returning the best immediate action.

Algorithm 6 Monte Carlo Tree Search for SBO Part 2

function $r = \textsc{DefaultPolicy}(v_l)$
  $r \leftarrow$ Get reward accumulated until v_l
  $d \leftarrow \textsc{Depth}(v_l)$
  while $d \leq \text{depth}_{max}$ **do**
    Select a randomly
    Update $b(f)$ and $\mathbf{p}$.
    $r_a \leftarrow$ Simulate a
    $r \leftarrow r + r_a$
    $d \leftarrow d + 1$
  end while
end function
function $\textsc{Backup}(v_l, r)$
  $v \leftarrow v_l$
  while $v \neq v_0$ **do**
    Increase visited counter for v
    Increase accumulated reward for v
    $v \leftarrow \textsc{Parent}(v)$
  end while
end function
function $v_c = \textsc{BestChild}(v_p)$
  $V \leftarrow$ Children of v_p
  for $v_i \in V$ **do**
    $N_p \leftarrow$ Visited counter of v_p
    $N_i \leftarrow$ Visited counter of v_i
    $R_i \leftarrow$ Accumulated reward
    $g(i) = \frac{R_i}{N_i} + \kappa_{MC}\sqrt{\frac{2\ln(N_p)}{N_i}}$
  end for
  $v_c \leftarrow \arg\max_{v_i \in V} g(i)$
end function

5.5 Experiments

In this section we present experiments where a robot attempts to learn the behaviour of a spatial-temporal process by choosing actions that maximise the expected reward. We show comparisons for two different problems, including one with time dependent behaviour.

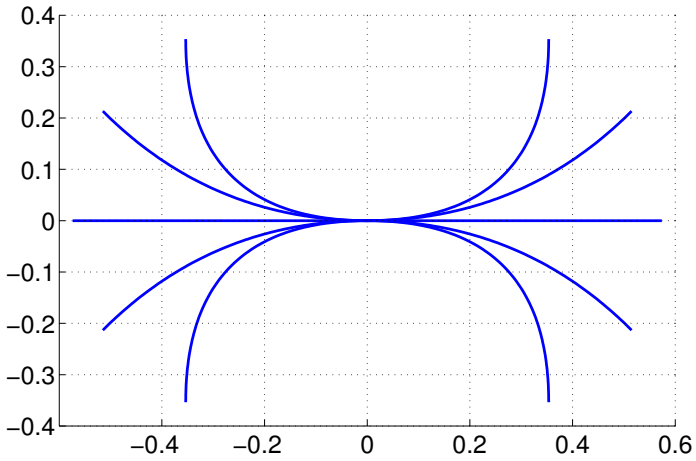


Figure 5.3 – Motion primitives for a mobile robot. Axes in kilometers.

For illustrative purposes we simulate $2D$ functions in space that can change with time, such that,

$$\begin{aligned} f: \quad \mathbb{R}^3 &\rightarrow \mathbb{R} \\ (x_1, x_2, t) &\rightarrow y\,. \end{aligned} \tag{5.12}$$

In these experiments, the pose $\mathbf{p} = (x_{1r}, x_{2r}, \theta_r)$ of a robot is the side-state for the SBO formulation and f is the unknown function to be estimated. The belief $b(f)$ is represented by a GP using a separable space-time covariance function [67]. The structure of the GP's covariance function can capture periodicity in f from the training data.

Since the robot travels at a certain speed $\dot{\mathbf{p}}$, the reachable area for sampling f depends on the side-state $\mathbf{p}$.

The action space A is determined by a set of motion primitives parametrised as 2D cubic splines (Section 4.3.1). With appropriate parametrisation, the curves generate ten primitives $A = \{\mathcal{C}_i\}_{i=1\ldots10}$ shown in Figure 5.3 for $\mathbf{p} = \mathbf{p}_0 = (0, 0, 0)$. For values of $\mathbf{p} = (x_{1r}, x_{2r}, \theta_r)$, the curves are rotated and translated using translation and rotation matrices. We define a transition function $T_{\mathbf{p}}(\mathbf{p}, \mathcal{C}_i, \mathbf{p}') = 1$ for a cubic spline

transformed from $\mathbf{p}$, with

$$\mathbf{p}' = \left(\mathcal{C}_i(u=1)_{x_1}, \mathcal{C}_i(u=1)_{x_2}, \left. \frac{\partial \mathcal{C}_{x_1}/\partial u}{\partial \mathcal{C}_{x_2}/\partial u} \right|_{u=1} \right) . \tag{5.13}$$

Before an action (curve) is selected for execution, the robot computes the optimal action using the MCTS algorithm (Algorithm 5). The robot gathers noisy samples from f along $\mathcal{C}$ while the action is being executed.

5.5.1 Static Function

In the first example, we simulate a static function, with expression

$$\begin{aligned} y = f(x_1, x_2, t) = \; & e^{-(x_1-4)^2} e^{-(x_2-1)^2} \\ & + 0.8 e^{-(x_1-1)^2} e^{-\left(\frac{x_2-4}{2.5}\right)^2} \\ & + 4 e^{-\left(\frac{x_1-10}{5}\right)^2} e^{-\left(\frac{x_2-10}{5}\right)^2} , \end{aligned} \tag{5.14}$$

where $x_1 \in [0, 5]$, $x_2 \in [0, 5]$, and $t \in [0, \infty]$. Figure 5.4 shows a plot for this function, where it is easy to distinguish two main peaks with different amplitudes. The robot is initially located at pose $\mathbf{p} = (0.5, 0.5, 0)$ and travels at a fixed speed of $0.2m/s$, gathering a sample every 5 minutes.

We first want to evaluate how the definition of the reward function R within the POMDP context impacts the action selection properties of the algorithm. We compare four different reward functions based on the UCB acquisition function, $r(\mathbf{x}) = \mu(\mathbf{x}) + \kappa_i \sigma(\mathbf{x})$, where $\kappa_i \in \{1.0 \times 10^6, 200, 20, 10\}$. It is a well known fact that the value of κ affects the exploration-exploitation trade off and this is clearly reflected in the resulting paths followed by each robot, as shown in Figure 5.5. The most explorative path sequence corresponds to $\kappa = 1.0 \times 10^6$ (Figure 5.5a) and the least explorative is $\kappa = 10$ (Figure 5.5d). Between these two extremes there are intermediate solutions where exploitation is favoured more strongly for lower values of κ.

