# Autonomous Exploration over Continuous Domains

**Gilad Francis**

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

**Faculty of Engineering and Information Technologies**
**University of Sydney**

2019

# 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.

**Gilad Francis**

17 April 2019

# Abstract

Motion planning is an essential aspect of robot autonomy, and as such it has been studied for decades, producing a wide range of planning methodologies. Path planners are generally categorised as either trajectory optimisers or sampling-based planners. The latter is the predominant planning paradigm as it can resolve a path efficiently while explicitly reasoning about path safety. Yet, with a limited sampling budget, the resulting paths are far from optimal, even in the local sense. In contrast, state-of-the-art trajectory optimisers explicitly trade-off between path safety and efficiency to produce locally optimal paths. However, these planners cannot incorporate updates from a partially observed model such as an occupancy map and fail in planning around information gaps caused by incomplete sensor coverage of the map.

Autonomous exploration adds another twist to path planning. The objective of exploration is to safely and efficiently traverse through an unknown environment in order to map it. The desired output of such a process is a sequence of paths that efficiently and safely minimise the uncertainty of the map. However, optimising over the entire space of trajectories is computationally intractable. Therefore, most exploration algorithms relax the general formulation by optimising a simpler one, for example finding the single *next best view*, resulting in suboptimal performance, as the gains and risks along the entire path are only considered over a limited set of points.

This thesis investigates methodologies for optimal and safe exploration over continuous paths. Contrary to existing exploration algorithms that break exploration into independent sub-problems of finding goal points and planning safe paths to these points, our holistic approach simultaneously optimises the coupled problems of where and how to explore. Thus, offering a shift in paradigm from *next best view* to *next best path*. With exploration defined as an optimisation problem over continuous paths, this thesis explores two different optimisation paradigms; Bayesian and functional.

The contributions of this thesis are as follows: First, we introduce an exploration method based on constrained Bayesian optimisation. This method finds optimal paths that inherently satisfy motion and safety constraints. As evaluating both the objective and constraint functions requires costly forward simulations, the Bayesian optimiser only proposes paths which are likely to yield optimal re-

sults and satisfy the constraints with high confidence, thus minimising the number of expensive function evaluations in search of the global optimum.

Second, we develop a stochastic functional gradient path planner for trajectory optimisation in occupancy maps. We show that drawing samples at random, instead of using a fixed resolution sampling, is crucial in order to circumvent the "information-gap" traps that render existing trajectory optimisation algorithms ineffective. Consequently, we extend the expressiveness of state-of-the-art functional planners by applying stochastic gradient update rule to optimise a path represented by a Gaussian process or kernel approximating features, resulting in an efficient trajectory optimisation methodology which is no longer limited by the path representation.

Third, we formulate exploration as a variational problem which allows us to directly optimise the search for exploration paths in the space of trajectories using functional gradient methods. We develop a *mutual information* (MI) variational objective for continuous occupancy maps, which replaces the common and expensive approach of computing MI explicitly over the entire map for each evaluated path with a simpler gradient update rule. We embed the MI objective in our stochastic functional gradient path planner to optimise safe paths with infinite expressivity.

# Acknowledgements

I would like to gratefully thank my supervisor Fabio Ramos for his guidance. With great patience and understanding, he provided all the support and encouragements for me to become an independent thinker.

I am also grateful to the University of Sydney and NICTA for the scholarship which allowed me to undertake this studies and to Lionel Ott, Rafael Dos Santos De Oliveira and Roman Marchant for their unfailing support from day one.

A very special gratitude goes out to my parents, Rahamim and Levana, for their faith, love and continuous support. For my children, Jonathan, Itamar, Avinoam and Elay, I thank for being there when I needed (and sometimes when I didn't).

Finally, and most importantly, I would like to thank my wife, Keren, for being my driving force. Without you, none of that would have happened.

# Nomenclature

## Notation

The following is a listing of the notation used throughout this thesis.

### General

| | |
|---|---|
| $\mathbb{R}$ | the real numbers |
| $\mathbf{x}$ | vector |
| $x_i$ | i-th element of vector $\mathbf{x}$ |
| $\mathbf{x}^T$ | transpose of vector $x$ |
| $I$ | The identity matrix |
| $K$ | matrix |
| $K_{(i,j)}$ | matrix element |
| $K^{-1}$ | Matrix inversion |
| $\|K\|$ | Matrix determinant |
| $N$ | number of points in a dataset |
| $\mathcal{N}(\mu, \Sigma)$ | normal distribution with mean $\mu$ and covariance $\Sigma$ |
| $\Phi_G(\mathbf{x})$ | standard normal cdf at $\mathbf{x}$, assuming $\mu = 0$, and $\Sigma = 1$) |
| $\phi_G(\mathbf{x})$ | standard normal pdf at $\mathbf{x}$, assuming $\mu = 0$, and $\Sigma = 1$) |

### Gaussian Process

| | |
|---|---|
| $\mathcal{GP}$ | Gaussian process |
| $X$ | Training set input locations |
| $Y$ | Training set target values |
| $\mathcal{D}$ | Training dataset $D = X, Y$ |
| $X^*$ | Query/test locations |
| $k(\mathbf{x}, \mathbf{x}')$ | Covariance function evaluated between $\mathbf{x}$ and $\mathbf{x}'$ |
| $m(\mathbf{x})$ | Mean function evaluated at $\mathbf{x}$ |
| $\theta, \upsilon$ | Process hyperparameters |
| $\sigma_n^2$ | Process noise variance |
| $\sigma_f^2$ | Covariance function signal variance |
| $\ell$ | Covariance function lengthscale |
| $K(X', X'')$ | Covariance matrix between of all pairs of inputs from $X'$ and $X''$ |

### Hilbert Maps

| | |
|---|---|
| $\Phi$ | Feature, defined by kernel $k$ |
| $\hat{\Phi}$ | Feature, approximating kernel matrix $K$ |
| $\mathbf{w}$ | LR weights vector |
| $\mathbf{z}_L$ | Linear projection |
| $\mathbf{z}_N L$ | Nonlinear projection |
| $\sigma$ | Sigmoid function |

### Functional optimisation

| | |
|---|---|
| $\xi$ | function |
| $\Xi$ | space of functions |
| $\mathcal{U}, \mathcal{F}$ | objective functional |
| $U_{obs}$ | Obstacles proximity (safety) functional |
| $U_{dyn}$ | Trajectory properties penalty functional |
| $\nabla_\xi$ | Functional gradient |
| $\nabla_x$ | Euclidean gradient |
| $\mathcal{C}$ | Configuration space, $\mathcal{C} \in \mathbb{R}^D$ |
| $\mathcal{W}$ | Robot workspace, $\mathcal{C} \in \mathbb{R}^2$ or $\mathbb{R}^3$ |
| $\Upsilon$ | Approximating (inner product) features |
| $\mathcal{T}$ | Finite set of samples |
| $\mathcal{Q}$ | Proposal sampling distribution |

### Functional Exploration

| | |
|---|---|
| $\hat{\mathbf{z}}_f$ | a set of unoccupied simulated observations |
| $H$ | Entropy of map model |
| $M\|\hat{\mathbf{z}}_f$ | modified Hilbert map conditioned on the expected observations |
| $H(M\|\hat{\mathbf{z}}_f)$ | Conditional entropy of map model |
| $U_{MI}$ | Mutual information functional |

**Abbreviations**

| | |
|---|---|
| BO | Bayesian optimisation |
| CBE | Constrained Bayesian exploration |
| CDF | Cumulative distribution function |
| CV | Cross-validation |
| EI | Expected improvement |
| GP | Gaussian process |
| GPC | Gaussian process classification |
| GPOM | Gaussian process occupancy maps |
| GPR | Gaussian process regression |
| IG | Information gain |
| KL | Kullback-Leibler (KL) divergence |
| LCB | Lower confidence bound |
| LR | Logistic regression |
| MAP | Maximum a posteriori |
| MI | Mutual information |
| ML | Maximum likelihood |
| NBV | Next best view |
| NLL | Negative log likelihood |
| NLML | Negative log marginal likelihood |
| OM | Occupancy map |
| OGM | Occupancy grid map |
| PDF | Probability density function |
| RKHS | Reproducing kernel Hilbert space |
| SDF | Signed Distance Field |
| SGD | Stochastic Gradient Descent |
| TSDF | Truncated Signed Distance Field |
| UCB | Upper confidence bound |
| UT | Unscented transform |

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Chapter 1

# Introduction

## 1.1 Motivation

The biologically inspired notion of autonomy is a common thread running through the various research disciplines in robotics. The prerequisite for autonomous robots is the ability to make decisions without explicit human intervention. Autonomous agents take various forms with varying levels of autonomy. Fig. 1.1 presents several examples; Roomba, the humble vacuum cleaning robot, Waymo, Google's self-driving car, and the elaborate Mars Rover.

An essential aspect of autonomy is motion planning. It is a prolific branch of robotics that has been studied for decades, producing a wide range of planning methods, typically adapted to the system's task and configuration. The goals of a planner are derived from the task the robot in undertaking. As an example, goals for a vacuum cleaning robot could be to the cover as much space as possible or to maximise dust collection. While the robot's task defines a context for planning, any path planning decision must also be feasible. A basic requirement from any path planner is safety, which encapsulates a variety of planning constraints, which can include obstacles, joint limits or time and energy budgets. Figure 1.2 schematically illustrates a path planning problem where a robot must avoid obstacles while traversing to its goal.

(a) iRobot's Roomba (b) Waymo (c) Mars Rover

Figure 1.1: Autonomous agents employed in various tasks: (a) iRobot's Roomba, the home cleaning robot (Courtesy of Wikimedia Commons), (b) Waymo, Google's self-driving car (Courtesy of Wikimedia Commons), (c) The Curiosity Mars Rover (courtesy NASA/JPL/Cornell University).

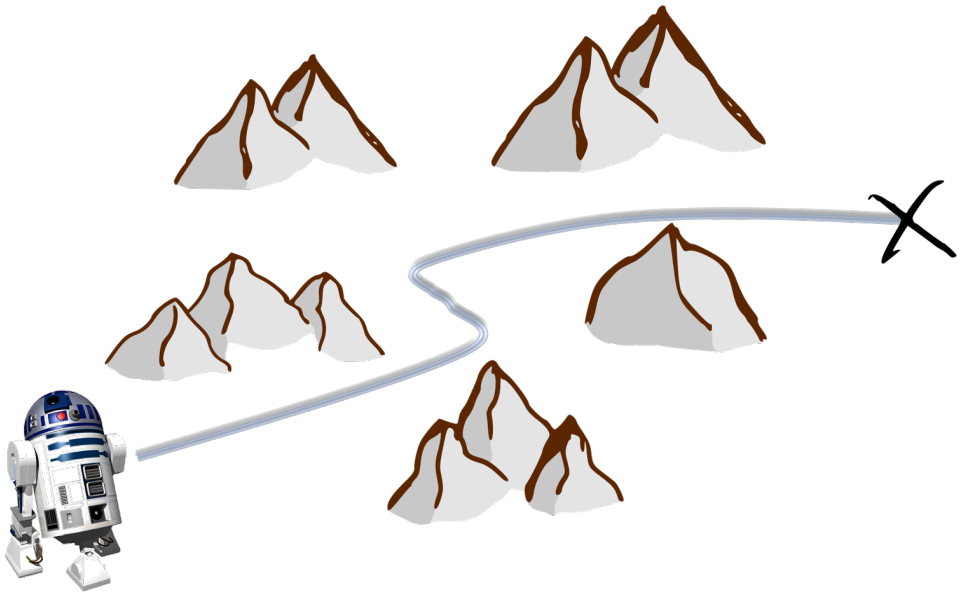


Figure 1.2: Path planning example. The robot must avoid obstacles on its way to its goal.

Autonomous path planning presents several challenges. Discrete motion planning problems can be solved using search algorithms such as Breadth first search or Dijkstra's algorithm (LaValle, 2006). However, in its broadest form, a path is represented by a continuous high-dimensional trajectory. This abstract representation transforms path planning into a complex optimisation problem, where the objective is non-convex and constraints are convoluted.

Another challenge in planning stems from the partial observability of the state of the robot and environment. A robot interacts with the environment using its sensors and actuators. For example, a laser range finder measures the angle and distance to the obstacles or an actuator moves a joint in a robotic arm to a desired angle. However, the robot's ability to understand and interact with its environment is corrupted by noise. Noise may result in incorrect estimation of the location of obstacles or may lead to an inappropriate positioning of a robotic arm. Hence, a robust planner must incorporate the noise in the robot's sensors and actuators in its decision-making process.

Autonomous exploration adds another layer of complexity to motion planning. The goal of exploration is to build a coherent representation of an unknown environment and thus is used in various remote sensing applications. But can a robot safely and efficiently explore an unknown environment? The inherent difficulty with exploration is the partially known model of the environment. Since this model does not have a closed-form, planning over it relies on samples. For

example, checking whether a trajectory is safe or not will require finitely-many samples along the trajectory. Greater confidence in the result of this process can be achieved by increasing the number of samples, albeit with dearer computational costs.

Evaluating the usefulness of a path also relies on samples. These evaluations produce a score based on the expected change in the environment model given a trajectory. However, can such a score be used in optimising path selection? Can both safety and optimality be incorporated in this process? Most exploration planners relax the general exploration problem by solving two simpler sub-problems. Instead of optimising the overall performance over the entire path, these methods optimise the selection of a finite set of goal points. With the choice of goal points, a safe path can be planned using sampling-based techniques. However, it is clear from the construction of such a solution, that this approach does not consider the usefulness of the entire trajectory.

## 1.2 Problem Statement

The general problem addressed in this thesis is: how to autonomously explore an unknown environment in a safe and efficient manner.

Exploration is a high level path planning problem that solves together two problems, where to go next and which way to take. Furthermore, any plan must also consider time or energy budgets and constraints, such as safety or the limits of actuators, which might be unknown.

Pragmatically, these two problems can be solved separately, deciding where to go followed by path planning to that point. However, solving both problems together, while more challenging, provides a solution which is more robust. A combined solution considers the overall gains, budget and constraints along the entire path. However, is such an abstraction of the exploration process computationally feasible? In this thesis, we will address the general exploration problem and suggest efficient optimisation methodologies to enable a tractable online autonomous exploration process.

## 1.3 Contributions

This thesis addresses the challenges of exploration, but also makes contributions to path planning in general. In the following we detail each of the contributions.

### 1.3.1 Formulation of Autonomous Exploration as an Optimisation over Continuous Paths

Exploration is an active sensing decision-making process that defines the paths a robot should traverse in order to build a high fidelity model of the environment. Optimal exploration decisions must consider not only where to go next, but also how to get there. These decisions encapsulate the gains and risks along the entire path. However, as this is a computationally intensive process, exploration is formulated as an optimisation over continuous paths. In this formulation, both the objective and constraints are considered expensive-to-evaluate black-box functions that are learned during optimisation.

### 1.3.2 Constrained Bayesian Exploration

*Constrained Bayesian Exploration* (CBE) is an innovative method to solve the exploration problem, i.e., optimise path decisions whilst keeping the robot safe and within its dynamic constraints. CBE provides a principled and robust approach for optimising exploration using Bayesian optimisation. As such, CBE guarantees convergence to a local solution and follows well known Bayesian optimisation's regret bounds (Srinivas et al., 2010).

### 1.3.3 Trajectory Optimiser for Continuous Occupancy Maps

Constrained Bayesian exploration offers a principled optimisation paradigm. However, it is effective only for low dimensional path representations, resulting in limited expressivity of the optimised trajectories. In comparison, path planners, which resolve a path between a start and goal points, can produce paths where the representation is only limited by the planning budget of the planner. Path planning using occupancy map is predominately performed by sampling-based planners such as *Rapidly exploring Random Trees* (RRT). While these planners are highly successful in finding safe paths, path optimisation is limited to the path's geometric properties, such as length or execution time. Therefore these planners can not optimise other objectives such as mutual information. Trajectory optimisers, on the other hand, directly optimise an objective function such as control cost or safety margins. However, these methods require a fully defined and precomputed cost map and cannot work with a partially observed model, such as occupancy maps, where parts of the model are unknown as they were never observed. The method presented in Chapter 4 provides a novel path optimisation approach for continuous occupancy maps. Effectively, this method extends previous work done on discrete precomputed cost maps to a continuous environment representation built from observations.

Unlike other functional gradient path planning techniques (e.g. (Marinho et al., 2016), (Zucker et al., 2013)), the stochastic functional gradient planner does not commit to a predetermined resolution, whether spatial or parametric. The optimisation update rule is replaced by a stochastic gradient approach. The stochastic samples allow flexible support for the path, which is maintained using a novel non-parametric path representation based on Gaussian processes.

### 1.3.4 Stochastic Trajectory Optimisation over an Expressive and Tractable Path Model based on Kernel Approximations

A non-parametric path representation offers great flexibility which works well with the stochastic updates required by path planning over occupancy map. The main disadvantage of such a representation is its high computational cost. Kernel approximation techniques, on the other hand, form an expressive and tractable non-linear model for path representation which uses fixed cost for updating and querying. Furthermore, a path model based on kernel approximation can be efficiently optimised using randomly drawn samples, which does not commit to a predetermined fixed sampling resolution.

### 1.3.5 Developing a Mutual Information Variational Objective for Continuous Occupancy Maps

Optimising exploration paths requires expensive forward-simulations to estimate MI over the entire map for each evaluated path. Hence, existing exploration optimisation algorithms optimise the selection of a discrete set of waypoints. We propose instead a *mutual information* (MI) variational objective which replaces the common and expensive approach of computing MI explicitly over the entire map for each evaluated path. This variational exploration method modifies the path using MI functional gradients without the need to compute MI explicitly. The MI gradients are obtained from local perturbations on the continuous map model which are derived in closed-form.

### 1.3.6 Next Best Path Exploration Method

An information-driven variational framework for safe autonomous exploration in continuous occupancy maps. Path optimisation is performed directly in the space of trajectories using a combined objective; which considers safety, efficiency and information. The method is invariant to the choice of path representation as it uses stochastic functional gradient descent to optimise the objective along the entire path.

## 1.4 Outline

This chapter reviewed challenging aspects in path planning that affect robotic autonomy, and presented the problem addressed in this thesis of safe and efficient exploration.

Chapter 2 provides the background necessary for understanding this thesis. Gaussian process regression, classification and model selection are reviewed in Section 2.1 before describing the unconstrained and constrained Bayesian optimisation framework in Section 2.2. Section 2.3 provides an overview of functional gradient descent optimisation methods for path planning. Discrete and continuous occupancy map representations are detailed in Section 2.4.

Chapter 3 presents the constrained Bayesian exploration method, where Bayesian optimisation is employed to optimise safe information collection over continuous paths. The problem is introduced in Section 3.1 and Section 3.2 reviews related work. Section 3.3 formally describes the problem of safe autonomous exploration for building occupancy maps, and Section 3.4 introduces the basic building blocks of constrained Bayesian exploration and details the algorithms behind it. Evaluation of the performance of constrained Bayesian exploration, in simulation and with a real robot, are described in Section 3.5, which demonstrate the advantages of this method over other exploration techniques.

Chapter 4 introduces the stochastic path planner which optimises trajectory selection using continuous occupancy maps. Section 4.1 provide motivation for a stochastic trajectory optimiser for occupancy maps. Section 4.2 reviews related work in path planning in occupancy maps. The functionals used in path planning and the required modifications for Hilbert maps are described in Section 4.3 and Section 4.4, respectively. Experiments in Section 4.5 demonstrate the necessity of stochastic samples to ensure convergence of path optimisation.

In Chapter 5 a scalable form of the stochastic path planner is introduced. Section 5.1 reviews the motivation for such a planner, which is an improvement of the stochastic path planner of Chapter 4. Section 5.2 derives the optimisation update rule and presents the stochastic functional algorithm under a kernel approximation paradigm. Section 5.2 also reviews a targeted sampling method that accelerate optimisation and provide a measure for functional optimisation convergence. Section 4.5 compares the performance of the approximate kernel planner, using different kernel approximation techniques, with other planning method for occupancy map. It also demonstrates the effectiveness of the targeted sampling schedule.

Chapter 6 derives a functional exploration algorithm for continuous occupancy maps that enables optimisation of an information functional over continuous

paths. Literature on autonomous exploration in continuous occupancy maps is surveyed in Section 6.2. Section 6.3 describes in detail the functional exploration algorithm, and the mutual information gradient derived from perturbation on the continuous map. In Section 6.4, the functional exploration algorithm is benchmarked against other exploration methods for continuous occupancy maps.

The thesis concludes in Chapter 7 with a summary of the work presented in Section 7.1 and directions for future research in Section 7.2.

## 1.5 List of Publications

The following is the list of publications contributing to this thesis and the corresponding chapters.

- G. Francis, L. Ott, R. Marchant and F. Ramos. Occupancy Map Building through Bayesian Exploration. In *accepted for publication in the International Journal of Robotics Research*, 2019. Publication summarises research presented in Chapter 3.

- G. Francis, L. Ott and F. Ramos. Stochastic Functional Gradient for Motion Planning in Continuous Occupancy Maps. In *Proceedings of IEEE International Conference on Robotics and Automation*, 2017. Appears in Chapter 4.

- G. Francis, L. Ott and F. Ramos. Stochastic Functional Gradient for Motion Planning in Continuous Occupancy Maps. In *Proceedings of the Third Machine Learning in Planning and Control of Robot Motion Workshop, IEEE International Conference on Robotics and Automation*, 2018. Appears in Chapter 5.

- G. Francis, L. Ott and F. Ramos. Fast Stochastic Functional Path Planning in Occupancy Maps. In *to appears in IEEE International Conference on Robotics and Automation*, 2019. Appears in Chapter 5.

- G. Francis, L. Ott and F. Ramos. Functional Path Optimisation for Exploration in Continuous Occupancy Maps. In *Proceedings of the International Symposium on Robotics Research* , 2017. Appears in Chapter 6

# Chapter 2

# Background

In this chapter we present the theoretical background and notation required to establish the planning algorithms presented in this thesis. As exploration, in its general form, is an optimisation problem over continuous domains, we introduce two global optimisation methods which we extend in subsequent chapters for path planning and safe autonomous exploration. In addition, we briefly introduce the occupancy mapping methods that are used in the various planning algorithms.

## 2.1 Gaussian Processes

A *Gaussian process* (GP) is a Bayesian non-parametric method used for regression and classification. GPs provide a structured mechanism that extends the Gaussian probability distribution, which is defined over a finite set of random variables, to a distribution over functions, i.e. an infinite set. GPs generate a non-parametric, non-linear tractable regression model that can produce probabilistic predictions anywhere in the model's domain. This trait is especially useful in autonomous planning problems, as will be shown throughout this dissertation.

The following section briefly reviews the theory of non-linear regression and classification using GPs. It then presents the covariance functions used in the various motion planning algorithms used in this work followed by a short description of how to learn the hyper-parameters of these functions. For extensive discussions on the various aspects of GPs, we refer the reader to the work of Rasmussen and Williams (2006).

### 2.1.1 Gaussian Process Regression

GPs generate a regression model of a function $f$ based on observations. The model can be queried at any location $\mathbf{x}$ to produce a random variable $f(\mathbf{x})$. The GP model is completely defined by a mean function $m(\mathbf{x})$ and a covariance function $k(\mathbf{x}, \mathbf{x}')$ and is notated as follows;

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

where $\theta$ is a set of hyper-parameters of the mean and covariance functions.

The GP regression model is updated by function observations. Each observation is defined by a location $\mathbf{x}_i \in \mathbb{R}^{\mathbb{D}}$ and the observed, potentially corrupted by noise, function value $y_i \in \mathbb{R}$. A GP maintains a dataset of $N$ observations $\mathcal{D} = \{X, Y\}$, where $X = \{\mathbf{x}_i\}_{i=1}^N$ and $Y = \{y_i\}_{i=1}^N$, which is used during inference. The observed target values $y$ are typically considered to be corrupted with noise, most commonly assumed Gaussian with zero mean and variance $\sigma_n^2$:

$$y = f(x) + \epsilon \qquad \epsilon \sim \mathcal{N}(0, \sigma_n^2) \tag{2.2}$$

The key concept behind GP inference is that any finite set of function values, i.e. the likelihood, has a joint Gaussian distribution. Therefore the predicted function values $f(X^*)$ at locations $X^*$ and previous function observations $f(\mathbf{x}^*)$ follow the prior GP distribution

$$\begin{bmatrix} Y \\ f(X^*) \end{bmatrix} \sim \mathcal{N}\left( \begin{bmatrix} m(X) \\ m(X^*) \end{bmatrix}, \begin{bmatrix} K(X,X) & K(X,X^*) \\ K(X^*,X) & K(X^*,X^*) \end{bmatrix} + \sigma_n^2 I \right), \tag{2.3}$$

where $m(X)$ and $m(X^*)$ are the mean function values at $X$ and $X^*$, respectively. $I$ is the identity matrix and $K$ is the covariance matrix, which is computed between two sets of inputs. Generally, $K(X, X^*)$ denotes a matrix holding the covariance values of all pairs of inputs from $X$ and $X^*$:

$$K(X, X^*)_{(i,j)} = k(x_i, x_j^*) \qquad \forall x_i \in X, x_j^* \in X^*. \tag{2.4}$$

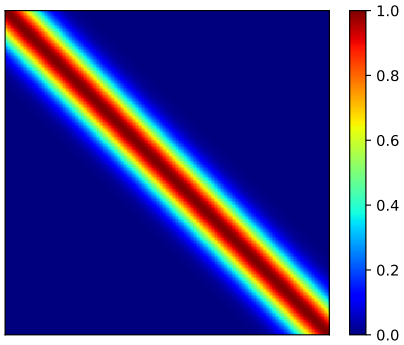
Equation (2.3) implies that any finite set of function values is jointly Gaussian. This trait enables tractable queries of the GP model. Given observations $\mathcal{D}$, the predicted function values at locations $X^*$ can be tractably computed from observations by the conditional distribution, $f^*|X^*, X, Y \sim \mathcal{N}(\mu^*, \Sigma^*)$, where the predictive mean is:

$$\mu^* = m(X^*) + K(X^*, X)[K(X, X) + \sigma_n^2 I]^{-1}(Y - m(X)), \tag{2.5}$$

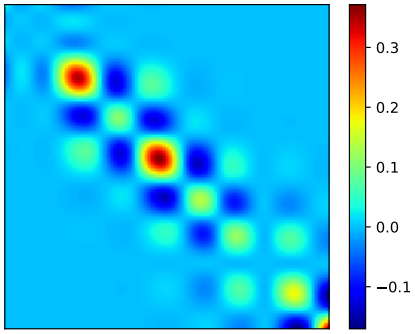
and the predictive variance is:

$$\Sigma^* = K(X^*, X^*) - K(X^*, X)[K(X, X) + \sigma_n^2 I]^{-1} K(X, X^*)). \tag{2.6}$$

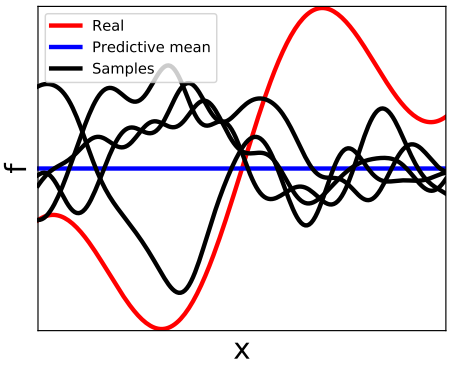
Figure 2.1 depicts the covariance matrices of the GP prior and posterior and function samples from these distributions. Before data was observed, the prior covariance, Fig. 2.1a, which is based on an RBF kernel dictates smooth function samples around a zero mean, Fig. 2.1c. When data is incorporated in the GP,



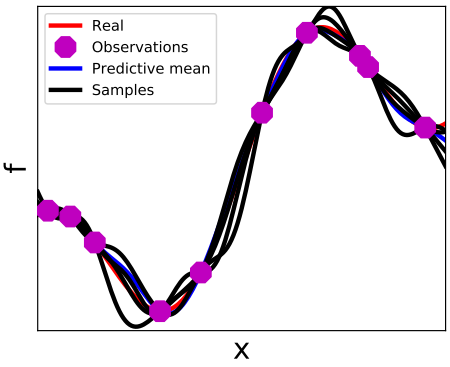

(a) Prior covariance matrix




(b) Posterior covariance matrix




(c) Function samples from prior




(d) Function samples from posterior

Figure 2.1: GP prior and posterior distributions: (a) prior covariance matrix, (b) posterior covariance matrix, (c) function samples from prior, and (d) samples from posterior.

the covariance, Fig. 2.1b, decreases around the observed locations. In effect, the observed data impose constraints on the GP resulting in function samples, Fig. 2.1d, that must comply with these observations.

### 2.1.2 Gaussian Process Classification

The GP regression paradigm explicitly assumes a Gaussian likelihood model. Given a Gaussian prior, inference has an analytic and tractable form. However, this model is unsuitable for classification, as the outputs of the classifier are categorical. Applying a non-Gaussian likelihood model, on the other hand, breaks the tractability of the GP framework. Consequently, GP classifiers employ one of several approximation methods (Rasmussen and Williams, 2006).

GP classifiers use a discriminative classification model. Given a set of possible classes $y = \mathcal{C}_1, ..., \mathcal{C}_C$, the classifier is defined by the conditional distribution $p(y|\mathbf{x})$. Modelling $p(y|\mathbf{x})$ follows the same approach taken by logistic regression. Meaning $p(y|\mathbf{x}) = \sigma\Big(f(\mathbf{x})\Big)$, where $f(\mathbf{x}) \in (-\infty, \infty)$ is a latent model and $\sigma(f)$ is

a "squashing" function which constrains the classifier output to a valid probability distribution $[0, 1]$ range. This two-step process for a binary classifier is depicted in Fig. 2.2. First, the categorical data is fitted into a latent regression model as shown in Fig. 2.2a. In a binary case, observations can be defined as $C_{-1} = -1$ and $C_1 = 1$. The latent function, in black, follows standard GP regression, producing $p(f^*|X, y, \mathbf{x}^*)$. Class label prediction, shown in Fig. 2.2b, is then obtained by squashing the latent GP output to a $[0, 1]$ range using a deterministic function, $\sigma(f)$. A common choice for $\sigma(f)$ is the normal cumulative distribution function, $\Phi_G(f)$ which is also known as the *probit* model, or the *logit* function, $\sigma(f) = \Big(1 + \exp(-f)\Big)^{-1}$.

The squashing function $\sigma(f)$ constrains the output of the GP into a valid $[0, 1]$ range. However, $\sigma(f)$ is a deterministic point estimate of class label, which does not consider the GP's model probability distribution. A full Bayesian treatment of the two-step classification requires to marginalise over the GP's latent function distribution. We starts by defining the distribution of the latent function at a query location $\mathbf{x}^*$ as:

$$p(f^*|X, y, \mathbf{x}^*) = \int p(f^*|X, \mathbf{x}^*, f)p(f|X, y)df. \tag{2.7}$$

Equation (2.7) is the latent-function interpretation for GP inference (Rasmussen and Williams, 2006). In GP regression, $p(f|X, y)$ is the posterior distribution of latent function given data $\mathcal{D}$, which is assumed Gaussian. However in classification, $p(f|X, y)$ is no longer Gaussian. Given a distribution on the latent function values, class prediction is obtained through the expectation:

$$\pi^* \equiv p(y^* = C_1|X, y, \mathbf{x}^*) = \int \sigma(f^*)p(f^*|X, y, \mathbf{x}^*)df^*. \tag{2.8}$$

Since $p(f|X, y)$ and $\sigma(f)$ are non-Gaussian, Eqs. (2.7) and (2.8) lose the analytical tractability of the GP regression model, and as a result, these integrals must be approximated. The most straightforward approach relies on Monte-Carlo numerical approximation (Neal, 1998), although such methods incur a cubic computational complexity. To reduce complexity, GP classification typically relies on one of the following approximation methods.

### The Laplace Approximation

The Laplace approximation (Rasmussen and Williams, 2006) fits a Gaussian approximation $q(f|X, y)$ to the non-Gaussian posterior $p(f|X, y)$ of Eq. (2.7). This approximation takes advantage of the fact that a second order Taylor expansion

of the log posterior $\log\Big(p(f|X,y)\Big)$ is a Gaussian:

$$q(f|X,y) \sim \mathcal{N}(f|\hat{f}, A^{-}1). \tag{2.9}$$

Here, $\hat{f} = \text{argmin}_f\, p(f|X,y)$ is obtained iteratively by maximising the posterior. $A^{-1}$ is the Hessian term of the posterior and can be derived from the un-normalised posterior $p(f|X,y) \approx p(y|f)p(f|X)$:

$$A = \nabla\nabla \log\Big(q(f|X,y)\Big) = \nabla\nabla \log\Big(p(y|f)\Big) - K^{-1}, \tag{2.10}$$

where the likelihood term $p(y|f)$ depends on the choice of "squashing" function.

### Expectation Propagation

The *expectation propagation* (EP) method (Minka, 2001) is a general sequential moment matching approximation method. To apply EP to GP classification, the posterior of the latent function $f$ is approximated by a Gaussian distribution. In order to maintain tractability, EP perform a sequential approximation based on a single observation at a time. Using Bayes' rule the posterior distribution over $f$ is as follows:

$$p(f|X,y) = \frac{1}{Z} p(f|X)p(y|f), \tag{2.11}$$

where $Z$ is a normalisation term. Since $p(y|f)$ is a non-Gaussian likelihood function, the posterior is approximated using a Gaussian distribution, $q(f|x,y) = \mathcal{N}(\mu, \Sigma)$. The parameters $\mu$ and $\Sigma$ can be found by minimising the Kullback-Leibler (KL) divergence $KL\Big(p(f|X,y)||q(f|X,y)\Big)$. However, since this is also an intractable computation EP factorises the posterior and approximates each factor $q_i$ using a Gaussian:

$$q(f|X,y) \approx \frac{1}{Z_{EP}} p(f|X) \prod_{i=1}^{N} \mathcal{N}(f_i|\mu_i, \sigma_i). \tag{2.12}$$

The parameters of each factor, $\mu_i$ and $\sigma_i$ are found by moment matching $q_{-1}(f_i)p(y_i|f_i)$, where $q_{-1}(f_i)$, also known as the cavity distribution, is the distribution on all cases except $i$. The computational cost of updating factor parameter is quadratic $\mathcal{O}(N^2)$, hence updating all factors requires $\mathcal{O}(N^3)$.

### Probabilistic Least-Squares Classifier

The *probabilistic least-squares classifier* (PLSC) is an highly tractable approximation to the classification problem. While other GPCs approximate the non-Gaussian posterior of a classifier, PLSC treats it as a regression problem as-

suming a Gaussian noise model. While such an assumption is not obvious for classification, experimental results (Rifkin and Klautau, 2004) show comparable performance to SVM.

The goal of the PLSC is to estimate $p(C|\mathbf{x})$ using a function $f$. Following the SVM literature, we define labels $C_+ = +1$ and $C_- = -1$ with corresponding probabilities $p(C_+|\mathbf{x})$ and $p(C_-|\mathbf{x})$, respectively. For any given $\mathbf{x}$, the classifier's output is a Bernoulli random variable with a probability $q \equiv p(C_+|\mathbf{x})$ and $p(C_-|\mathbf{x}) = 1 - q$. Hence, the squared error is $E(f) = q(f-1)^2 + (1-q)(f+1)^2$, which is minimised when $f = 2q - 1$. This result provides asymptomatic guarantees that the output of the PLSC estimates $p(C|\mathbf{x})$ correctly (Rifkin and Klautau, 2004).