In the next experiment we focus on the depth of the action selection search, i.e. the number n of lookahead steps for decision making. This corresponds to the maximum

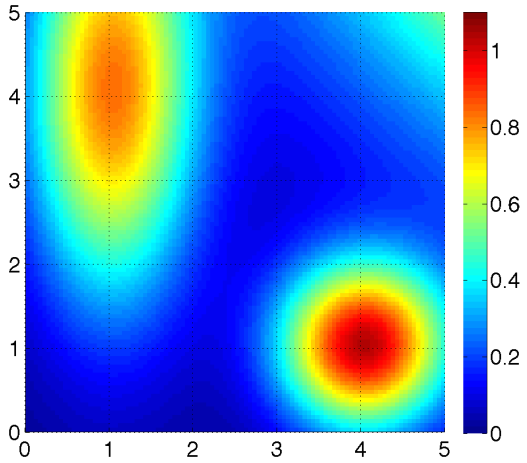


Figure 5.4 – Static goal function. Axes in kilometers.

Table 5.1 – Experiment for Depth and Algorithm Type Comparison

Id	Algorithm	Max Depth	Iterations
1	FT	1	10
2	FT	2	110
3	FT	3	1110
4	MCTS	3	100
5	MCTS	4	150
6	MCTS	5	400

depth allowed in the search tree. We first evaluate the entire decision tree, which means simulating all the possible combinations of actions. This approach, which we call *Full Tree* (FT) strategy, will need $|A|^n$ simulations which becomes impractical quickly. In fact, in this thesis we only consider FT strategies with $n \leq 3$. We compare the performance of FT against MCTS (Algorithm 5) where the number of simulations is a parameter. Clearly, for the same depth, MCTS is bounded by FT, however MCTS can find near-optimal solutions much faster. For this reason we were able to experiment with depths up to $n \leq 5$. We compare six different combinations of depth and algorithm type as indicated in Table 5.1.

The reward function used for these simulations was $r(\mathbf{x}) = \mu(\mathbf{x}) + 1.0 \times 10^6 \sigma(\mathbf{x})$ for all cases. Therefore, the only difference in action selection is due to the number of

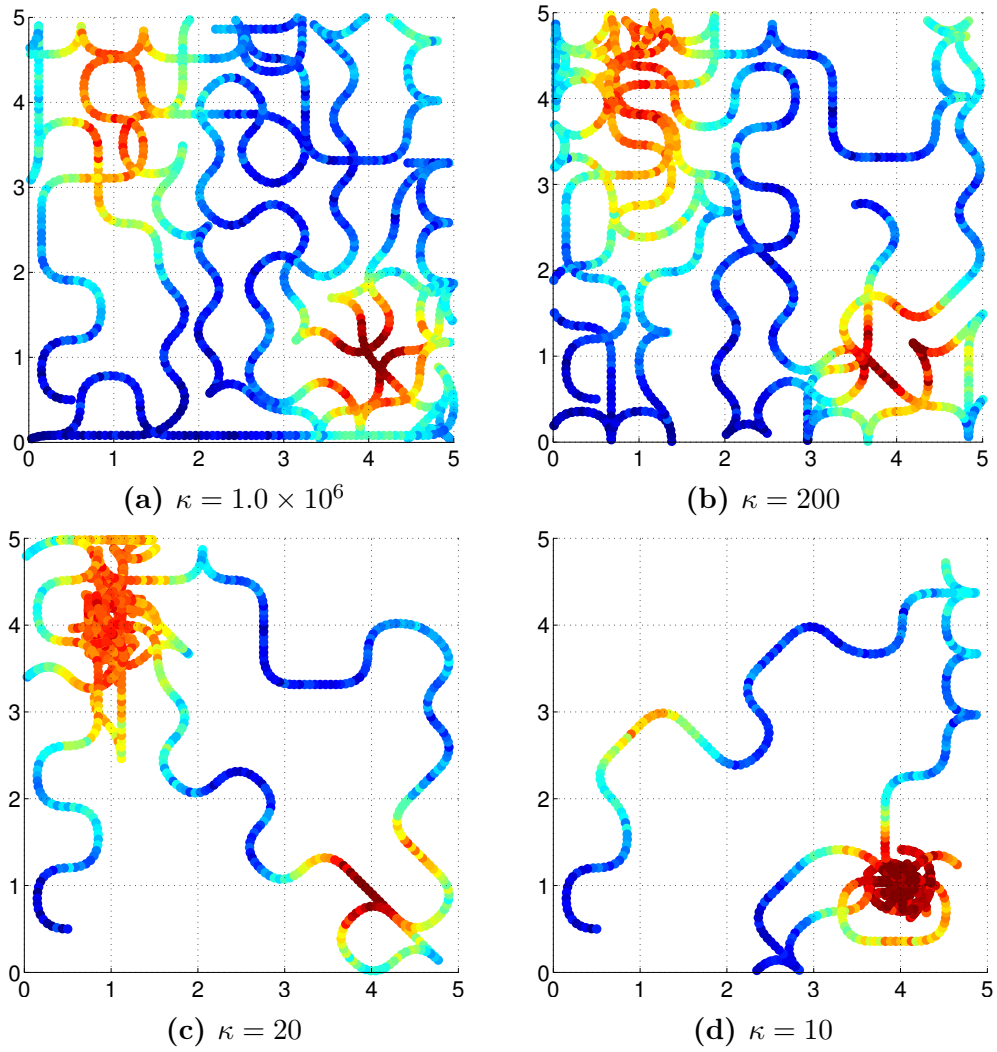


Figure 5.5 – Comparison of followed paths using Full Tree and UCB reward function with different values of κ. Axes in kilometers. Colour scale represents the value of sampled values.

lookahead steps. Figure 5.6 shows the paths followed by the robot at $t = 2.3$ days, when it had already gathered 616 samples from f. This figure does not show all cases, only the four most relevant ones. It is interesting to observe that the search with FT Depth 1 (Figure 5.6a) has not achieved a full coverage of the area and is highly susceptible to get trapped and collide into the edges of the domain, which is clearly sub-optimal from an exploration point of view. On the other hand, the FT Depth 2 shows increased coverage capability, which is improved further for deeper search strategies. FT Depth 3 and MCTS Depth 3 show similar result, with the clear

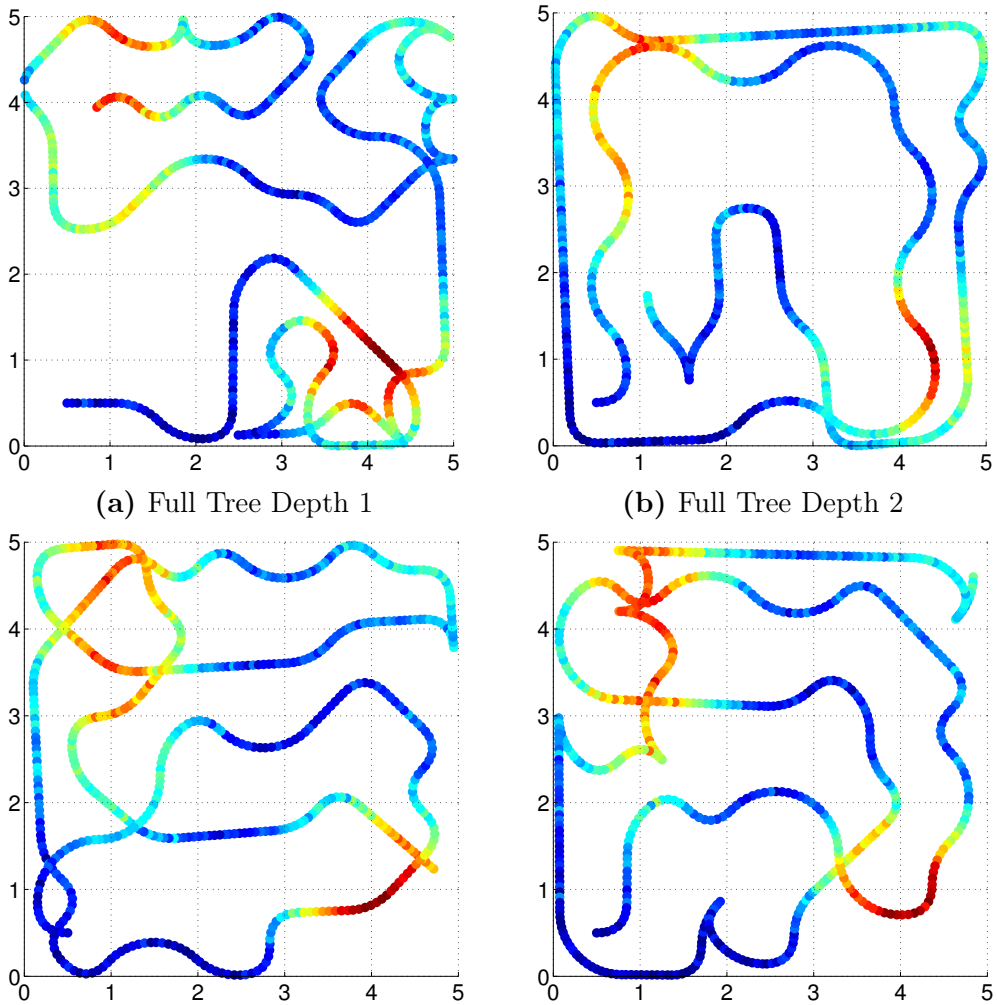


Figure 5.6 – Comparison of paths for purely exploration behaviour using FT and MCTS-UCT. Axes in kilometers. Colour scale represents the value of samples.

advantage that MCTS requires only 10% of the number of simulations of FT.

We also compare the accumulated reward over time for each case in Figure 5.7. This illustrates the advantage of using a multi-step lookahead strategy in increasing the total accumulated reward. However, it is not clear the advantage of using higher depths than two, as they do not show a clear improvement in accumulated reward. The main reason behind this is that f does not change over time, thus making the problem simple enough such that any strategy with depth greater than 1 would be

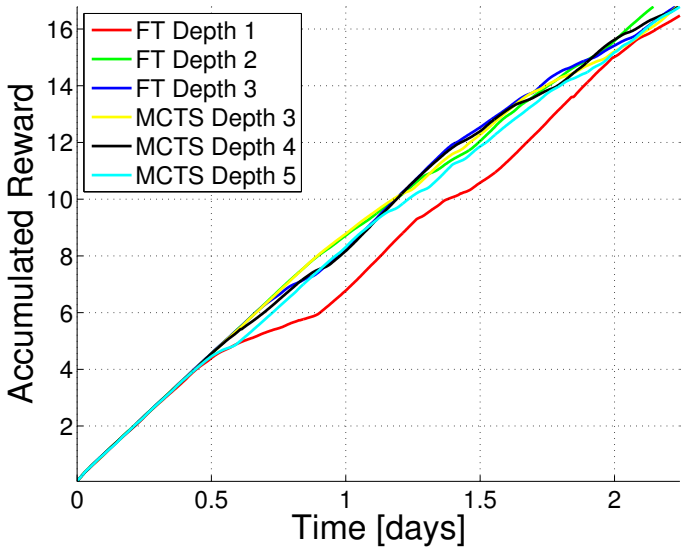


Figure 5.7 – Accumulated reward for static goal function.

very close to the optimal solution.

5.5.2 Dynamic Function

In this second experiment we use a more complex function that changes over time,

$$y = f(x_1, x_2, t) = \quad e^{-\left(\frac{x_1 - 2 - f_1(t)}{0.7}\right)^2} e^{-\left(\frac{x_2 - 2 - f_2(t)}{0.7}\right)^2}, \tag{5.15}$$

with $f_1(t) = 1.5\sin(2\pi t)$, $f_2(t) = 1.5\cos(2\pi t)$, $x_1 \in [0, 5]$, $x_2 \in [0, 5]$, and $t \in [0, \infty]$. This expression generates a function where the peak moves over time. The peak circles clockwise around $(x_1, x_2) = (2, 2)$ periodically, with a period of 1 day. The motivation for this example comes from air pollution monitoring tasks where we are interested in following peaks of pollution through time while learning how the entire process evolves in space and time. Figure 5.8 shows the goal function for 6 time steps within one period.