To represent $p(C|\mathbf{x})$, the approximating function $f(x)$ should be sufficiently flexible. The desired flexibility is attained by extending the least-squares classifier using the kernel trick, as done in SVMs (Poggio and Girosi, 1990; Suykens and Vandewalle, 1999). While the output $f$ of a GP regressor lies in the range $(-\infty, \infty)$, a probabilistic output $[0, 1]$ is achieved by post-processing $f$ through a squashing sigmoid (Platt, 1999). Since $f$ is modelled by a GP, $f(\mathbf{x}^*) \sim \mathcal{N}(f|\mu(\mathbf{x}), \sigma^2(\mathbf{x}))$. Using a Gaussian sigmoid, the probability of predicting a class label $C^* = C_+$ at $\mathbf{x}^*$ is:

$$p(C^* = C_+|\mathbf{x}^*, X, y, \alpha, \beta) = \int \Phi_G\Big( + 1(\alpha f + \beta)\Big)\mathcal{N}\Big(f|\mu(\mathbf{x}^*), \sigma^2(\mathbf{x}^*)\Big)df, \quad (2.13)$$

where $\alpha$ and $\beta$ are hyper-parameters of the classifier. The integral of Eq. (2.13) can be computed in closed-form as:

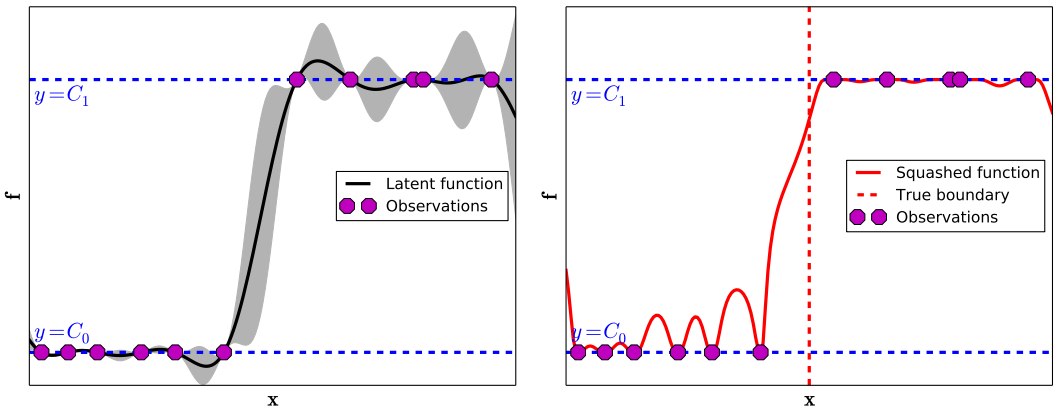
$$p(C^* = C_+|\mathbf{x}^*, X, y, \alpha, \beta) = \Phi\left(\frac{+1(\alpha\mu(x^*) + \beta)}{\sqrt{1 + \alpha^2\sigma^2(x^*)}}\right). \quad (2.14)$$

### 2.1.3 Covariance Functions

Previous sections reviewed the use of GPs in regression and classification problems. In either case, a latent Gaussian model $p(f|X, y)$ is inferred from the data. The covariance function is the heart of that inference process.

Formally, a GP is fully defined by its mean $m(\mathbf{x})$ and covariance $k(\mathbf{x}, \mathbf{x}')$ functions. While $m(\mathbf{x})$ depends only on location $\mathbf{x}$, the covariance function specifies the similarity between two inputs. This abstract notion of similarity is fundamental for GP learning. The choice of covariance function encodes prior assumptions on the underlying process, such as smoothness and differentiability.

The covariance function $k$ defines the covariance matrix $K$. Given a set of




(a) Latent function. (b) Squashed classifier model, $p(y|\mathbf{x})$.

Figure 2.2: A GP binary classifier example. A GP classifier uses two-step process to produce a valid discriminative classification model, $p(y|\mathbf{x})$. (a) A latent regression model is produced based only the categorical observations. (b) A squashing function transforms the output of the latent regression step from a $(-\infty, \infty)$ domain to a valid probability in the range $[0, 1]$.

input points $X = \{x_i | i = 1, ..., N\}$, one can define the covariance matrix as an $N \times N$, symmetric and real-valued matrix, where each element $K_{i,j} = k(x_i, x_j)$. A valid covariance function is positive semi-definite, i.e. $K$ satisfies $v^T K v \geq 0$ for all vectors $v \in \mathbb{R}^D$. The rich library of covariance functions available in the literature (Rasmussen and Williams, 2006) can be categorically grouped into stationary and non-stationary functions. A stationary covariance function is invariant to translation in the input space, meaning it is only a function of $\mathbf{r} \equiv |\mathbf{x} - \mathbf{x}'|$. Non-stationary covariance functions (Rasmussen and Williams, 2006) employ a more general dependence in $\mathbf{x}$ and $\mathbf{x}'$. This thesis only utilises stationary kernels, such as the ones described next and visualised in Fig. (2.3).

Stationary covariance functions only depend on $\mathbf{r}$. However, the design of covariance function allows for additional free parameters, also called hyper-parameters. Most stationary covariance functions use at least two hyper-parameters, $\ell$ and $\sigma_f$. $\sigma_f^2$ is the intrinsic signal variance, which indicates the maximum variance a GP can produce[1].

The lengthscale $\ell$ can be considered as a characteristic distance of influence. In effect, $\ell$ scales $\mathbf{r}$ into effective distance of influence $\mathbf{r}_\ell = \frac{\mathbf{r}}{\ell}$. When using a single $\ell$ value for $D$ dimensions of $\mathbf{r}$, the kernel is considered isotropic. However, in most cases only part of the input $\mathbf{r}$ affects the covariance, which can not be captured by a single $\ell$. Therefore, a more general approach for scaling $\mathbf{r}$ is used:

$$\mathbf{r}_\ell^2(\mathbf{x}, \mathbf{x}') = (\mathbf{x} - \mathbf{x}')^T M (\mathbf{x} - \mathbf{x}'), \tag{2.15}$$

[1]Under the assumption of zero target noise

where $M$ is positive semidefinite matrix. *Automatic relevance determination* (ARD) (Neal, 1996) uses diagonal $M = \text{diag}(\ell^{-2})$, where $\ell$ is a vector of positive values. In effect, ARD assigns different lengthscales for various inputs.

#### Radial Basis Function

The *radial basis function* (RBF), also known as *squared exponential* (SE) kernel, is widely-used within machine learning. the RBF kernel is infinitely differentiable, which leads to very smooth functions, as shown in Fig. (2.3). Its general form is:

$$k(\mathbf{r}_\ell) = \sigma_f \exp\left(\frac{-\mathbf{r}_\ell^2}{2}\right), \tag{2.16}$$

#### Matérn class

This is a family of covariance functions with the general form (Rasmussen and Williams, 2006):

$$k_\nu(\mathbf{r}_\ell) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\sqrt{2\nu}\mathbf{r}_\ell\right)_\nu^K\left(\sqrt{2\nu\mathbf{r}_\ell}\right), \tag{2.17}$$

where $\nu$ is a non-negative parameter of the covariance function and $K_\nu$ is a modified Bessel function of the second kind.

For several values of $\nu$, the Matérn covariance function take a simpler form;

$$k_{\nu=1/2}(\mathbf{r}_\ell) = \exp(-|\mathbf{r}_\ell|), \tag{2.18}$$

which forms a non-differentiable covariance.

$$k_{\nu=3/2}(\mathbf{r}_\ell) = (1+\sqrt{3}\mathbf{r}_\ell)\exp(-\sqrt{3}|\mathbf{r}_\ell|), \tag{2.19}$$

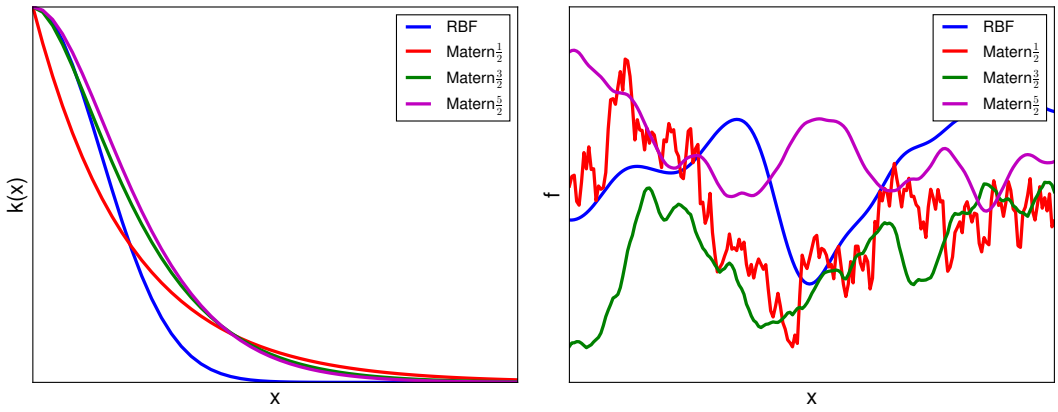
which is once differentiable, and

$$k_{\nu=5/2}(\mathbf{r}_\ell) = (1+\sqrt{5}\mathbf{r}_\ell + \frac{5}{3}\mathbf{r}_\ell^2)\exp(-\sqrt{5}|\mathbf{r}_\ell|), \tag{2.20}$$

which is twice differentiable. Figure 2.3a provides a comparison between these covariance functions, and the implication on the differentiability of the process.

### 2.1.4 Parameter Learning

A crucial aspect of GP is model selection, as this determines how well the model can generalise and fit the data. This is typically a semi-automatic process. Some properties, such as the covariance function(s), is often chosen a-priori by the user. Other properties, such as the hyperparameters $\theta$, are learned from the available data. The two most commonly used methods for training a model are to maximise




(a) Covariance functions. (b) Sampled functions, $p(y|\mathbf{x})$.

Figure 2.3: The covariance function defines the properties of a GP, such as differentiability and smoothness. (a) Stationary covariance functions. (b)Random functions sampled from a GP prior based on the covariance functions shown in (a).

the marginal likelihood over the data or perform cross-validation (Rasmussen and Williams, 2006).

### Marginal Likelihood

The marginal likelihood, often referred to as evidence, is defined as $p(Y|X,\theta)$. It is used in Bayesian model selection, as it automatically trades-off between the model fit and its complexity. The general integral form of the marginal likelihood is often intractable. However, in a GP, the log of the marginal likelihood takes a closed-form:

$$\log\Big(p(Y|X,\theta)\Big) = -\frac{1}{2}Y^T K^{-1}Y - \frac{1}{2}\log(|K|) - \frac{N}{2}\log(2\pi). \tag{2.21}$$

Here, the matrix $K = k(X,X|\theta)$ and $|K|$ is its determinant. Training of the hyperparameters is then defined as the optimisation of Eq. (2.21):

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \log\Big(p(Y|X,\theta)\Big). \tag{2.22}$$

One can identify two explicit parts to Eq. (2.21), the data fit component $Y^T K^{-}1Y$ and complexity term $\log(|K|)$ that depends solely on the covariance function. Optimising of their sum in Eq. (2.22) balances model fit and complexity, as desired in Bayesian model selection.

### Cross-validation

*Cross-validation* (CV) (Bishop, 2006), often also referred to as $S$-fold CV, is a common method for model training in machine learning. $S$-fold CV partitions the data into $S$ disjoint sets. It uses an iterative procedure that uses $S-1$ sets to train the model, which is then evaluated on the remaining set. This process is repeated $S$ times with CV alternates the sets used for training and validation. *leave-one-out cross-validation* (LOOCV) is a special case of $S$-fold CV where $S$ is the size of the dataset, $S = N$.

Using cross-validation for GP model selection takes advantage of the tractable nature of the normal distribution, where the predictive probability is given in closed-form (Rasmussen and Williams, 2006). Leaving out the $i^{th}$ training point in $\mathcal{D}$, the predictive log probability becomes:

$$\log\Big(p(y_i|X, Y_{-i}, \theta)\Big) = -\frac{(y_i - \mu_i)^2}{2\sigma_i^2} - \frac{1}{2}\log(\sigma_i^2) - \frac{1}{2}\log(2\pi), \tag{2.23}$$

where $Y_{-i}$ indicates all targets in $Y$ except of the $i^{th}$ observation. When computed over the entire dataset, the LOOCV log predictive probability is given by:

$$L_{LOOCV}(X, Y, \theta) = \sum_{i=1}^{N} \log\Big(p(y_i|X, Y_{-i}, \theta)\Big). \tag{2.24}$$

$L_{LOOCV}$ can be considered as a pseudo-likelihood function, which depends on the hyperparameters $\theta$. Hence, it can be used as a loss during model selection. Training is, then, done by maximising $L_{LOOCV}$:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, L_{LOOCV}(X, Y, \theta). \tag{2.25}$$

## 2.2 Bayesian Optimisation

*Bayesian optimisation* (BO) (Brochu et al., 2010) is a powerful global optimiser, which is most effective when the objective function does not have a closed-form expression, is costly to evaluate, and there is no access to derivative information. Given a limited set of noisy observations and prior beliefs about the properties of the objective function, BO exploits Bayes' theorem to determine the most effective course of action.

A motivating example for BO can be found in the field of experimental design, such as in the development of a new drug. There are many factors that affect the potency of a drug; for instance its ingredients and manufacturing process. These factors, and their cross-interactions, form an enormous and complex set of potential drugs. However, the potency of each drug is a-priori unknown. As-

sessing a new drug is expensive in both time and money. Therefore, it is not feasible to explore the entire space of possibilities. BO provides a mechanism to address this problem. It uses the observations from previous experiments to build a probabilistic model of the problem at hand. It then suggests additional experiments that efficiently explore the space of possibilities. Consequently, the number of experiments required in order to find the optimum is reduced.

### 2.2.1 Unconstrained optimisation

Formally, the goal of BO is to find the extremum of a function $f$, and thus can be written as the following minimisation problem:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}}\, f(\mathbf{x}). \tag{2.26}$$

$f$ is a black-box function, meaning its exact form is unknown to the optimiser. Moreover, the only knowledge of $f$ is obtained from a finite set of noisy observations. With only a limited access to $f$, BO replaces Eq. (2.26) with an iterative surrogate optimisation process. The building blocks of this process are the surrogate and acquisition functions, which will be described next.

The surrogate function is the estimated model of the objective function $f$. It holds our current belief of the underlying function, which is inferred from observations and prior knowledge of its properties. Gaussian processes (GPs) are generally used for modelling the surrogate function due to their Bayesian non-parametric properties and analytical form. When modelled using a GP, the surrogate function is represented by the posterior mean and variance.

The acquisition function $s(\mathbf{x})$ guides the selection of new observation points to sample from the unknown objective function. Based on the current surrogate model of the objective function, it provides a quantitative measure for the probability of finding the global extremum in a specific location. The Bayesian optimiser uses this measure as a utility proxy to select the next observation point. In essence, the original optimisation problem, Eq. (2.26), is transformed into an iterative optimisation process over the acquisition function with the aim of finding the next $n^{th}$ sampling point:

$$\mathbf{x}_n = \underset{\mathbf{x}}{\operatorname{argmin}}\, s(\mathbf{x}). \tag{2.27}$$

The pseudo code shown in Algorithm 1 outlines the typical steps performed by BO. In each iteration, a new sampling location, $\mathbf{x}_n$, is found by minimising the acquisition function $s(\mathbf{x})$. BO evaluates the objective function, $f$, at $\mathbf{x}_n$ and checks whether a new extremum has been found. In addition, the new

**Algorithm 1:** Bayesian Optimisation

**Input:** $f(\mathbf{x})$: Objective function.
$s(\mathbf{x})$: Acquisition function.
$f^*_{min}$: Current minimum.
**Output:** $\mathbf{x}_{min}$
**for** $n = 1, 2, 3, ...$ **do**
  Find: $\mathbf{x}_n \leftarrow \underset{\mathbf{x}}{\text{argmin}}\ s(\mathbf{x})$
  Sample objective function: $f_n \leftarrow f(\mathbf{x}_n)$
  Update GP model with new observation $(\mathbf{x}_n, f_n)$
  **if** $f_n < f^*_{min}$ **then**
    $\mathbf{x}_{min} \leftarrow \mathbf{x}_n$
    $f^*_{min} \leftarrow f_n$
  **end**
**end**

observation, $f_n$, updates the surrogate (GP) model, which holds our belief of $f$. The updated model is then used on the next iteration of BO. By choosing an appropriate GP model and acquisition function, BO keeps the number of function evaluations low, leading to an efficient optimisation process.

A one dimensional example of BO is depicted in Fig. 2.4. The optimiser has no knowledge of the objective function (blue line) other than the noisy samples (red asterisks). A GP model is generated based on these observations. The model is accurate and confident around the sampling points, where the posterior mean (black dashed line) converges to the objective function values and its variance (grey shade) is low. Initially, the LCB acquisition function resembles the GP variance, which leads to an aggressive exploratory behaviour at the beginning of the optimisation. As the model becomes more confident, the optimiser focuses its search around the global minimum.

Although Eq. (2.27) is still a non-convex optimisation problem, using an appropriate acquisition function makes the search for the extremum more efficient. A proper acquisition function balances the exploration-exploitation trade-off. Therefore, the optimisation process considers areas where it believes the extremum lies as well as promising location in unexplored regions. Consequently, the number of expensive evaluations of the objective function is kept to a minimum. Brochu et al. (2010) list the most commonly used acquisition functions. Based on the predictive mean $\mu$ and variance $\sigma^2$ defined in Eqs. (2.5) and (2.6), the acquisition functions take the following analytic form:

1. *Expected Improvement* ($EI$). $EI$ is defined as the expected difference from the true extremum. On its $n^{th}$ iteration, the optimiser finds a location that maximises the expected difference from the true extremum, $f(\mathbf{x}_{min})$. In a

minimisation problem, finding $f(\mathbf{x}_{min})$, $EI$ is defined as follows:

$$EI(\mathbf{x}) = \mathbb{E}\{f(\mathbf{x}) - f(\mathbf{x}_{min})\}. \tag{2.28}$$

We follow a slightly modified version of $EI$ that uses the predicted mean, $f^*_{min}$, which is inferred from the GP model (Gramacy and Lee, 2011). Furthermore, we exploit the GP structure to produce a concise closed-form for $EI$:

$$EI(\mathbf{x}) = \begin{cases} -\sigma(\mathbf{x})[Z\Phi_G(Z) + \phi_G(Z)] & \sigma(\mathbf{x}) > 0 \\ 0 & \sigma(\mathbf{x}) = 0, \end{cases} \tag{2.29}$$

where $\phi_G$ and $\Phi_G$ represent the normal distribution PDF and CDF, respectively. $\sigma(\mathbf{x})$ is the standard deviation of the posterior distribution in $\mathbf{x}$. $Z$ is given by

$$Z = \begin{cases} (f^*_{min} - \mu(\mathbf{x}) - \zeta)/\sigma(\mathbf{x}) & \sigma(\mathbf{x}) > 0 \\ 0 & \sigma(\mathbf{x}) = 0 \end{cases},$$

where $\zeta$ is a user-defined parameter that balances the exploration-exploitation trade-off.