Similarly to the previous experiment, the robot is initially located at pose $\mathbf{p} = (0.5, 0.5, 0)$, travels at a fixed speed of $0.12m/s$ and gathers a sample every 15 minutes. The goal in this experiment is to find and follow the maximum of f over time. There-

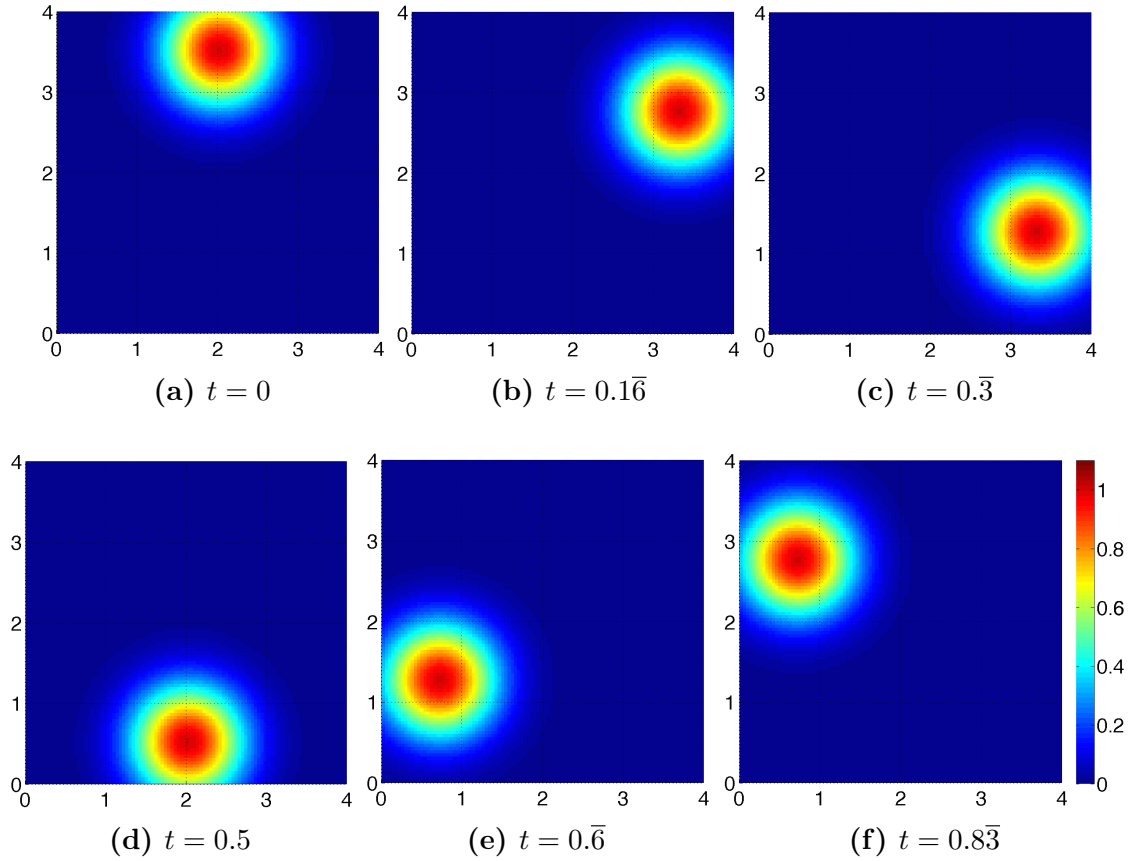
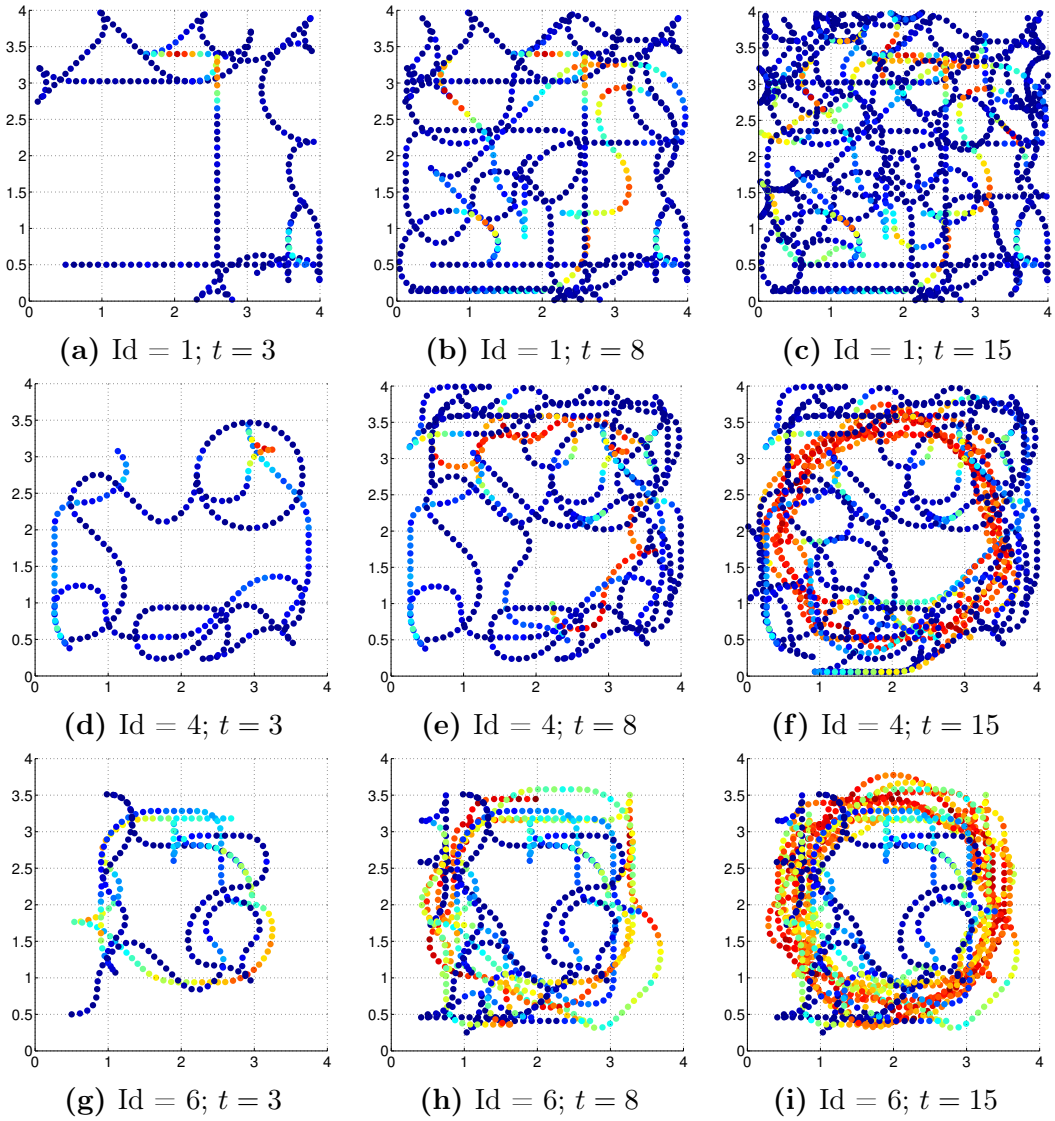


Figure 5.8 – Dynamic goal function within one period. Axes in kilometers.

fore, we select the reward function $r(\mathbf{x}) = \mu(\mathbf{x}) + 10\sigma(\mathbf{x})$, which according to Figure 5.5, should generate paths concentrated over the maximum values of f. Ideally, the robot should learn to follow the peak through time which would be possible for speeds greater than 0.109m/s. We try the same set of depth-algorithm pairs as in Section 5.5.1 and detailed in Table 5.1. We only show results for the extreme cases with the purpose of avoiding clutter in the figures.

Figure 5.9 illustrates the advantages of using multi-step lookahead strategies. The first row shows paths for FT Depth 1, where it can be seen that the robot does not learn how to follow the peak around a circle within 15 days. The second row, MCTS Depth 2, which only does 15 more simulations per iteration than FT Depth 1, the robot is already able to learn the circular pattern at $t = 12$ days. With deeper search strategies, the time required to learn the pattern decreases significantly indicating a



(a) Id = 1; $t = 3$ **(b)** Id = 1; $t = 8$ **(c)** Id = 1; $t = 15$

(d) Id = 4; $t = 3$ **(e)** Id = 4; $t = 8$ **(f)** Id = 4; $t = 15$

(g) Id = 6; $t = 3$ **(h)** Id = 6; $t = 8$ **(i)** Id = 6; $t = 15$

Figure 5.9 – Comparison of followed paths for FT and MCTS-UCT in a dynamic function. First row shows the paths for FT, Depth 1; Second row shows the paths for MCTS, Depth 2; Third row shows the paths for MCTS, Depth 5. Colour scale represents the value of samples.

better exploration and exploitation solution. In fact, for MCTS Depth 5 the pattern is learnt in $t = 8$ days.

Figure 5.10 shows the benefits of using non-myopic strategies for action selection. The cumulative reward is clearly larger for multi-step lookahead decision making algorithms. The best solution is MCTS Depth 5, that is clearly superior for the

entire duration of the simulation. A steeper slope for accumulated reward indicates that a method has learnt how to follow the peak. Then from this plot it is also clear that FT Depth 1 is not able to capture space-time dependencies properly.

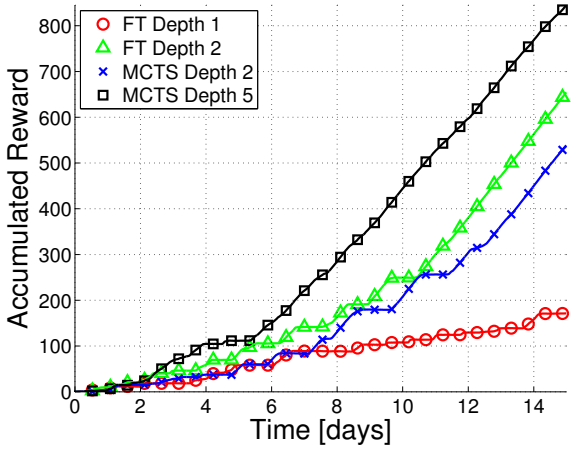


Figure 5.10 – Accumulated reward for dynamic goal function.

It is important also to compare FT Depth 2 with MCTS Depth 2. The fact that FT is an upper bound for MCTS Depth 2 can be confirmed from Figure 5.10. In addition, it can be seen that both strategies accumulate similar rewards, which is a good indication that MCTS will approximate the FT solution, even with only 25% of the total tree.

Finally, Figure 5.11 shows how MCTS prioritises the search over promising paths. The pose of the robot at this instant is $\mathbf{p} = (1.5, 3, 0)$. Red paths are the result of the function DefaultPolicy that did not get further explored and blue paths are the paths present in the tree. It can be seen how the tree automatically grows towards potentially informative areas, i.e. where the reward is higher. The green curve is the best branch of the tree.