2. *Lower Confidence Bound* ($LCB$). The $LCB$ lacks the rigour of $EI$. However, its user-defined parameter $\kappa$ provides a simple mechanism to adjust the exploration-exploitation trade-off:

$$LCB(\mathbf{x}) = \mu(\mathbf{x}) - \kappa\sigma(\mathbf{x}). \tag{2.30}$$

Srinivas et al. (2010) propose a schedule for $\kappa$ that provides no-regret bounds for BO. In this work, however, we follow the work of (Marchant and Ramos, 2014), and fix $\kappa$.

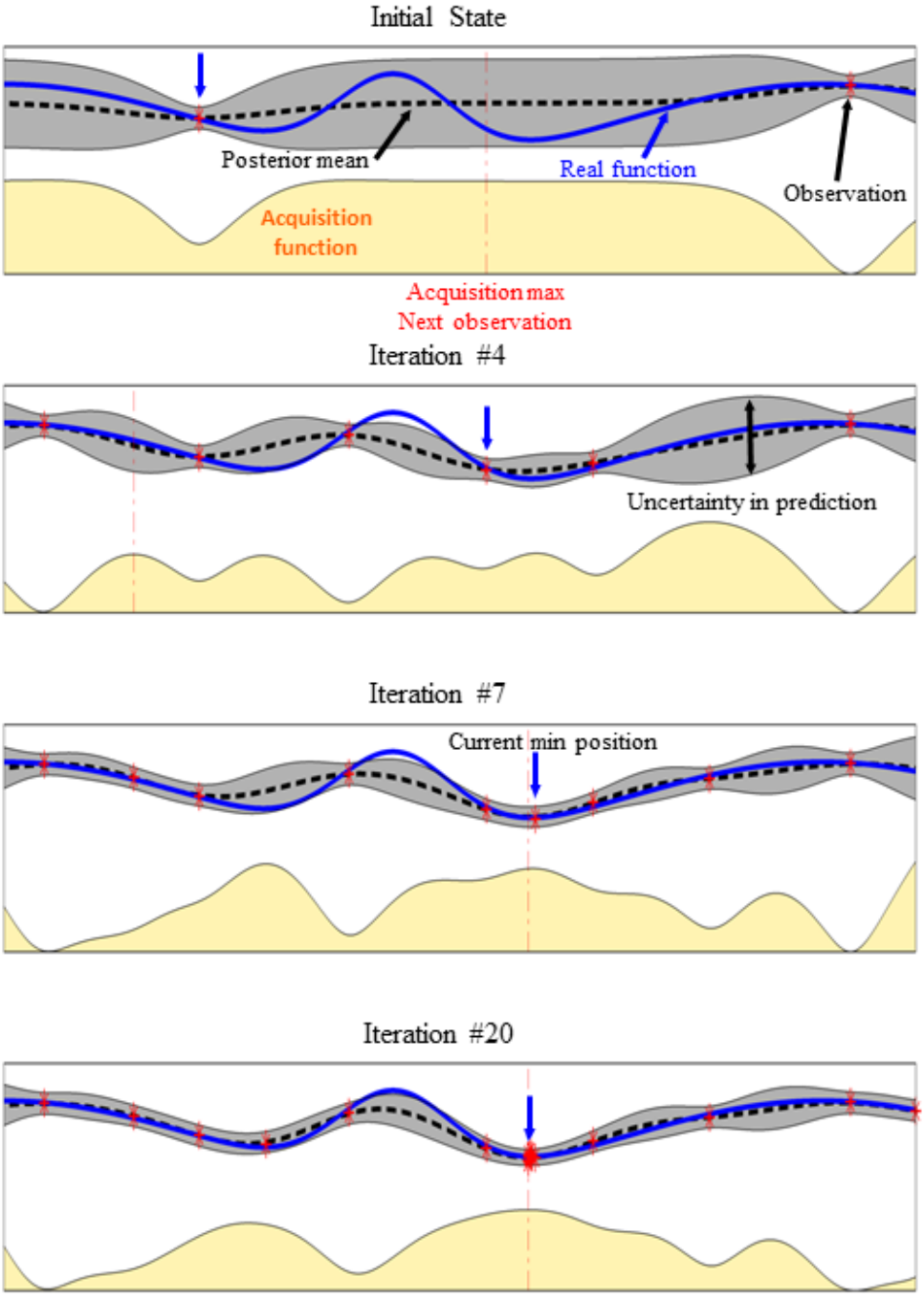



Figure 2.4: One dimensional example of BO. The continuous blue line is the unknown objective function. The red asterisks are samples (with added noise) of this function. The black dashed line and shade represent the posterior GP mean and variance calculated from samples, respectively. The yellow shade is the acquisition function (LCB) which is scaled and with an offset for visualisation purposes. The red vertical dash-dot line represents the next sampling locations, while blue downwards arrow marks the location on the current minimum.

### 2.2.2 Constrained BO

Constrained BO handles optimisation with unknown constraints. Similarly to the optimisation process discussed in the previous section, our only knowledge of the underlying constraints stems from observations. Furthermore, the objective function is undefined wherever the constraints are violated. To tackle the uncertainty in both objective function and constraints, we employ a constraint weighted acquisition function (Gelbart et al., 2014). Consequently, the optimisation balances the expected reward with the confidence in the constraint model and its associated risk.

In the literature, there are two different modifications to the basic BO acquisition functions relevant to our case. Gramacy and Lee (2011) propose a variant of $EI$ called *integrated expected conditional improvement* ($IECI$). $IECI$ represents the marginal effect that a new observation will have on the overall uncertainty of the GP model regardless of its actual value and is defined as

$$IECI(\mathbf{x}) = \int \mathbb{E}\{EI(\mathbf{x}'|\mathbf{x})\}c(\mathbf{x}')d\mathbf{x}' \tag{2.31}$$

where $p(\mathbf{x}')$ is an arbitrary probability density function. To incorporate constraints, Gramacy and Lee integrated the conditional improvement with $p(\mathbf{x}')$ as the probability density of the constraint. The main caveat of this method is its scalability. Calculating the integral over the expected conditional improvement requires heavy Monte Carlo sampling of the GP model. Hence, $IECI$ is not a practical method for real-time problems. Furthermore, this method might not be suitable for most standard constrained optimisation problems since it assumes that the objective function can be sampled in regions where the constraint is violated.

The other modification to BO, which we use in this work, is the constraint weighted acquisition function proposed by Gelbart et al. (2014). The confidence in the validity of the solution scales the expected utility of the acquisition function. With independent constraints $\Lambda_k$, a Constrained $LCB$ ($CLCB$) is thus defined as:

$$CLCB(\mathbf{x}) = LCB(\mathbf{x}) \prod_{k=1}^{K} \Pr(\Lambda(\mathbf{x})) < 1 - \delta_k, \tag{2.32}$$

where $\delta_k$ is a user defined constrained confidence bound over constraint $k$.

In order to stochastically model the constraints $\mathcal{C}_k(\mathbf{x})$, we employ GPCs, $g_k$, to provide an estimate for the likelihood constraint $k$ is satisfied within the user

**Algorithm 2:** Constrained Bayesian Optimisation

**Input:** $f(\mathbf{x})$: Objective function.
$s(\mathbf{x})$: Acquisition function.
$g_k(\mathbf{x})$: $k$-th constraint function.
$\delta_k$: $k$-th constraint tolerance.
$f^*_{min}$: Current minimum.
**Output:** $\mathbf{x}_{min}$
**for** $n = 1, 2, 3, ...$ **do**
  feasible region $\mathcal{C}$: $\mathcal{C}(\mathbf{x}) = \prod_{k=1}^{K} \Pr(g_k(\mathbf{x}) < 1 - \delta_k)$
  Find next sample point:
  $\mathbf{x}_n \leftarrow \underset{\mathbf{x}\in\mathcal{C}}{\operatorname{argmin}}\ s(\mathbf{x}) \prod_{k=1}^{K} \Pr(g_k(\mathbf{x}) < 1 - \delta_k)$
  **if** $\mathbf{x}_n$ *valid* **then**
    Sample objective function: $f_n \leftarrow f(\mathbf{x}_n)$
    Update GP model with new observation $(\mathbf{x}_n, f_n)$
    If $f_n < f^*_{min}$: $\mathbf{x}_{min} \leftarrow \mathbf{x}_n$
  **end**
  Update GPCs with new observation
**end**

defined confidence bounds $\delta_k$:

$$CLCB(\mathbf{x}) = LCB(\mathbf{x}) \prod_{k=1}^{K} \Pr(g_k(\mathbf{x}) < 1 - \delta_k). \tag{2.33}$$

Incorporating learned constraints complicates the optimisation algorithm as can be seen from the pseudo code in Algorithm 2. Since the objective function is undefined in regions where the constraints are not satisfied, a preprocessing step finds feasible and valid regions. Within these regions, the optimiser finds the next sampling location using the utility function defined in Eq. (2.33). With every new observation point, the constraints are assessed and their respective GPC model is updated. The GP model for the goal function, on the other hand, is only updated when all constraints are met.

A one dimensional example for constrained BO is shown in Fig. 2.5. The unknown constraint is indicated by the area shaded in green, while its predictive probability is represented by the blue area. As with the regression of the objective function, the confidence in the constraints value, whether valid or invalid, is higher around observations. As evident from Fig. 2.5, BO tries to evaluate points outside the constrained region, however this only updates the constraint model while the objective GP model is unchanged (hence uncertainty is high). With every observation, BO becomes more confident in the model of the objective function, borders of the constraint, and the location of the global minimum.

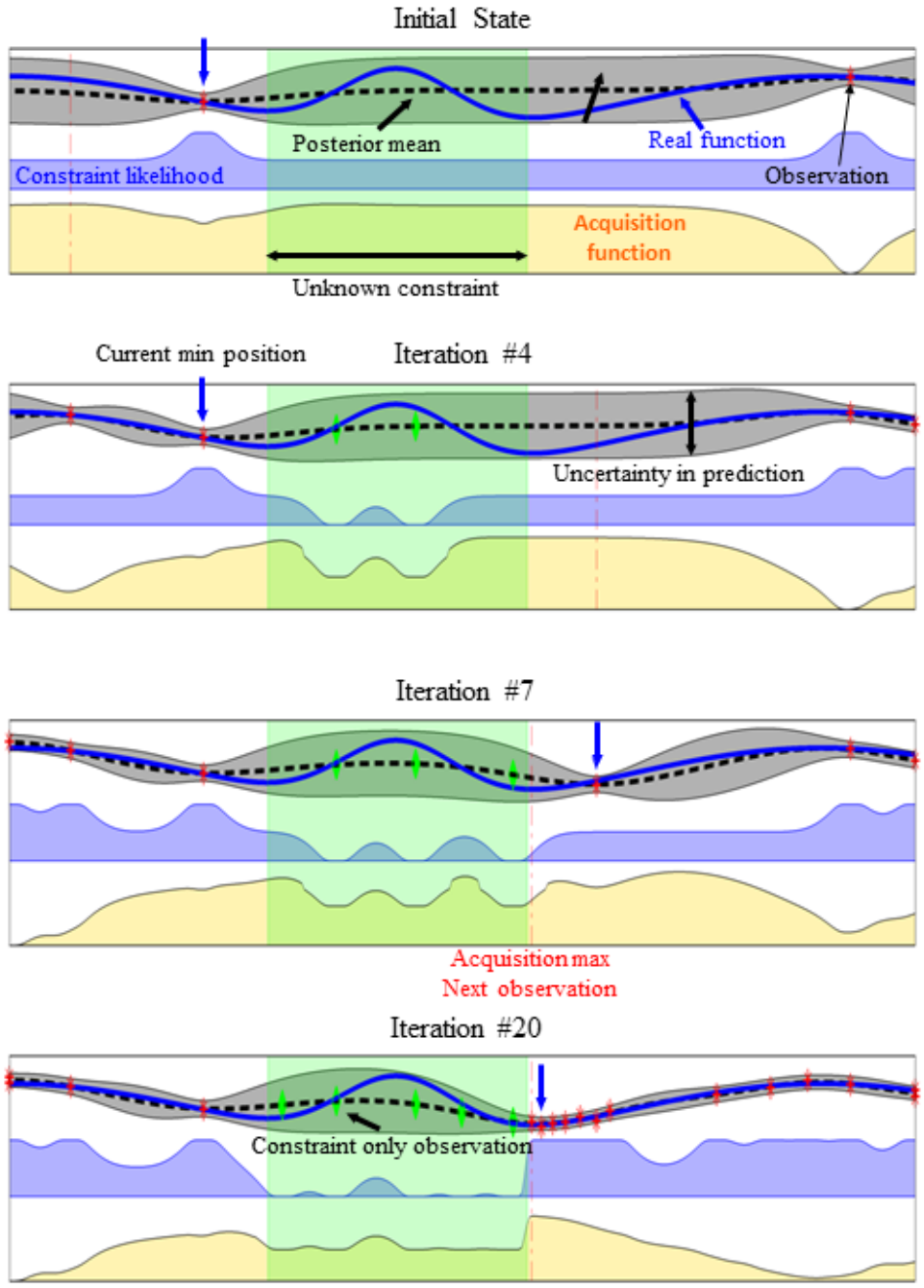



Figure 2.5: One dimensional example of BO with an unknown constraint. The continuous blue line is the unknown goal function and the green area indicates the unknown constraint. The green diamonds are observations of violated constraint, hence the goal function can not be sampled. The red asterisks are samples (with added noise) of the goal function, where the constraint is met. The black dashed line and shade represent the posterior GP mean and variance calculated from samples, respectively. The area in blue is the constraint likelihood function, where high values stand for high probability that the constraint will be satisfied. The yellow area is the basic unconstrained acquisition function ($LCB$). Both acquisition and constraint likelihood functions are scaled and with an offset for visualisation purposes. The red vertical dash-dot line represents the next sampling locations, while blue downwards arrow marks the location on the current minimum

## 2.3 Functional Gradient Descent Optimisation

*Functional gradient descent* (FGD) is a variational framework to optimise non-linear models. It has been successfully applied to motion planning problem in recent years with the main objective of producing safe, collision-free paths.

We first introduce notation. A path, $\xi : [0,1] \rightarrow C \in \mathbb{R}^D$, is a function that maps a time-like parameter $t \in [0,1]$ into configuration space $\mathcal{C}$. We define an objective functional $\mathcal{U}(\xi) : \Xi \rightarrow \mathbb{R}$ that returns a real number for each path $\xi \in \Xi$. The objective functional is used in the optimisation process to capture the path optimisation criteria such as smoothness and safety. The goal of the optimisation process is to find a $\xi$ that minimises the overall costs:

$$\xi_{optimal} = \underset{\xi}{\operatorname{argmin}} \mathcal{U}(\xi). \tag{2.34}$$

Regardless of the exact choice of cost functional $\mathcal{U}(\xi)$, $\xi$ can be optimised by following the functional gradient. Similar to other gradient descent methods, optimisation is performed iteratively. The functional gradient update rule is derived from a linear approximation of the cost functional around the current trajectory, $\xi_n$:

$$\mathcal{U}(\xi) \approx \mathcal{U}(\xi_n) + \nabla_\xi \mathcal{U}(\xi_n)(\xi - \xi_n), \tag{2.35}$$

where $n$ indicates the iteration number. To ensure convexity of the objective function, we add a regularisation term based on the norm of the update:

$$\xi_{n+1} = \underset{\xi}{\operatorname{argmin}}\ \mathcal{U}(\xi_n) + (\xi - \xi_n)^T \nabla_\xi \mathcal{U}(\xi_n) + \frac{1}{2\eta_n} \|\xi - \xi_n\|_A^2. \tag{2.36}$$

The regularisation term $\|\xi - \xi_n\|_A^2 = (\xi - \xi_n)^T A (\xi - \xi_n)$ is the squared norm with respect to a metric tensor $A$ and is used to prevent over-confident updates. $\eta_n$ is a user-defined learning rate which must satisfy the Robbins-Monro conditions (Robbins and Monro, 1951); $\sum_{n=1} \eta_n^2 < \infty$ and $\sum_{n=1} \eta_n = \infty$. A closed form solution of Eq. (2.36) is derived by differentiating the right hand side of Eq. (2.36) with respect to $\xi$ and setting it to zero, yielding:

$$\xi_{n+1} = \xi_n - \eta_n A^{-1} \nabla_\xi \mathcal{U}(\xi_n). \tag{2.37}$$

The form of the update rule in Eq. (2.37) is general, and thus, invariant to the choice of the objective function $\mathcal{U}(\xi)$ or the solution space representation. The only requirements are that $A$ is invertible and the gradient $\nabla_\xi \mathcal{U}(\xi_n)$ exists.

The general rule for computing the objective functional gradient $\nabla_\xi \mathcal{U}$ originates from the calculus of variations. As a variational method, the objective functional

must take the form of an integral or sum. Generally, the objective functional $\mathcal{U}$ takes the form $\mathcal{U}(\xi) = \int_a^b v(t, \xi, \xi')dt$, therefore the functional gradient can be computed using the Euler-Lagrange equation (Zucker et al., 2013):

$$\nabla \mathcal{U}_\xi(\xi) = \frac{\partial v}{\partial \xi} - \frac{d}{dt}\frac{\partial v}{\partial \xi'}. \tag{2.38}$$

These gradients are then used to compute the iterative update rule of Eq. (2.37).