5.6 Summary

In this chapter we proposed formulating the sequential BO problem as a POMDP. Our main contribution was to determine a non-myopic decision making solution that

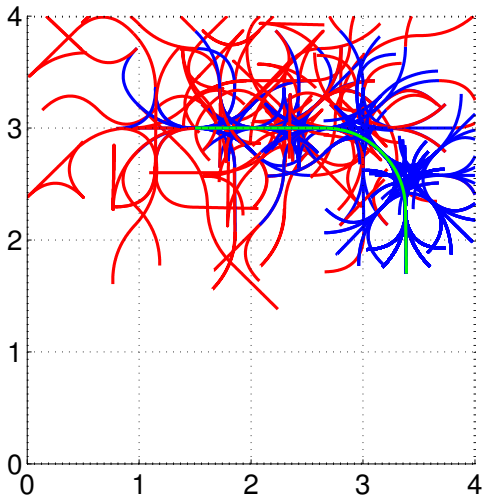


Figure 5.11 – Example of tree built for MCTS Depth 5.

maximises reward and takes into account the belief of an unknown space time process and the state of a mobile robot acting as a sensor. We formulated the solution for the POMDP analogue of SBO using a modified version of the UCT algorithm for MCTS, which is a scalable and efficient way of finding asymptotically optimal decisions. We demonstrate empirically the advantage of using non-myopic planning solutions, which becomes especially important when the objective function dynamically changes over time.

Even though long-term decision-making under uncertainty is a very complex problem, we solved it using a scalable method that works for realistic scenarios with state-dependent restrictions and time variation. We believe that using multi-step lookahead path planning is a convenient and practical way for solving many robotic problems requiring the accurate representation of real space-time phenomena, such as environment monitoring.

Chapter 6

Conclusions And Future Work

This thesis has addressed the problem of planning in dynamic environments. The planning problem is related to optimal sensing of a environmental phenomenon that changes with space and time. The planning algorithms proposed in this thesis solved the optimal sensing problem simultaneously satisfying two important objectives. The first challenge is learning the spatial and temporal patterns of the phenomenon, and the second challenge is to find the areas of interest (e.g. high pollution). Consequently, the proposed algorithms can naturally deal with the exploration-exploitation trade-off.

Three different kind of planning algorithms were proposed in the thesis, each with a higher level of complexity. The first one plans over waypoints, which is a simple but efficient solution. The second algorithm allows planning over continuous paths, solving not only the question of *where* and *when* to sample, but *how* to get there. Finally, the third planning algorithm is non myopic as it considers a sequence of decisions. It ensures that the next decision is the best, taking into account future actions and their corresponding outcomes.

The remaining of this chapter is organised as follows. Section 6.1 provides a summary of the contributions and draws conclusions of the developed theory and results obtained in previous chapters. Finally, Section 6.2 presents directions for future research, enumerating open challenges and interesting research opportunities.

6.1 Summary of Contributions

***Bayesian Optimisation* (BO) for trajectory planning in robotics**

BO is an optimisation technique which is useful for optimising unknown, noisy and costly to evaluate functions. While it can be readily applied to cases where sampling locations can be chosen freely over the input domain, this is not the case for the real robotic problems addressed in this thesis. All the proposed planning algorithms of this thesis extended the BO framework to work under real robotic deployments. The goal function to be optimised corresponds to the realisation of a real environmental phenomenon that changes with space and time. Chapters 3, 4 and 5 extended the plain BO algorithm to decision-making in a robotics context. These algorithms where implemented on real platforms and compared accordingly. The results show that the proposed algorithms are able to simultaneously learn and monitor interesting areas of an environmental phenomenon.

New family of acquisition functions for BO, which considers side state

The case of study addressed in this thesis, spatial-temporal monitoring, involves the use of an autonomous robot to perform sampling over the initially unknown environmental phenomenon. The state of the moving robot was used as extra information for the proposed algorithms, i.e., are defined as side state. Chapter 3 presented a new family of acquisition functions, which are state aware. The discrete planning algorithm used the side state information to select the next sampling location relatively close to the current position of the robot. Results show that incorporating side state can help reduce excessive displacement over the input space, while maintaining the accuracy of the spatial-temporal model.

Generalisation of BO for planning over continuous paths

The plain BO algorithm conducts an internal optimisation procedure, the maximisation of the acquisition function, which determines the next best location to sample. The algorithms in Chapters 4 and 5 showed extensions of the original BO algorithm

to optimise over continuous paths. The inner optimisation of the acquisition function was no longer conducted over discrete locations. It was adapted to optimise the acquisition function over the space of paths. Thus, the path planning techniques, which are not restricted to any path parametrisation in particular, evaluated the cumulative reward along a continuous path rather than a discrete location. Experimental evaluation shows the advantages of planning over continuous paths rather than discrete locations.

Layered BO for planning in spatial-temporal monitoring

The inner optimisation routine of the plain BO algorithm (Line 2 in Algorithm 1) was modified to optimise over the parameter space of a path. The reward function is no longer a cheap to evaluate function with derivative information; it now corresponds to a costly to evaluate function that involves integrating along a path. Chapter 4 showed how to stack two layers of BO, with the first one optimising over the goal function f and the second to optimise the acquisition function. This resulted in a more efficient search for the optimal path parameter set, which took much longer and resulted in local optimum values for standard optimisation routines.

***Sequential Bayesian Optimisation* (SBO)**

A non-myopic extension of the BO algorithm was formulated in Chapter 5. Defined as SBO, this new optimisation routine takes into account a number of future decisions to accumulate reward. The expected reward, which is shortsighted for existing BO literature, was generalised to a sequential formulation that evaluates the collective reward for a set of discrete locations or paths. The next best decision is now chosen based on all possible future decisions including their associated outcomes weighted by their probability of occurrence.

SBO as a *Partially Observed Markov Decision Process* (POMDP)

Chapter 5 showed how SBO can be formulated under the POMDP framework. All the elements of a POMDP were extracted from the SBO problem, including the definition

of state, action space, transition function and reward function. The POMDP analogue of SBO was solved using an online solver based on *Monte Carlo Tree Search* (MCTS) and *Upper Confidence Bound for Trees* (UCT). The experiments show the advantages of using deeper search strategies for decision-making. Increasing the number of lookahead decisions improves the overall performance of the planning algorithm.

6.2 Future Research

Although the proposed planning algorithms presented a valid solution for monitoring dynamic phenomena, there are several ways in which they can be extended and improved. This section presents some directions for future work.

Localisation Uncertainty in Spatial Temporal Model

The spatial-temporal models of environmental phenomena used in this thesis assume that samples from the phenomenon are referenced perfectly over the input domain. While there is a reasonable accuracy for most GPS and laser-based localisation systems, this is not always the case for more complex scenarios. For example, underwater in the absence of visual features, the uncertainty in localisation cannot be neglected. Using previous work by McHutchon and Rasmussen [49], it is posible to account for localisation errors and propagate uncertainties into the predictive probability density function of the phenomenon.

Integration of Observations Along Continuous Paths

Even though the developed algorithms can find optimal paths defined over a continuous domain, they only integrate observations at discrete locations. A clear improvement would be to use previous research by O'Callaghan and Ramos [51] and use integral kernels to include observations along continuous paths. This can be particularly useful when sensors have high latency, i.e. when observations actually occur over a path while the robot is moving and not at a discrete location.

Finding the Full Policy for the POMDP Analogue of SBO

The current solution for the POMDP analogue of SBO can only determine the next best decision based on the reachable belief state. Theoretically speaking there is a full solution to the POMDP, which corresponds to finding the optimal policy. This optimal policy could be calculated offline and provide the optimal action under any belief state.

Bibliography

[1] Jonathan Binney and Gaurav S. Sukhatme. Branch and Bound for Informative Path Planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[2] Sofiane Brahim-Belhouari, SofianeBelhouari and Amine Bermak. Gaussian Process for Non-Stationary Time Series Prediction. *Computational Statistics and Data Analysis*, 2004.

[3] Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. Technical Report arXiv:1012.2599, University of British Columbia, 2010.

[4] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 2012.

[5] Lawrence A. Bush, Brian Williams, and Nicholas Roy. Computing Exploration Policies via Closed-form Least-Squares Value Iteration. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2008.

[6] C. Chekuri and M. Pal. A Recursive Greedy Algorithm for Walks in Directed Graphs. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2005.

[7] Jen Jen Chung, Nicholas R.J. Lawrence, and Salah Sukkarieh. Gaussian Process for Informative Exploration in Reinforcement Learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[8] Emile Contal, Vianney Perchet, and Nicolas Vayatis. Gaussian Process Optimization with Mutual Information. In *International Conference on Machine Learning (ICML)*, 2014.

[9] Dennis D. Cox and Susan John. A Statistical Method for Global Optimization. In *IEEE Conference on Systems, Man, and Cybernetics (SMC)*, 1992.

[10] John P. Cunningham, Zoubin Ghahramani, and Carl Edward Rasmussen. Gaussian Processes for Time-Marked Time-Series Data. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

[11] Roger Daley. *Atmospheric Data Analysis*. Press Syndicate of the University of Cambridge, 1992.

[12] Marc Peter Deisenroth, Carl Edward Rasmussen, and Jan Peters. Gaussian Process Dynamic Programming. *Neurocomputing*, 2009.

[13] Eric Delmelle. Spatial sampling. In *Spatial Analysis*. SAGE Publications Ltd, London, 2009.

[14] Matthew Dunbabin and Lino Marques. Robotics for environmental monitoring. *IEEE Robotics Automation Magazine*, 2012.

[15] B. Ferris, D. Hahnel, and Dieter Fox. Gaussian processes for signal strength-based location estimation. In *Robotics Science and Systems (RSS)*, 2006.

[16] Dieter Fox. Kdl-sampling: Adaptive particle filters. In *Neural Information Processing Systems (NIPS)*, 1998.

[17] Carlos Gaetan and Xavier Guyon. *Spatial Statistics and Modeling*. Springer New York, 2010.

[18] Walter Gauschi. *Numerical Analysis*. Springer New York, 2012.

[19] David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. A multi-points criterion for deterministic parallel global optimization based on gaussian processes. HAL: hal- 00260579, 2008.

[20] Daniel Golovin and Andreas Krause. Adaptive Submodularity: Theory and Applications in Active Learning and Stochastic Optimization. *Journal of Artificial Intelligence Research*, 2011.

[21] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, 2007.

[22] Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. Near-optimal sensor placements in Gaussian processes. In *International Conference on Machine Learning (ICML)*, 2005.

[23] James Hensman, Nicolo Fusi, and Neil Lawrence. Gaussian Process for Big Data. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2013.

[24] Dave Higdon. Space and space-time modeling using process convolutions. *Quantitative methods for current environmental issues*, 2002.

[25] Matthew Hoffman, Eric Brochu, and Nando de Freitas. Portfolio allocation for bayesian optimization. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2011.

[26] Geoffrey A. Hollinger and Gaurav S. Sukhatme. Sampling-based Motion Planning for Robotic Information Gathering. In *Robotics Science and Systems (RSS)*, 2013.

[27] Geoffrey A. Hollinger, Brendan Englot, Franz Hover, Urbashi Mitra, and Gaurav S. Sukhatme. Uncertainty-driven view planning for underwater inspection. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[28] Shervin Javdani, Yuxin Chen, Amin Kerbasi, Andreas Krause, J. Andrew Bagnell, and Siddharta Srinivasa. Near Optimal Bayesian Active Learning for Decision Making. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2014.

[29] Donald R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 2001.

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

[31] Andreas Krause and Carlos Guestrin. Near-optimal Observation Selection using Submodular Functions. In *AAAI Conference on Artificial Inteligence*, 2007.

[32] H.J. Kushner. A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise. *Journal of Basic Engineering*, 1964.

[33] Phaedon Kyriakidis and André Journel. Geostatistical Space–Time Models: A Review. *Mathematical Geology*, 1999.

[34] Xiaodong Lan and Mac Schwager. Planning Periodic Persistent Monitoring Trajectories for Sensing Robots in Gaussian Random Fields. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[35] Steven M. LaValle. Rapidly-Exploring Random Trees: A New Tool for Path Planning. Technical report, 1998.

[36] Neil Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *Neural Information Processing Systems (NIPS)*, 2003.

[37] Nhu D. Le and James V. Zidek. *Statistical Analysis of Environmental Space-Time Processes*. Springer New York, 2006.

[38] Jerome Le Ny and George J. Pappas. On Trajectory Optimization for Active Sensing in Gaussian Process Models. In *International Conference on Decision and Control (CDC)*, 2009.

[39] Zhan Wei Lim, David Hsu, and Wee Sun Lee. Adaptive Informative Path Planning in Metric Spaces. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2014.