## 2.4 Occupancy Maps

Occupancy maps are an invaluable tool for autonomous navigation and planning. In its simplest form, it provides an occupied or free classification for any region of the map based on sensors observations. There are various mapping methods, each with its own way to incorporate observations and assumptions on the properties of the map. However, the most common approach, which has been extensively used in the last several decades, is grid mapping (Elfes, 1989). An example of the difference between occupancy representations is shown in Figure 2.6, which plots a grid map and a Hilbert map generated using the Intel-Lab dataset

Formally, mapping is an inference problem where given a set of noisy observations $\mathbf{z}$ taken at partially known robot poses $\mathbf{x}$, the posterior distribution over the map $\mathbf{m}$ is given by

$$p(\mathbf{m}|\mathbf{z}, \mathbf{x}). \tag{2.39}$$

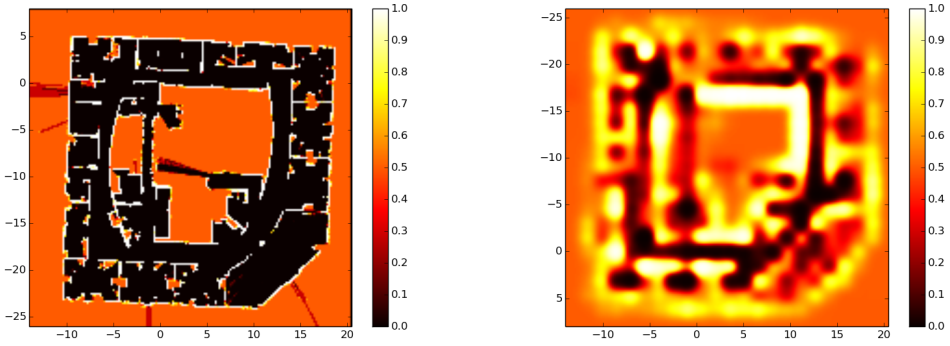
The posterior models the hypothesis of occupancy over the entire map. Unfortunately, solving the general mapping inference problem is computationally intractable due to the dimensionality of the problem. For example, even if a 2-D map is discretised using $|\mathbf{m}|$ cells, each can be either occupied or free, the number of possible maps would be exponential, $2^{|\mathbf{m}|}$.

### 2.4.1 Grid Maps

Grid maps relax Eq. (2.39) by a cell independence assumption (Elfes, 1989). In a grid map, the 2D world is discretised into cells, where each grid cell, $m_i$, is an independent Bernoulli random variable. This assumption simplifies calculations since the map posterior is now a product of the individual cells:

$$p(\mathbf{m}|\mathbf{z}, \mathbf{x}) = \prod_i p(m_i|\mathbf{z}, \mathbf{x}). \tag{2.40}$$

[1] available at http://radish.sourceforge.net/.



(a) Grid map. (b) Hilbert map.

Figure 2.6: Occupancy maps generated using the Intel-Lab dataset. (a) A Grid map forms a discretised occupancy representation using independent cells. (b) Hilbert map uses a logistic regression classifier inferred over an RKHS for continuous occupancy mapping.

This is a strong assumption that ignores the natural dependencies between adjacent cells. Although the spatial correlations are lost in a grid map, the independence assumption is essential for tractability of the model, as the posterior of each cell can now be computed separately. Given a new observation $z_t$ at pose $x_t$, the new posterior for map cell $m_i$ can be derived from Bayes' rule:

$$p(m_i|z_t, \mathbf{z}) = \frac{p(z_t|m_i, \mathbf{z})p(m_i|\mathbf{z})}{p(\mathbf{z}_t|\mathbf{z})}. \tag{2.41}$$

$p(m_i|\mathbf{z})$ is the current map model, which is based on all past observations. The likelihood $p(z_t|m_i, \mathbf{z})$ is computed according to a known sensor model.

To overcome numerical instabilities around $p(m_i|\mathbf{z}) \approx 0, 1$, Eq. (2.41) can be converted to an equation over its log-odds $l_{t,i}$, which is defined by (Thrun et al., 2005):

$$l_{t,i} = \frac{p(m_i|z_t, \mathbf{z})}{1 - p(m_i|z_t, \mathbf{z})}. \tag{2.42}$$

Using the log-odds of the current map model $l_i$, defined similarly to $l_{t,i}$, and an inverse sensor model $\varpi(m_i, x_t, z_t)$ (Thrun et al., 2005), (2.41) transforms into:

$$l_{t,i} = l_i + \varpi(m_i, x_t, z_t). \tag{2.43}$$

$\varpi$ is a user defined function that return a log-odds likelihood value, which considers properties of the sensor, such as its range, resolution and noise.

Figure 2.6a shows a grid map generated using the Intel-Lab dataset. The resulting map forms a confident model of the actual floor plan of that building. However, it relies on sensor noise and localisation errors being negligible. Uncer-

tainty in the robot's pose may lead to a corrupt occupancy map, forming fake or double walls.

### 2.4.2 Hilbert Maps

Hilbert maps are a scalable, fast approach for continuous occupancy mapping that was recently presented by Ramos and Ott (2015). It is a logistic regression classifier inferred over a *reproducing kernel Hilbert space* (RKHS) defined by a kernel $k(\cdot, \cdot)$. To ensure its scalability, Hilbert maps use a finite feature vector, which approximates the inner product defined by $k$, i.e. $k(\mathbf{x}, \mathbf{x}') \approx \hat{\Phi}(\mathbf{x})^T \hat{\Phi}(\mathbf{x}')$.

Traditional occupancy grid maps discretise the map into a fixed grid in order to estimate the occupancy posterior (Elfes, 1989). Although such a procedure ensures tractability, its drawback is the loss of spatial relationship between neighbouring cells. To alleviate this problem, a non-parametric approach based on Gaussian Processes (GPs) was proposed in (O'Callaghan and Ramos, 2012). The *Gaussian processes occupancy map* (GPOM) utilities a GP-based PLSC as an occupancy classifier based on sensor observations. GPOM provides a fully non-parametric approach that maintains a complete covariance matrix between all observations, which results in a highly expressive occupancy map model. As such, GPOM captures spatial relationships, which enables continuous predictions of occupancy. However, GPOM's expressivity comes with a heavy computational complexity, which scales cubically with the number of observations.

Hilbert maps take the advantages of both grid maps and GPOM. Similar to GPOM, the use of kernels retain spatial relationship which enables continuous inference. The computational complexity, on the other hand, depends linearly on the number of features. A Hilbert map is a discriminative model based on the *logistic regression classifier* (LR) that predicts the occupancy at a new query point, $\mathbf{x}^*$. Given a vector of parameters $\mathbf{w}$ the probability that $\mathbf{x}^*$ is occupied is given by:

$$p(y^* = +1|\mathbf{x}^*, \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x}^*)}. \tag{2.44}$$

As occupancy is a binary random variable, the probability of non-occupancy is given by $p(y^* = -1|\mathbf{x}^*, \mathbf{w}) = 1 - p(y^* = +1|\mathbf{x}^*, \mathbf{w})$. We note that Eq. (2.44) is the *logit* sigmoid function, $\sigma(\mathbf{z}_L)$ applied on the linear projection $\mathbf{z}_L = \mathbf{w}^T \mathbf{x}$.

The linear projection of the basic LR classifier cannot capture the complexity of a real environment. To support a richer family of functions, Hilbert maps employ non-linear classification using approximate kernels. The kernel $k(\cdot, \cdot)$ defines a non-linear, and potentially infinite dimensional, mapping $\Phi(\mathbf{x}, \cdot)$ that projects the input into a high dimensional RKHS. The inner product between two features is then $k(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}, \cdot), \Phi(\mathbf{x}', \cdot) \rangle$. To reduce model training time, Hilbert maps

replace the kernel with an approximation, $\hat{\Phi}(\cdot)$ (Shalev-Shwartz et al., 2011). $\hat{\Phi}(\cdot)$ defines a finite feature vector where the dot product of these features can, in expectation, approximate the selected kernel; $k(\mathbf{x}, \mathbf{x}') \approx \hat{\Phi}(\mathbf{x})^T \hat{\Phi}(\mathbf{x}')$. Under these assumptions, the predictive occupancy posterior becomes:

$$p(y^* = +1|\mathbf{x}^*, \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w}^T \hat{\Phi}(\mathbf{x}^*))} \equiv \sigma(\mathbf{z}_{NL}), \tag{2.45}$$

where $\mathbf{z}_{NL}$ denotes the non-linear mapping achieved by the feature vector.

There are several methods to generate features to approximate a kernel (Ramos and Ott, 2015). For the RBF kernel defined by Eq. (2.16), there are three different approximations; Random Fourier features (Rahimi and Recht, 2009), the Nyström method (Williams and Seeger, 2001) and the sparse random features (Melkumyan and Ramos, 2009).

The discriminative model of Eq. (2.45) defines a flexible likelihood model. However, it still need to be fitted to the occupancy data, i.e. learn the weights $\mathbf{w}$. This is, in effect, a maximum-likelihood procedure, where the objective function used to estimate $\mathbf{w}$ is a regularised *negative log-likelihood* (NLL) (Ramos and Ott, 2015) defined by:

$$NLL(\mathbf{w}) = \sum_{i=1}^{N} \log(1 + \exp(-y_i \mathbf{w}^T \hat{\Phi}(\mathbf{x}_i))) + R(\mathbf{w}). \tag{2.46}$$

Here, $\{\mathbf{x}_i, y_i\}_1^N$, contains $N$ occupancy observations, captured by the robot while moving through the environment, where $\mathbf{x}_i \in \mathbb{R}^D$ is a $2D$ or $3D$ position and $y_i \in \{-1, +1\}$ represents observed occupancy at $\mathbf{x}_i$. $R(w)$ is a regularisation term that prevents overfitting.

Training a map requires optimising the NLL objective in Eq. (2.46). In large occupancy datasets, computing the gradients and Hessians for all training points might be restrictive. To overcome this limitation, the optimisation of NLL is performed using stochastic gradient descent (SGD) methods. SGD iteratively updates the weights based on a single data or a small mini-batch, which are selected randomly from the data set. This optimisation process asymptotically converges to the expected risk, while keeping the cost per iteration fixed. We note that unlike GPOM, a Hilbert map provides a maximum likelihood estimate for occupancy, and therefore lacks any measure of uncertainty. A Bayesian variant of Hilbert maps proposed by Senanayake and Ramos (2017) maintains a model uncertainty. However, as it is limited in its ability to incorporate data from high-throughput sensors such as LIDAR, Bayesian HM were not used in this work.

A Hilbert occupancy map of the Intel Lab is shown in Fig. 2.6b. Compared

with the grid occupancy map, the continuous map representation of the Hilbert map generates a smoother transition in occupancy around obstacles, which captures the confidence in the location of that obstacle.

## 2.5 Summary

In this chapter we introduced the background for the various path planning and exploration algorithms developed throughout this thesis. The material presented provides an overview of the key concepts, which will be further discussed, together with the relevant literature review, in the remainder of the thesis.

Bayesian optimisation offers a powerful tool that has been gaining popularity in the robotics community in recent years. Black-box optimisation is a recurrent theme in many robotic problems such as control, localisation, reinforcement learning, and many more. As BO offers a structured framework for global optimisation of such problems, its appeal for robotic application is clear. To better the robustness of BO, especially for autonomous planning tasks, the optimisation process must handle constraints, including those that are a-priori unknown. Constrained BO is a promising approach for achieving that goal.

Functional gradient optimisation methods take a different approach to optimisation. Using FGD, the process of optimisation is invariant of the choice of solution representation. Such abstraction is not available in BO, where the solution domain has to be predefined and is typically low dimensional ($<10$). With FGD, the optimisation can use a richer representation, which is useful for path planning and robotic manipulation. However, a number of open issues still remain, especially when planning in occupancy maps, which we will address in the following chapters.

The introduction of Hilbert maps has opened up new possibilities for planning algorithms. Hilbert maps provide a fast discriminative model that still retains spatial relationships. As the occupancy is given in closed-form, many other properties of the map, such as the spatial gradients of occupancy or the map's entropy, can be tractably computed. This trait makes Hilbert maps a favourable occupancy representation for various trajectory optimisation planning algorithms that will be presented throughout this compilation.

# Chapter 3

# Bayesian Autonomous Exploration

## 3.1 Introduction

Autonomous exploration is a challenging dynamic decision-making process, where the goal is to build a representation of an initially unknown environment. While exploring, the robot determines its own position and decides where to move next based on its objective. Ideally, these decisions correspond to continuous trajectories which determine the goal pose of the robot and a path leading to that point. Such a path would maximise the robot's objective, while ensuring safety. However, given the dimensionality and shape of the search space imposed by the motion constraints, a closed-form solution to the general exploration problem is intractable. The numerous exploration techniques available in the literature provide different approaches for making a decision on which path to follow, each with its own strategy for dealing with the uncertainty of the model and planning horizon. Risk and safety are typically addressed during execution of the chosen path, and not as an integral part of the exploration decision-making process.

In this chapter, we present a novel framework for autonomous exploration over continuous paths using constrained *Bayesian optimisation* (BO) (Gelbart et al., 2014), which we term *constrained Bayesian exploration* (CBE). *A-priori*, the exploration objective and constraints functions are unknown, i.e. have no closed-form expression. Knowledge about any of these functions is obtained solely from noisy observations, typically using forward simulation. A naive approach for optimising path decision in such a case is to employ an exhaustive search. Of course, such a process is computationally infeasible due to the size of the search space and cost associated with evaluating the reward and constraints functions over entire paths. BO uses a completely different approach, which turns autonomous exploration into an active learning process over the continuous search space. Instead of exhaustive sampling, BO learns probabilistic surrogate models for the expensive-to-evaluate reward and constraints functions (Brochu et al., 2010). A simple and cheap heuristic function, called acquisition function, guides an effi-

Part of this chapter has been accepted for publication in the *International Journal of Robotics Research*.

cient sampling schedule based on the posterior mean and variance of the surrogate models. The acquisition function balances the exploration-exploitation trade-off which guarantees convergence to the the optimum while ensuring probabilistic completeness of the objective and constraints.

The main contributions of this chapter are; First, we formulate the problem of building an occupancy map of an unknown environment as an optimisation problem over continuous paths, instead of the common approach of optimising the selection of a single goal point. Second, we develop CBE, an innovative method to solve this problem, i.e. optimise path decision whilst keeping the robot safe and within its dynamic constraints. CBE provides a principled and robust approach for optimising exploration using Bayesian optimisation. As such, CBE guarantees convergence to a local solution and follows well known BO's regret bounds (Srinivas et al., 2010).

The remainder of this chapter is organised as follows. Section 3.2 surveys the work related to autonomous exploration. In Section 3.3 we define the problem of map exploration. Section 3.4 gives an introduction of the basic building blocks of constrained Bayesian exploration and details the algorithms behind CBE. Experimental results and analysis for various scenarios are shown in Section 3.5. Finally, Section 3.6 draws conclusions on the proposed method.

## 3.2 Related work

Autonomous exploration can be seen as an active learning process aimed at minimising uncertainty and producing high-fidelity maps (Makarenko et al., 2002; Stachniss, 2009). Exploration requires solving simultaneously mapping, path planning and localisation. Due to its complexity, existing research has mainly focused on solving a relaxed form of this problem, by either decoupling processes or by limiting the solution search space.

The plethora of autonomous exploration methods is categorically divided into two branches; frontier-driven and information-theoretic. A quantitative comparison between the various exploration algorithms is presented in (Juliá et al., 2012).

The key concept in any frontier-based exploration is moving towards the edges of the known space, i.e. the boundary between free space and unmapped regions (Yamauchi, 1997). In its simplest form, after identifying and clustering frontiers, the robot moves towards the closest one. Other authors suggest various utility functions to prioritise candidate frontier locations. González-Baños and Latombe (2002) used expected information gain at the frontier and travelling cost. Basilico and Amigoni (2011) incorporated the overlap with known space as a measure for self-localisation. Extensions for 3D autonomous exploration have also been

suggested by various authors (Dornhege and Kleiner, 2013; Shen et al., 2012; Shade and Newman, 2011).

Information driven exploration strategies minimise a utility associated with the uncertainty of the map. Early work only addressed finding the *next best view* (NBV), which is the discrete location that will have the greatest effect on the utility function. Whaite and Ferrie (1997) proposed minimising the entropy of the map. *Mutual information* (MI) has also been suggested as a measure for the predicted reduction of map uncertainty (Elfes, 1996; Bourgault et al., 2002). Julian et al. (2014) suggested MI to encode geometric dependencies and drive exploration into unexplored regions in a similar fashion to frontier based approaches. Makarenko et al. (2002) proposed an integrated exploration method that combines the goal functions of map and localisation uncertainties with the cost of navigation, balancing exploration with *simultaneous localisation and mapping* (SLAM) loop-closures. Tovar et al. (2006) extended this technique by selecting several observation points using a tree search. Yet, the decision on the path passing through these points is still not part of the optimisation process. Stachniss et al. (2005) used a particle filter to calculate the expected information gain of an action. However, this formulation still defines point actions of either loop-closure or exploration. A method to generate a path based on information potential fields is proposed by Vallvé and Andrade-Cetto (2015). The path is generated by applying a grid-step gradient on the potential fields, resulting in a path that does not necessarily comply with the robot's kinematic restrictions.

Several non-myopic exploration methods have emerged in recent years. These methods treat exploration as a sequential decision process. Yang et al. (2013) used a *rapidly-exploring random tree* (RRT) planner with Gaussian process occupancy map to generate a safe path that minimises MI. Similarly to CBE, the RRT planner does not set a goal point but rather explores promising paths. The difference in algorithms lies in the optimisation process. While RRT uses predefined valid branches for its tree, CBE optimises path selection over the continuous domain. Furthermore, CBE learns the path constraints, which makes it a more flexible algorithm. Hollinger and Sukhatme (2014) developed sampling-based robotic information gathering algorithms, which build an exploration graph/tree from randomly drawn configurations and enforce motion and planning constraints via a node connection process ('steer'). While in the limit of infinite number of samples, these algorithms result in a continuous and optimal exploration path, in a realistic scenario of a fixed sampling budget, solutions are not guaranteed to be optimal even in a local sense. Charrow et al. (2015) combined both frontier and information-theoretic approaches. Global goal candidates are produced by identifying frontiers. A coarse path to each candidate is generated using local mo-

tion primitives that satisfy the kinematic envelope of the robot. After assessing the information gain of all candidates, the best path is refined using a *sequential quadratic programming* (SQP) solver. While this method optimises control inputs in continuous space, the search is limited to a single promising path. CBE, on the other hand, does not define a goal point, and its optimisation is done on the entire domain of controls. Kollar and Roy (2008) proposed an exploration procedure that maximises map coverage, by choosing a set of observation points that the robot trajectory must pass through. The executed path minimises the errors introduced by the robot motion. The control policy is implemented by a *support vector machine* (SVM) classifier trained off-line.