[40] Daniel James Lizotte. *Practical Bayesian Optimization*. PhD thesis, University of Alberta, 2008.

[41] Jaakko Luttinen and Alexander Ilin. Efficient Gaussian Process Inference for Short-Scale Spatio-Temporal Modeling. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

[42] David Mackay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[43] Roman Marchant and Fabio Ramos. Bayesian Optimisation for Intelligent Environmental Monitoring. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[44] Roman Marchant and Fabio Ramos. Bayesian Optimisation for Informative Continuous Path Planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[45] Roman Marchant, Fabio Ramos, and Scott Sanner. Sequential Bayesian Optimisation for Spatial-Temporal Monitoring. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2014.

[46] Ruben Martinez-Cantin, Nando de Freitas, Arnaud Doucet, and José Castellanos. Active Policy Learning for Robot Planning and Exploration under Uncertainty. In *Robotics Science and Systems (RSS)*, 2007.

[47] Ruben Martinez-Cantin, Nando de Freitas, Eric Brochu, José Castellanos, and Arnaud Doucet. A bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Autonomous Robots*, 2009.

[48] Georges Matheron. The intrinsic random functions and thier applications. *Advances in Applied Probability*, 1973.

[49] Andrew McHutchon and Carl Edward Rasmussen. Gaussian process training with input noise. In *Neural Information Processing Systems (NIPS)*, 2011.

[50] Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The Application of Bayesian Methods for Seeking the Extremum. *Towards Global Optimization*, 1978.

[51] Simon O'Callaghan and Fabio Ramos. Continuous occupancy mapping with integral kernels. In *AAAI Conference on Artificial Inteligence*, 2011.

[52] Simon O'Callaghan and Fabio Ramos. Gaussian Process Occupancy Maps. *International Journal of Robotics Research*, 2012.

[53] Michael A. Osborne. *Bayesian Gaussian Process for Sequential Prediction, Optimisation and Quadrature*. PhD thesis, University of Oxford, 2010.

[54] Michael A. Osborne, Roman Garnett, and S.J. Roberts. Gaussian processes for global optimization. In *International Conference on Learning and Intelligent Optimization (LION)*, 2009.

[55] Robert Platt, Russ Tedrake, Leslie Kaelbling, and Tomas Lozano-Perez. Belief Space Planning Assuming Maximum Likelihood Observations. In *Robotics Science and Systems (RSS)*, 2010.

[56] Josep M. Porta, Nikos Vlassis, T.J. Span Matthijs, and Pascal Poupart. Point-Based Value Iteration for Continous POMDPs. *Journal of Machine Learning Research*, 2006.

[57] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A Unifying View of Sparse Approximate Gaussian Process Regression. *Journal of Machine Learning Research*, 2005.

[58] Carl Edward Rasmussen and C.K.I. Williams. *Gaussian processes for machine learning*. The MIT Press, Cambridge, Massachusetts, 2006.

[59] Stephane Ross, Joelle Pineau, Sebastien Paquet, and Brahim Chaib-draa. Online Planning Algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 2008.

[60] Simo Sarkka and Jouni Hartikainen. Infinite-Dimensional Kalman Filtering Approach to Spatio-Temporal Gaussian Process Regression. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

[61] Matthias Schonlau. *Computer Experiments and Global Optimization*. PhD thesis, University of Waterloo, 1997.

[62] Bernhard Schöolkopf and Alexander Smola. *Learning with Kernels*. The MIT Press, Cambridge, Massachuset, 2002.

[63] Yirong Shen, Andrew Ng, and Matthias Seeger. Fast Gaussian Process Regression using KD-Trees. In *Neural Information Processing Systems (NIPS)*, 2006.

[64] David Silver and Joel Veness. Monte-Carlo Planning in Large POMDPs. In *Neural Information Processing Systems (NIPS)*, 2010.

[65] Amarjeet Singh and Andreas Krause. Nonmyopic adaptive informative path planning for multiple robots. In *International Joint Conference on Artificial Inteligence (IJCAI)*, 2009.

[66] Amarjeet Singh, Andreas Krause, Carlos Guestrin, William Kaiser, and Maxim Batalin. Efficient Planning of Informative Paths for Multiple Robots. In *International Joint Conference on Artificial Inteligence (IJCAI)*, 2007.

[67] Amarjeet Singh, Fabio Ramos, Hugh Durrant Whyte, and William J Kaiser. Modeling and decision making in spatio-temporal processes for environmental surveillance. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[68] Jefferson Souza, Roman Marchant, Lionel Ott, Denis F. Wolf, and Fabio Ramos. Bayesian Optimisation for Active Perception and Smoot Navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[69] Cyrill Stachniss, C Plagemann, and Achim J. Lilienthal. Gas distribution modeling using sparse Gaussian process mixture models. In *Robotics Science and Systems (RSS)*, 2008.

[70] Ruben Stranders, Alex Rogers, and N. Jennings. A decentralized, on-line coordination mechanism for monitoring spatial phenomena with mobile sensors. In *International Workshop on Agent Technology for Sensor Networks (ATSN)*, 2008.

[71] Junghun Suh and Songhwai Oh. A Cost-Aware Path Planning Algorithm for Mobile Robots. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[72] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, Massachusetts, 1998.

[73] Marc Toussaint. The Bayesian Search Game. *Theory and Principled Methods for the Design of Metaheuristics*, 2012.

[74] Volker Tresp. Mixtures of Gaussian Processes. In *Neural Information Processing Systems (NIPS)*, 2000.

[75] Shrihari Vasudevan, Fabio Ramos, Eric Nettleton, and Hugh Durrant-Whyte. Gaussian Process Modelling of Large-Scale Terrain. *Journal of Field Robotics*, 2009.

[76] Andrew Gordon Wilson and Ryan P. Adams. Gaussian Process Kernels for Pattern Discovery and Extrapolation. In *International Conference on Machine Learning (ICML)*, 2013.

[77] Jonas Witt and Matthew Dunbabin. Go with the Flow: Optimal AUV Path Planning in Coastal Environments. In *Australian Conference on Robotics and Automation (ACRA)*, 2008.