Another exploration approach was introduced by Marchant and Ramos (2014). In this case, Bayesian optimisation was used to learn and optimise a utility function along continuous paths. They employed two instantiations of BO, one for the probabilistic model of the utility and another to select a continuous informative path. However, their optimisation process does not consider any motion or safety constraints, which are learned and incorporated in the CBE framework. In a more recent work, Marchant et al. (2014) developed a sequential Bayesian optimisation method within a *partially observable Markov decision process* (POMDP) framework. They used *Monte Carlo tree search* (MCTS) to approximate the unconstrained solution for a spatio-temporal monitoring problem. Martinez-Cantin et al. (2007) and more recently Martinez-Cantin et al. (2009) utilised Bayesian optimisation to find control policies that minimise the state error of a robot and landmarks. While they used a different cost function, their method resembles the approach taken in this work. However, CBE extends this method by incorporating and learning unknown constraints during optimisation.

Lauri and Ritala (2015) used a POMDP with an MI objective to plan exploration paths with a fixed horizon using tree search methods to optimise the exploration policy. Their method relies on a *Monte-Carlo* (MC) approximation for the MI objective forward simulation. CBE, on the other hand, uses BO to efficiently manage the objective and constraints functions sampling. Another POMDP continuous-domain planning technique was presented by Indelman et al. (2015). They treated the exploration as an optimisation of the expected cost over several look-ahead steps. The cost at each step is inferred from the joint probability distribution of the robot and environment state. As the expected cost has a closed form expression, the authors use gradient descent to locally optimise the policy selection. A similar approach was taken by Rafieisakhaei et al. (2016) which defined a penalised cost function based on the *maximum-a-posteriori* (MAP) state estimate and control effort penalties.

Our method takes a different approach to optimisation. Instead of using a

closed-form expression for the objective, CBE learns it from samples, which results in a more flexible solution. Therefore, there is no need to define an expression for the objective and constraints. Rather the non-parametric structure captures our belief about the cost function. Coupling that with the BO framework, it provides better guarantees that a solution will converge to the global optimum. The safe exploration for active learning method (Schreiter et al., 2015) provides a similar method for constrained BO. However, the optimisation objective of (Schreiter et al., 2015) uses only the exploratory component of BO and does not exploit the model to optimise the objective. The *SafeOpt* optimisation method proposed by (Sui et al., 2015) uses regularity assumptions about the objective function to perform iterative optimisation in one-step reachability regions. Similar approach for handling safety constraints is taken by *safeMDP*, proposed by Turchetta et al. (2016). Both *SafeOpt* and *safeMDP* utilise a Gaussian process to model the objective function. This model is then used to generate the set of safe states. The next state is chosen according to its associated model's uncertainty, i.e. it only considers the exploratory component of the objective function. In contrast, CBE maintains global models of the constraints and of the objective function based on appropriate samples of these functions. These models are then used to drive an efficient optimisation process under a constrained Bayesian optimisation framework. In addition, *SafeOpt* and *safeMDP* perform optimisation for a finite set of states, whilst CBE optimises over continuous paths.

In summary, in this work we take a more holistic approach to exploration. Ideally, one would like to select the path that yields the highest reward. However, evaluating path reward is expensive and, as such, simple global optimisation strategies are rendered impractical. Therefore, most exploration techniques break this problem into two sub-problems; finding the next observation point(s) and selecting a path through these points. By contrast, our method uses a modified version of BO that finds a solution for these two sub-problems at the same time. The reward function and any associated motion constraints, such as turn rate limits or obstacles, are treated as functions which are learned by the optimiser. The output of the optimisation procedure is a path that maximises the reward without violating any of the constraints. Using BO, CBE attains a robust and efficient solution for the complex problem of optimising exploration objective over continuous paths as it relies on BO's local convergence guarantees and regret bounds (Srinivas et al., 2010).

## 3.3 Exploration as an Optimisation Problem

In this section we formally describe the problem of safe autonomous exploration for building occupancy maps. We rely on the description of the map representation in Section 2.4 for formulating the exploration process as an optimisation problem over the robot's action space.

An optimal exploration strategy generates a path that minimises a utility function which is representative of the uncertainty of the map. In its broadest form, this path is a series of finitely many control inputs $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ over a valid configuration space $\mathcal{C}$. Given a desired objective function $f$, exploration can be defined as the following optimisation problem:

$$\mathbf{u}^* = \underset{\mathbf{u}}{\operatorname{argmin}}\, f(\mathbf{m}, \mathbf{u}) \quad \text{s.t.} \quad \mathbf{u} \in \mathcal{C}. \tag{3.1}$$

One can define a constraint mapping function $\Lambda$ such that

$$\Lambda(\mathbf{u}) = \begin{cases} 1 & \mathbf{u} \in \mathcal{C} \\ 0 & \text{otherwise} \end{cases}. \tag{3.2}$$

Using this, we can write Eq. (3.1) as follows:

$$\mathbf{u}^* = \underset{\mathbf{u}}{\operatorname{argmin}}\, f(\mathbf{m}, \mathbf{u}) \quad \text{s.t.} \quad \Lambda(u) = 1. \tag{3.3}$$

In most cases, there is no closed-form expression for $f$ or $\Lambda$. Rather, both are *a-priori* unknowns and thus can only be estimated from sparse, expensive-to-evaluate and potentially noisy observations (samples). Accordingly, stochastic models are well suited to represent both $f$ and $\Lambda$. However, using probabilistic models for $f$ and $\Lambda$ requires changes in the formulation of the optimisation problem (Gelbart et al., 2014):

(i) optimisation is performed on the expected value of the objective function;

(ii) the constraints are replaced by probabilistic classifiers.

Consequently, Eq. (3.3) is transformed into:

$$\mathbf{u}^* = \underset{\mathbf{u}}{\operatorname{argmin}}\, \mathbb{E}[f(\mathbf{m}, \mathbf{u})] \text{ s.t. } \Pr(\Lambda(\mathbf{u})) < 1 - \delta. \tag{3.4}$$

Here, $\Pr(\Lambda(\mathbf{u}))$ stands for the probabilistic model of the constraint, and $\delta$ is its respective confidence bound. The constraint is estimated using $\delta$, indicating there is high probability that the constraint is met.

However, solving Eq. (3.4) in a continuous action space, $\mathbf{u}$, is computationally infeasible. A common approach, used by most information-theoretic methods

to reduce complexity is to search for NBV by optimising in pose space. The path planning process is divided into two separate sub-processes. The goal of the first sub-process is to define a set of discrete view points. Each point is the spatial local extermum of the objective function. The second sub-process plans a path from the current location of the robot to its next observation pose and is determined according to obstacles and robot's kinematic constraints (see for example Makarenko et al. (2002); Kollar and Roy (2008)). Although the discrete view points approach simplifies the optimisation process, the resulting path is suboptimal. The main drawback of this approach is that it only considers a limited set of points and thus disregards the potential gains (or the lack thereof) along the entire path. A less obvious, yet significant disadvantage of discrete optimisation stems from the fact that the expected cost of the resulting path are not an integral part of the view point selection. Penalty heuristics, e.g. distance to goal point, are typically incorporated in the objective function in order to encode cost, yet their limited form underestimates the real cost imposed by motion and safety constraints. As a simple example, consider the case in which the next view point is located close to the robot but is separated by obstacles. While a valid path to that point might exist, it is less desirable due to excessive cost.

In this work, we treat exploration as a black-box optimisation problem over continuous paths, where the objective function and constraints are learned from observations. To overcome the computational limitations of Eq. (3.4), we parameterise the paths using $\theta \in \mathbb{R}^D$. In our method, $\theta$ parametrises a quadratic spline over the workspace of the robot. We define a unique transform $\xi$ that maps a parameter to a trajectory. The objective function and constraints are observed along these trajectories, $\xi(\theta)$. Given $\xi(\theta)$, the optimisation can be written with regards to the parameter $\theta$:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}\Big[f\Big(\mathbf{m}, \xi(\theta)\Big)\Big] \quad \text{s.t.} \quad \Pr\Big(\Lambda\Big(\xi(\theta)\Big)\Big) < 1 - \delta. \tag{3.5}$$

The objective function $f$ can be defined in various ways, as discussed in Section 3.2. In this work, we use the accumulated MI over the entire path as the objective function. However Eq. (3.5) provides a general optimisation method for any choice of $f$ and $\Lambda$.

The following section describes our approach for solving Eq. (3.5) as a black-box optimisation problem. It provides details on how to obtain and incorporate noisy observations of both $f$ and $\Lambda$, which leads to an optimisation process that maximises the utility of the entire path while minimising the risk of violating any constraints.

## 3.4 Constrained Bayesian Exploration

Finding the best path in a continuous space requires optimising a reward function evaluated for any given trajectory. However, computing such a reward for the entire space of paths is computationally infeasible. Furthermore, obstacles and the robot's kinematic envelope impose constraints that might not have a closed form expression or are not known *a-priori*. BO provides a strategy to learn the reward function and constraints while searching for the valid extremum.

BO guides the optimisation process. Given a sparse training set, BO builds models of the reward function and constraints using Gaussian processes (GPs). With these models, BO identifies promising trajectories that correspond to the optimal path with high probability. Constraints are handled in a statistical manner, with BO balancing risk and rewards.

A schematic overview of CBE is shown in Fig. 3.1. As with any autonomous exploration technique, the expected output of our method is a path that updates our belief about the map. As shown, constrained BO exploration is an iterative process of optimisation and learning. Using the current map as an input, BO explores the solution space by sampling promising path candidates. The samples update the surrogate GP models for the reward and constraints, which is followed by the next BO suggestion. This process continues until resources are exhausted and the optimal path is then selected and executed.

When planning a path for autonomous exploration, the objective is to acquire new knowledge and improve the existing map in an efficient manner. Yet, both robot and environment impose restrictions on the solution space. Any selected path should be safe, i.e. free of obstacles, and within the kinematic envelope of the robot. Constrained BO is a very flexible tool to find such a solution. Safety and motion restrictions are treated as constraints which are learnt while the optimiser searches for the global extremum among the possible paths.

CBE is an iterative process of finding a promising sampling location, observing the values of the unknown objective and constraints functions at the location, and updating the model using the observed values. In the context of robotic exploration, sampling a location corresponds to testing a new path candidate. Each path candidate is defined by parameters $\theta$, as discussed in Section 3.3. The objective function and constraints are observed along trajectories, $\xi(\theta)$. The pseudo code for assessing a new parameter set, i.e. new path, is shown in Algorithm 3, which consists of two major parts that are detailed next:

- Path Candidate Validity Assessment,
- Reward calculation.

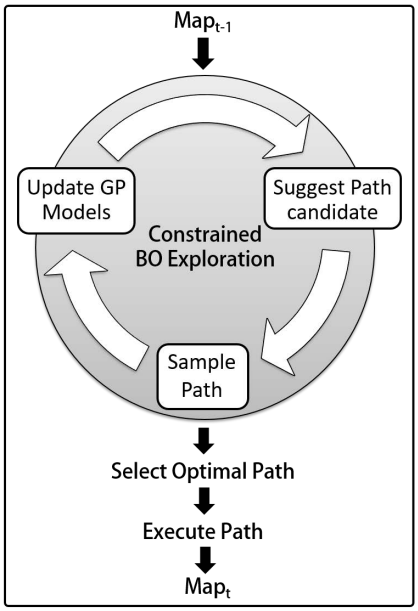



Figure 3.1: A schematic overview of the constrained BO exploration process. Using the current map, BO explores the solution space by guiding the sampling process.The utility of each sampled path is evaluated using forward simulation over the current map. These samples are then used to update the model belief path utility represented by GPs. This process continues until resources are exhausted, and the optimal path is select and executed.

### 3.4.1 Path Candidate Validity Assessment

The role of the path candidate validity assessment is to determine whether a path is valid or not. A valid path is safe from collisions with obstacles and within the kinematic capabilities of the robot.

There are two tests to assess the validity of a new path. First, the maximum curvature, $\kappa_{max}$, along the path is evaluated. If it exceeds a user-defined threshold, $\kappa_{max} > \delta_\kappa$, then the path is considered invalid and no other tests are needed. Although this is a relativity simple constraint, it has the potential to incorporate other motion considerations, such as energy or execution time budgets. Furthermore, learning the motion constraints provides greater flexibility when responding to changing driving conditions.

The second test validates the safety of a path. Given the occupancy map, it identifies obstacles along the path and dead-ends. Additionally, to ensure safety, the planned path should not traverse unobserved parts of the map. Formally, we require that the occupancy along a safe path should not exceed a user-defined

confidence threshold, $\delta_{safe}$:

$$Safe(\theta) = \begin{cases} 1 & \max\Big(p(\mathbf{m}[\xi(\theta)])\Big) \ < \delta_{safe} \\ 0 & \text{otherwise} \end{cases}, \tag{3.6}$$

where $\max\Big(p(\mathbf{m}[\xi(\theta)])\Big)$ is the maximum occupancy along the path $\xi(\theta)$.

It is important to note again that the result of Eq. (3.6) is used as a point observation in the generation of a stochastic model for the safety constraints. As this is a learned model, which is based solely on observations, the exact implementation of the safety criteria may vary. Instead of using $\mathbf{m}[\xi(\theta)] \ < \delta_{safe}$, the user can define a different test to assess path safety that better suits the robot and environment configuration. The simplest example changes the confidence safety threshold, $\delta_{safe}$, which will modify the risk-reward balance. More complicated methods may use different occupancy maps or include safety test for dynamic obstacles.

If the path is invalid, an additional post-processing step is taken in order to better define the valid solution space. By identifying the point where the path becomes invalid (e.g. hit an obstacle), the invalid path can be broken down into subsets of valid and invalid paths. The parameters $\theta$ of these derived paths can be easily computed from the parameters of the original path. Consequently, this post-process produces several safety observations from a single path sample. These observations are then used in the update of the GP models and help reduce the uncertainty of the relatively sparse GP models. But more importantly, it provides the GP model with the boundaries, in parameter space, between the valid and invalid space.

### 3.4.2 Reward calculation

Equation (3.5) defines the underlying optimisation process of CBE and has a general form, which is invariant to the choice of objective function $f$. CBE only requires access to $f$ through noisy observations. The implementation of CBE in this paper uses *mutual information* (MI), often referred to also as information gain, as the objective function. MI measures the reduction in the map entropy after observations are made:

$$MI\Big(\xi(\theta)\Big) = H\Big(\mathbf{m}\Big) - H\Big(\mathbf{m}|\xi(\theta)\Big). \tag{3.7}$$

Here $H(\mathbf{m})$ is the entropy of the map. The conditional entropy $H\Big(\mathbf{m}|\xi(\theta)\Big)$ is the entropy of the map after a path, defined by $\theta$, is executed. In order to approximate

$H\Big(\mathbf{m}|\xi(\theta)\Big)$ along an entire path, CBE builds a hypothetical map $\mathbf{m}'$ for each path sample. $\mathbf{m}'$ is built by simulating laser scans as the robot executes the path. The simulated scans are generated under the assumption of an optimistic agent, i.e. it adheres to the existing map $\mathbf{m}$ but assumes free space (maximum range reading) for unknown areas. Under this assumption, the entropy of the hypothetical is an upper bound for the empirical MI:

$$MI\Big(\xi(\theta)\Big) \leq H\Big(\mathbf{m}'\Big). \tag{3.8}$$

Another approach for computing MI was proposed by Charrow et al. (2015), where $\mathbf{m}'$ is computed by marginalising out cell occupancy over each simulated beam. This approach is useful for maps built using noisy 3D sensors such as RGB-D, but adds little value when dealing with laser scanners. With high-update-rate and low-noise sensors such as laser scanners, only the transition areas, from unoccupied to occupied e.g. next to walls, are uncertain. In other observed regions of the map, $p(\mathbf{m})$ tends to 0 or 1, making marginalisation redundant and similar MI results in both methods. In unobserved areas of the map, where $p(\mathbf{m}) = 0.5$, the marginalised MI of (Charrow et al., 2015) differs from the optimistic agent MI estimate only by a multiplicative factor. Consequently, the procedure proposed by Charrow et al. (2015) offers little value for the exploration scenario presented in the work especially given its additional computational overhead. We emphasise again that the exact method for MI calculation is irrelevant as CBE, a black-box optimiser, learns a representation of the MI response surface from forward-simulation observations, regardless of the mechanism used to generate them.

parametric optimisation of Eq. (3.5) was chosen as a trade-off of computational complexity with the expressivity of the solution. In certain scenarios, the limited path expressiveness may result in safe but non-informative paths. To overcome this, the following heuristics are used:

- The first term provides a global context to the overall objective function. A coarse path is planned from the robot's location to the nearest frontier. This path does not have to be traversable by the robot and it can violate safety or kinematic constraints. However, it biases path selection towards a region of unexplored space. We define a penalty term, $P_H\Big(\xi(\theta)\Big) = cos(\omega)$, where $\omega$ is the difference between the direction of development of the candidate path and the coarse path. Therefore, a path that develops in the opposite direction of the global coarse path will have higher penalty than a path that is oriented towards a similar direction. As $P_H$ is a cosine, the amplitude of this penalty is $|P_H| \leq 1$.

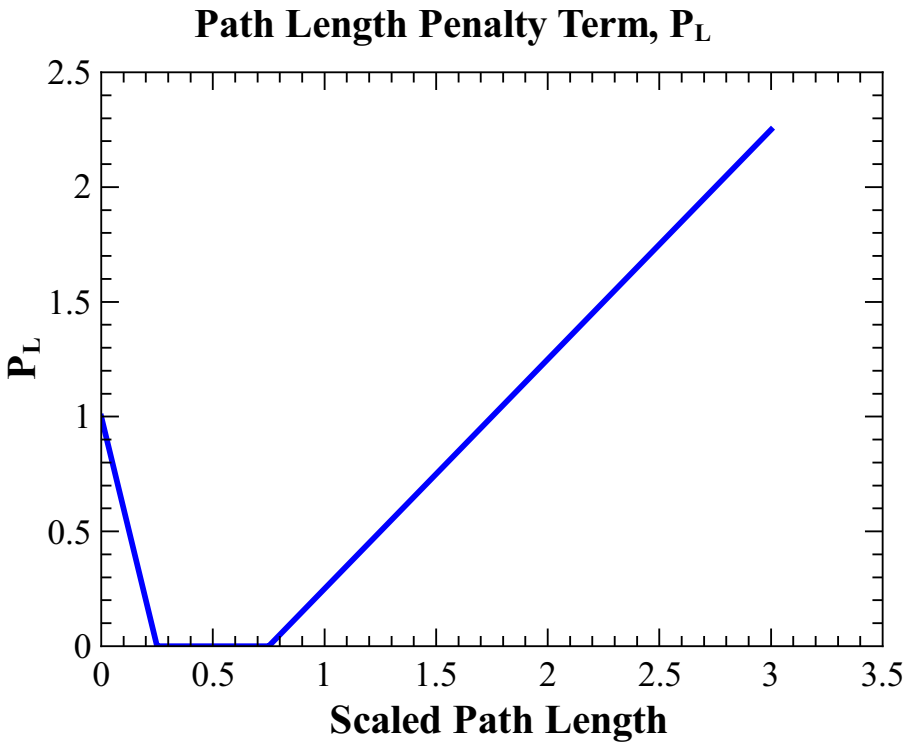



Figure 3.2: $P_L$, path length penalty term, as a function of path length. The length is scaled with respect to the sensor maximum range. This penalty term penalises very short and long paths. Short paths are undesirable as they have little effect on map building. Longer paths are penalised as a function of their length, in order to prevent overly confident solutions.

- The second term $P_L$ is a function of the path length. Figure 3.2 depicts the choice of $P_L$ used in this paper, where the path length is scaled with respect to the maximum sensor range. The rationale behind $P_L$ is to penalise very short and long paths. Very short paths are undesirable as their ability to reduce uncertainty is negligible. Longer paths are penalised in order to prevent overly confident decisions.

The additional penalties are added to the MI reward with corresponding weights, $W_1$ and $W_2$. These weights keep the penalties small compared to the typical MI utility:

$$MI_{Modified}\Big(\xi(\theta)\Big) = MI\Big(\xi(\theta)\Big) + W_1 \cdot P_H\Big(\xi(\theta)\Big) + W_2 \cdot P_L\Big(\xi(\theta)\Big). \tag{3.9}$$

The weights $W_1$ and $W_2$ are user defined and capture the user's approach to exploration. For example, increasing $W_1$ will result in a process which resembles frontier exploration. In our implementation, the goal of $P_H$ is to pull the robot from areas in the map that produce non-informative paths, for which the MI re-

ward is negligible. Consequently, we set $W_1 = 100$. This value is approximately 10% of the average MI reward in standard scenarios. This value will have little effect in standard planning scenarios. While in situations where MI is close to zero, $W_1 = 100$ guarantees that $P_H$ will exceed the expected noise of MI observations. The weight of the second penalty $W_2$ was set to 50, which affects the overall reward only for very short paths or if the expected path exceeds 5 times the maximum laser range.

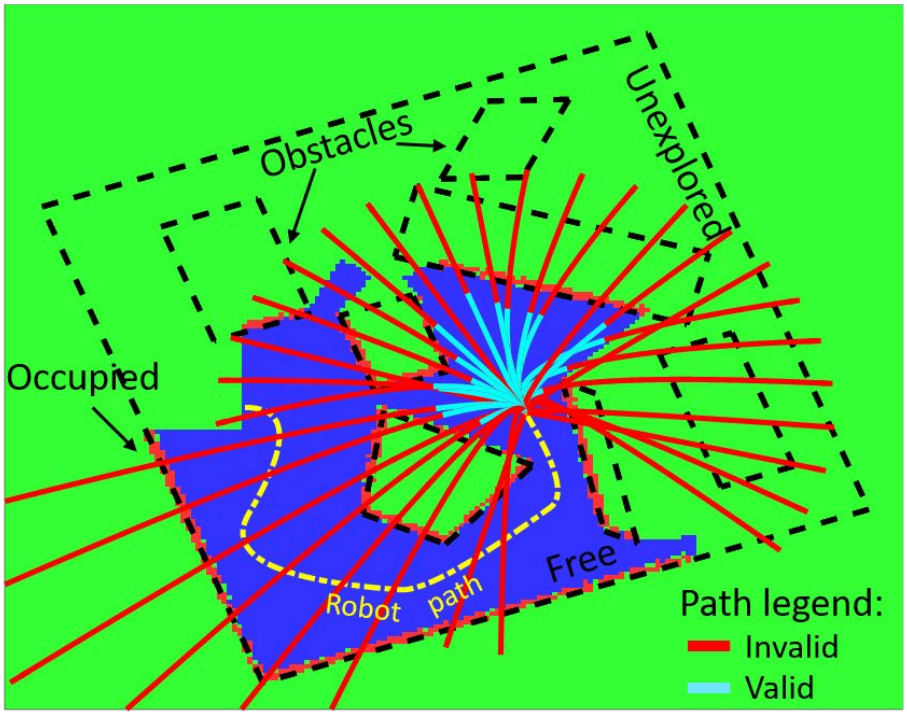
Even with the simplest occupancy map representation, the forward projection model needed to estimate MI is expensive to evaluate. This is the main motivation for using BO; optimising decision making while keeping sampling low. Instead of optimising by explicitly calculating the forward simulation MI results, BO learns a model for MI from sparse samples. It then uses these models to infer the next sampling location. The efficiency of BO relies on the accuracy of the learned GP models. However, a high fidelity GP surrogate model requires a substantial number of function observations. We take advantage of the way MI is sampled in order to increase the number of observations without increasing the computational cost. We notice that MI along the path is a non-decreasing monotonic function. Since the robot motion along the path is a set of sequential observation points, MI in any given point is the sum of accumulated effect of all previous observations and the contribution of the current observation:

$$\begin{aligned} MI(\theta_{k+1}) &\equiv MI([u_1 ... u_{k+1}]) \\ &= MI([u_1 ... u_k]) + \delta MI(u_{k+1}|z_{k+1}) \\ &\equiv MI(\theta_{k+1}) + \delta MI(u_{k+1}|z_{k+1}). \end{aligned} \tag{3.10}$$

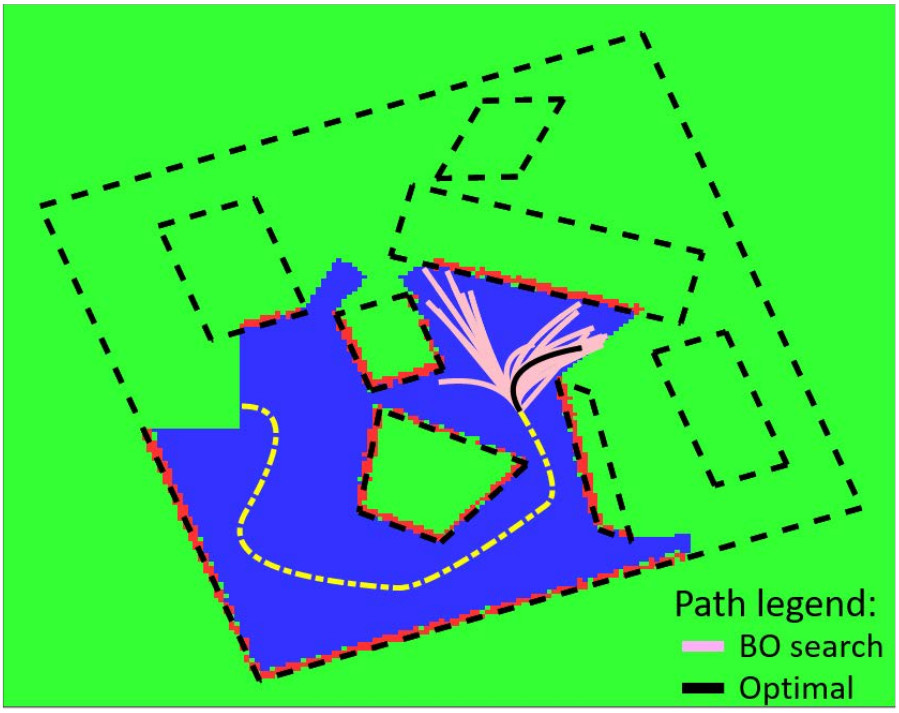
Thus, by evaluating MI sequentially, CBE produces several reward observations from a single path sample with no additional computational cost. More samples produce a denser and more accurate GP model of the objective function.

### 3.4.3 CBE Algorithm

A pseudo code for CBE using constrained Bayesian optimisation is given in Algorithm 4. Figure 3.3 provides a visual explanation of this process. The algorithm is divided into two parts. To improve efficiency, we initialise the GP surrogate models by sampling the reward and constraints functions using a training set $\Theta$. The training set is a list of path parameters $\theta \in \Theta$ which defines a small set of paths, typically 20 to 50, to evaluate. The paths are distributed evenly in all directions in order to obtain a balanced training set. The paths associated with the parameters in the training set are shown in Fig. 3.3a by the red (invalid) and cyan (valid) paths. As explained before, the algorithm tries to extract valid path




(a) Generating training set




(b) Optimisation

Figure 3.3: CBE searches for an optimal path in an unexplored room. Walls and obstacles are denoted by the dashed black lines, which may be unknown to the robot. Green areas are unexplored, blue are known to be free while red areas contain known obstacles. (a) The paths used in the training set are shown in red (invalid) and cyan (valid). (b) CBE produces new path candidates for assessment (pink) with the final output of the optimiser shown in black.

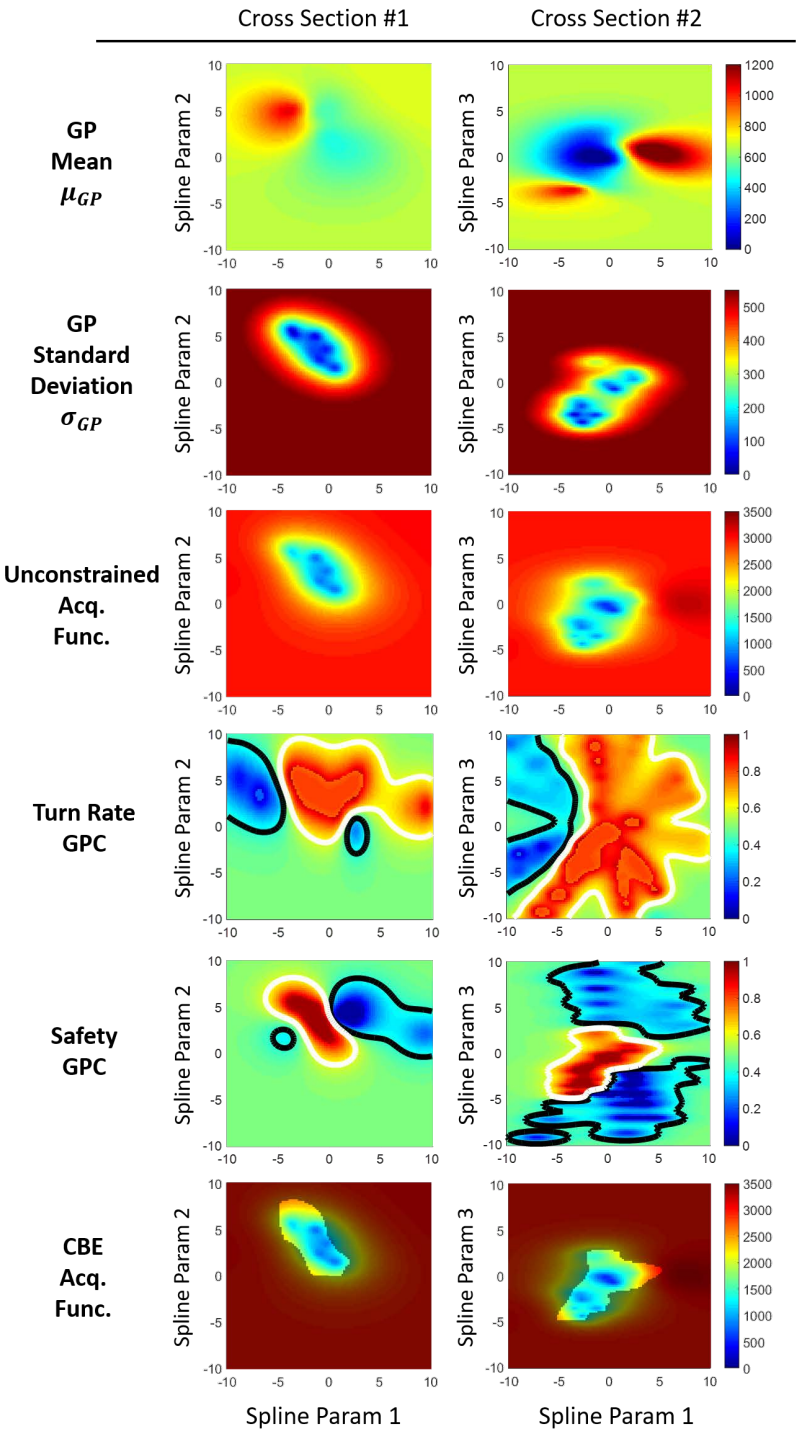



Figure 3.4: Images (cross sectional) of the various components of CBE for the scenario shown in Fig. 3.3. Rows depict different component of BO; GP regression mean, GP standard deviation, unconstrained BO acquisition function, turn rate GPC, safety GPC and the CBE acquisition function. Columns show two orthogonal cross sections. Contours in GPC images represent valid (white) and invalid (black) thresholds. Black overlay defines the CBE acquisition function (last row) valid (transparent) and invalid (semitransparent) regions for optimisation given observations.. CBE will try to maximise the acquisition function in valid regions, which produces a suggestion for the next observation point.

**Algorithm 3:** CBE Path assessment

**Input:** $\xi(\theta)$: assessed path
$f_{min}$: current objective minimum
**Output:** $\mathbf{P}_{valid}$, $\mathbf{P}_{reward}$, $f_{min}$

$\mathbf{P}_{valid} \leftarrow$ Check: Motion Constraints
$\mathbf{P}_{valid} \leftarrow$ Check: Safety
**if** $\mathbf{P}_{valid}$ **then**
    $\mathbf{P}_{reward} \leftarrow$ Evaluate reward: eq. 3.9;
    If $\mathbf{P}_{reward} < f_{min}$: $f_{min}$=$\mathbf{P}_{reward}$
**else**
    Path assessment(valid subset of $\mathbf{P}$)
**end**

**Algorithm 4:** CBE

/* Generate initial training set: */
$N$ = Size of training set
$\Theta \leftarrow$ Generate training path set(N)
**for** $\theta_k \in \Theta$ **do**
    $\theta_k \leftarrow$ Path assessment($\theta_k$) (Algorithm 3)
    Update GP and GPCs: $\theta_k$
**end**
/* Constrained BO: */
**for** $i = 1, 2, 3, ...$ **do**
    feasible region $\mathcal{C}$: $\mathcal{G}(\theta) = \prod_{k=1}^{K} \Pr(g_k(\theta) < 1 - \delta_k)$
    Find: $\theta_i \leftarrow \underset{\theta \in \mathcal{G}}{\operatorname{argmin}}\ \mathcal{G}(\theta) \cdot LCB(\theta)$
    $\theta_i \leftarrow$Path assessment($\theta_i$) (Algorithm 3)
    Update GP and GPCs: $\theta_i$
**end**
Execute optimal path

segments from an invalid path as can be seen in Fig. 3.3a. These paths, valid as well as invalid, are used in the update of the GP and GPCs models, which serve as a prior model for the subsequent constrained BO stage.

The second part of Algorithm 4 is the constrained BO. In Fig. 3.3b, the outputs of this stage, which corresponds to the paths suggested by the optimiser, are depicted in pink. With every attempt, BO updates the GP and GPCs and becomes more confident in the model of the objective function, the constraints, and the location of the global minimum. This learning process is evident from the distribution of the suggested paths. Although most are bundled around two main directions, there are some stray paths that check potentially rewarding alternatives. Also, some paths are on the borders of the unexplored regions, suggesting the optimiser tries to learn about the motion constraints. The final output of

the optimiser, the optimal path where the accumulated reward is maximised, is shown in black.

### CBE Example

To gain additional insight into the optimisation process, Fig. 3.4 presents images of key CBE elements. As CBE is a high-dimensional optimisation process, cross-sections are used for the visualisation. The first key element of CBE is the surrogate GP model of the objective function, shown in the first two cross sections, $\mu_{GP}$ and $\sigma_{GP}$. The GP represents our belief about the learned objective function. The main benefit of using a GP, as with other Bayesian regression techniques, is the ability to obtain an inference confidence measure. The non-parametric structure provides great flexibility in expressing the model's expected value and variance around observations. Previous methods using BO for exploration, e.g. in (Martinez-Cantin et al., 2009), optimise using the unconstrained acquisition function which is shown in Fig. 3.4. Instead of only optimising over the expected value, the use of an acquisition function incorporates the model uncertainty. However, unconstrained BO is not suitable for autonomous exploration as reward samples can only be acquired along valid trajectories. Beyond the valid region, the GP model provides only its intrinsic parameters; the model mean and maximum variance. Consequently, this breaks the internal BO feedback loop of sampling and updating, as the GP model is kept unchanged after any invalid sample.

The constrained BO framework is more suitable for autonomous exploration. The learned constraints, Turn Rate GPC and Safety GPC in Fig. 3.4, provide the optimiser with an additional layer that incorporates invalid samples without the need to define a closed-form expression for the constraints. Using GPCs provides an efficient method to query the certainty in which the constraints are met. Furthermore, the user can easily modify the validity threshold (shown as black and white lines), to adjust the optimiser's risk-reward balance. The combined acquisition function, which is shown in the last cross-section of Fig. 3.4, is the unconstrained BO acquisition function overlaid with the GPCs valid zone. Unlike unconstrained BO, integrating new invalid samples will not break the BO feedback loop. It will instead modify the GPCs. This behaviour allows the CBE to explore promising paths that are on the borders between valid and invalid.

### Computational Cost

Finally, we conclude this section with an estimate of the computational complexity of CBE. As GPs are used extensively throughout this algorithm, it is

not surprising that the CBE computational costs are mainly associated with GP inference complexity. Similarly to other non-parametric methods, the computational complexity depends on the size of training set $n$. In the CBE framework, however, two separate training sets are defined;

- $N$ includes the entire training set of valid and invalid points.
- $M$ ($M < N$) includes only the valid points used by the reward GP.

The computational complexity of a typical GP is $\mathcal{O}(n^3)$ and is due to the Cholesky decomposition of the covariance matrix (Rasmussen and Williams, 2006). GP prediction carries a lower complexity of $\mathcal{O}(n^2)$ arising from the solution of a triangular linear system. As we use a small number of $c$ GPCs, the overall complexity of the Cholesky decomposition of the various components of CBE is $\mathcal{O}(M^3 + cN^3)$. In addition, during optimisation, GP and GPCs model may be queried repeatedly leading to $\mathcal{O}(mM^2 + clN^2)$, where $l$ and $m$ are the number of queries of GPCs and GP, respectively, and $m < l$. Given $M < N$ the overall complexity of CBE, Cholesky decomposition and model queries can be estimated as $\mathcal{O}(cN^3 + clN^2)$. We can further simplify this expression by noting that the typical training set contains several hundred points, as is the number of optimisation steps; $l \sim N$. Thus, we can concisely write the overall CBE complexity as $\mathcal{O}(cN^3)$.

### 3.4.4 Incorporating Uncertainty in CBE

In the context of autonomous exploration, path planning is a decision making process aimed at improving the map fidelity. Any uncertainty, whether it is in sensor observations or in the robot pose, propagates into our belief over the map and corrupts it. While sensor uncertainty is typically fixed and arises from the system configuration, the robot location uncertainty is controlled by the robot's decisions. Reducing pose uncertainty is commonly addressed in the literature by incorporating a "loop closing" heuristic in the optimisation of the next observation point (for example (Makarenko et al., 2002; Indelman et al., 2015; Rafieisakhaei et al., 2016)). With the standard BO framework, incorporating such a heuristic requires modifications to the forward simulation reward calculation as described in the work of Martinez-Cantin et al. (2009). Handling uncertainty in the robot pose necessitates some adaptations to the CBE algorithm to ensure the safety of resulting paths. Therefore, we will leave the "loop closing" reward modification for future work, and discuss the required changes to CBE in the following section.

At the end of optimisation, CBE returns an optimal path. When considering only the nominal pose, the optimal path is safe and valid. However, the actual outcome of that path, and more importantly, its safety, depend on the real pose

of the robot. Figure 3.5 depicts an example of such a case, by plotting the same path for several starting poses, drawn from the robot's state distribution. It is clear that by not incorporating the pose uncertainty, the risk of collision is greatly under-estimated.

A common approach for handling partial observability in robotics is to locally re-plan during path execution. This simplifies path planning, as the safety of the path is maintained mainly during path execution. However, this approach reduces the safety margin for execution, limiting the action space for re-planning. For example, while planning a path along a corridor, a standard planner might yield a path that runs close to the walls. While this is a safe path, following it limits re-planning to a single side of the corridor. CBE, on the other hand, incorporates safety in its optimisation objective which widen the safety margins of a planned path. In the corridor example, this equates to a path that runs in the middle of the corridor, furthest from obstacles. Furthermore, by incorporating safety, CBE reduces the risk of diverging from the original plan due to re-planning, resulting in higher utility paths.

To better estimate the safety risk, we need to project the variance of the robot's location and orientation into the safety GPC model. However, the resulting probability density function might have a non-trivial form. An efficient solution to alleviate this problem utilises an *unscented transform* (UT) (Julier and Uhlmann, 1997). UT employs a deterministic sampling schedule to estimate the mean and variance of the desired distribution. The sample set, termed "sigma points", consists of $2n+1$ samples and weights for a $n$-dimensional space. Given the pose mean $\mu_{\mathbf{p}}$ and covariance $\mathbf{\Sigma_{pp}}$ the "sigma points" $\chi$ are defined by the following equations:

$$\chi_i = \begin{cases} \mu_{\mathbf{p}} & i = 0 \\ \mu_{\mathbf{p}} + (\sqrt{(n+\lambda)\Sigma_p})_i & i = 1, ..., n \\ \mu_{\mathbf{p}} - (\sqrt{(n+\lambda)\Sigma_p})_{i-n} & i = n+1, ..., 2n \end{cases} \tag{3.11}$$

Here $\lambda = \vartheta^2(n+\nu) - n$. $\vartheta$ determines the spread of the sigma points around the mean, $\mu_p$, and is typically a small positive number (in our experiment $\vartheta = 10^{-3}$). $\nu$ is a second order term to adjust kurtosis and is usually set to zero (Wan and Van Der Merwe, 2000).

For the sigma weights, we follow Wan and Van Der Merwe (2000), and define

separate values for mean and covariance calculations:

$$
\begin{aligned}
w_0^{mean} &= \frac{\lambda}{n+\lambda} \\
w_0^{cov} &= \frac{\lambda}{n+\lambda} + (1 - \psi^2 + \varrho) \\
w_i^{mean} &= w_i^{cov} = \frac{1}{2(n+\lambda)} \qquad i = 1, ..., 2\delta.
\end{aligned}
\tag{3.12}
$$

The parameter $\varrho$ is used to encode prior knowledge of the initial distribution. Since we assume a Gaussian distribution for the pose, we take $\varrho = 2$.

These sigma points serve as starting poses for alternative path outcomes as shown in Fig. 3.5. We employ the same mapping from actions to trajectory space, $\xi(\theta)$, but replace the implicit noiseless pose with the sigma points $\xi(\theta, \chi_i)$. As a result, we probe how $\xi$ changes the shape of the initial pose uncertainty and thus recover a stochastic estimate for the robot pose along the path as, $\mathbf{x} \sim \mathcal{N}(\rho, \Sigma_x)$:

$$
\begin{aligned}
\rho &= \sum_{i=0}^{2n} w_i^{mean} \xi(\theta, \chi_i) \\
\Sigma_x &= \sum_{i=0}^{2n} w_i^{cov} (\xi(\theta, \chi_i) - \rho)(\xi(\theta, \chi_i) - \rho)^T.
\end{aligned}
\tag{3.13}
$$

Given $\xi(\theta, \chi_i)$ and map, $\mathbf{m}$, one can now stochastically reason about the safety of a path, $\xi(\theta)$. Although straight forward, Algorithm 5 simplifies the safety estimation even further. Instead of inferring the pose distribution along the path, each sigma path, originating from a specific sigma point, is validated separately. The overall validity of a path, $\xi(\theta)$, is then determined by the worst-case scenario over all paths.

**Algorithm 5:** CBE Path assessment assuming partially observable pose

**Input:** $\xi(\theta)$: assessed path
$f_{min}$: current objective minimum
$\chi$: sigma points
**Output:** $\mathbf{P}_{valid}$, $\mathbf{P}_{reward}$, $f_{min}$

**foreach** $p \in \chi$ **do**
  $\mathbf{P}_{valid} \leftarrow$ Check: Motion Constraints($p$)
  $\mathbf{P}_{valid} \leftarrow$ Check: Safety($p$)
**end**
**if** $\mathbf{P}_{valid}$ **then**
  $\mathbf{P}_{reward} \leftarrow$ Evaluate reward: Eq. 3.9;
  If $\mathbf{P}_{reward} < f_{min}$: $f_{min}$=$\mathbf{P}_{reward}$
**else**
  Path assessment(valid subset of $\mathbf{P}$)
**end**

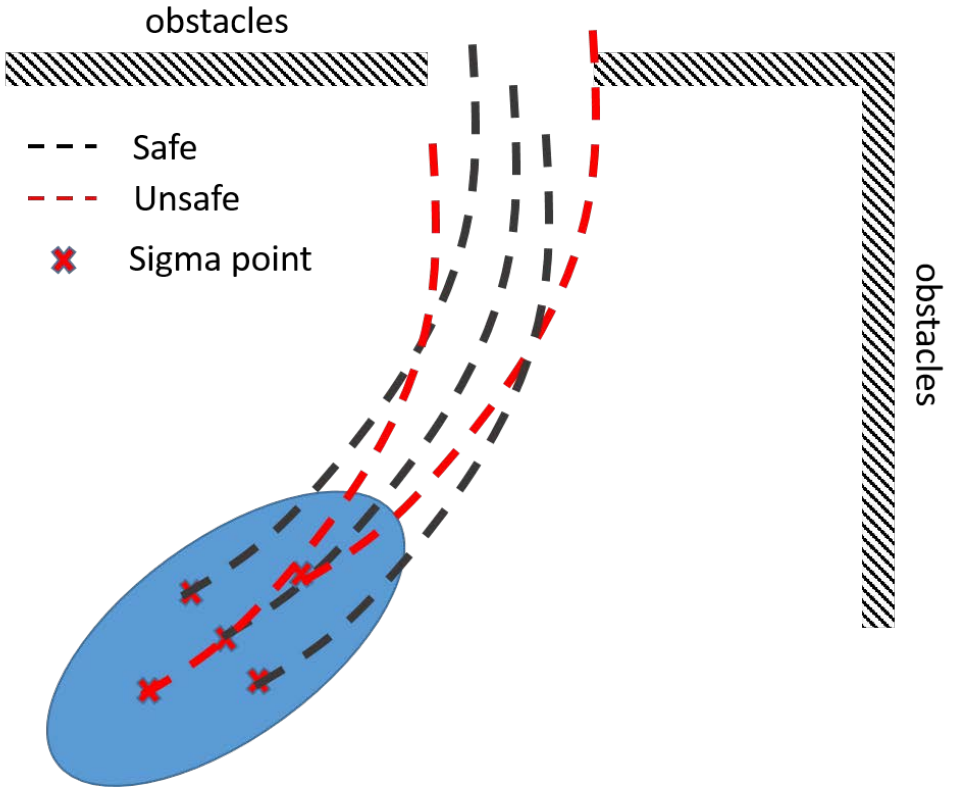



Figure 3.5: Uncertainty in path execution due to uncertain location and orientation results in a non-trivial distribution of the overall path safety.

## 3.5 Experiments

In this section, we evaluate the performance of CBE in simulation and with a real robot.

### 3.5.1 Simulations

We divide our simulation experiments into two categories. First, the robot maps various randomly generated cluttered unstructured environments. The second experiment involves exploration of a large scale complex networks of city roads. In both cases, we simulate a ground robot equipped with a 180° field of view (FOV) laser scanner driving at a constant speed. The turn rate of the robot is limited, forcing the optimiser to plan within the robot's kinematic envelope using quadratic splines. In all simulations, we assume full knowledge of the robot's pose.

#### Unstructured Environments

Many exploration experiments involve structured man-made scenarios. A structured environment is constructed of a network of corridors, for example, underground mines and buildings. Although the unexplored regions include obstacles, the corridor-like structure pulls the robot towards an obvious general path. Unstructured scenes, with randomly positioned obstacles, break any large-scale formation, and thus lack this implicit guidance. Furthermore, these scenarios exhibit additional difficulties, such as isolated areas with only a single access point, non-traversable narrow gaps, and long barriers dividing the world into several almost independent parts. Such an arrangement complicates the exploration process, since it introduces many more options the robot has to choose from. Fig. 3.6 shows examples of randomly generated worlds used to compare constrained BO exploration to other techniques.

A qualitative comparison between CBE and other map building techniques is shown in Fig. 3.7. The methods used for comparison follow the common exploration paradigm where the path is determined in two separate stages; (i) goal point selection, (ii) path planning:

A qualitative comparison between CBE and other exploration techniques is shown in Fig. 3.7. As a benchmark, we used a state-of-the-art SQP solver (Charrow et al., 2015) and also more traditional exploration methods that follow the common exploration paradigm, where the path is determined in two separate stages;

(i) Selecting the next observation point.

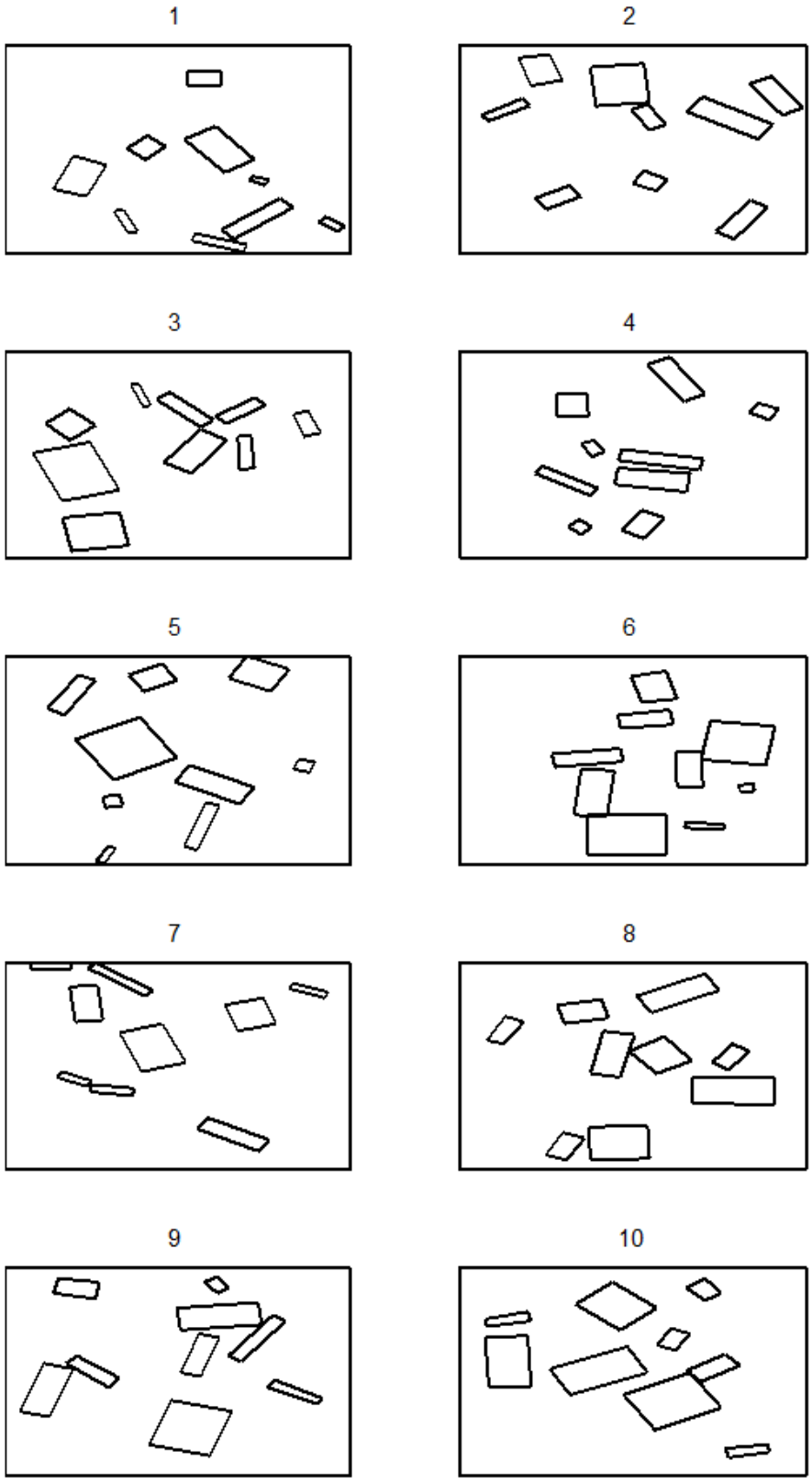



Figure 3.6: A sample of randomly generated unstructured worlds used for comparison of exploration methods shown in Table 3.1.

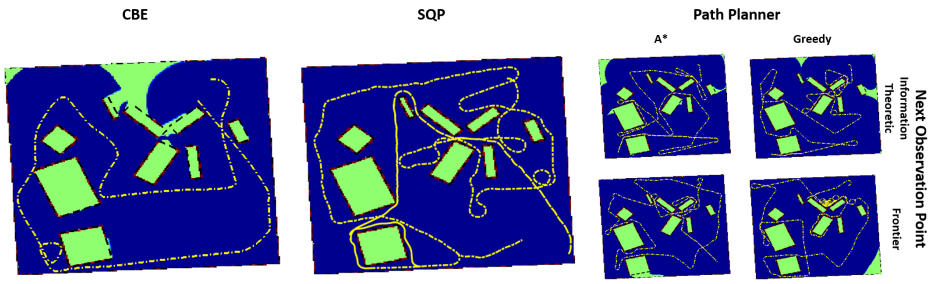



Figure 3.7: Comparison of simulation results for a randomly generated world (world 3).(left) CBE results, (middle) state-of-the-art SQP solver (Charrow et al., 2015), (right) simulation results with two planners for next observation point; Information-theoretic (Bourgault et al., 2002) and frontier (Yamauchi, 1997), and two smooth safe path planners, greedy and A*. The walls and obstacles are marked with black lines. In the grid map, green is unexplored regions, blue is free space and red occupied. The executed paths are shown as dashed yellow lines. CBE maximises the accumulated information gain at every decision point by avoiding previously traversed paths. In contrast, SQP, information-theoretic and frontier based planners exhibit a clear criss-cross pattern in the executed paths, as these methods only reason on the gain of a finite set of goal points.

We employ two methods for the selection of the next observation point;

a) a frontier-based method (Yamauchi, 1997) and

b) an information-theoretic approach based on the MI utility proposed by Bourgault et al. (2002). This utility has similar form as Eq. (3.7). However Eq. (3.7) describes the cumulative MI reward over an entire path, while the MI reward in (Bourgault et al., 2002) is calculated for a single goal point.

(ii) Planning a safe path to that goal point.

To emphasise the importance of the path, and not only of the end goal point, we employ two separate path planning techniques for stage two;

a) $A^*$ planner which finds the shortest traversable path to the goal point and

b) a fast greedy planner using the distance to the goal point as its heuristic.

Both path planners enforce the robot's safety and manoeuvrability limitations by generating a path from a valid set of motion primitives.

The main advantage of CBE visible in Fig. 3.7 is that the number of overlapping paths is smaller when compared to other planners. The BO planner takes a

relatively short path that minimises the time the robot moves through already visited parts of the environment. CBE maximises the accumulated information gain at every decision point by avoiding previously traversed paths, which is achieved by choosing paths without explicitly defining an end goal point. In contrast to CBE, both information-theoretic and frontier based planners exhibit a clear criss-cross pattern in the executed paths, regardless of the path planner used. This suboptimal performance arises from the two stage exploration process. Choosing a goal point first and then planning a path, prevents reasoning on the potential reward along the driven path. As a result, the knowledge gained while travelling to the goal point is not considered in the decision making. Similar results hold for the SQP solver, as it also relies on a finite set of viewing points, and does not consider the utility over the entire path.

With the traditional methods, the type of path planner used has also great impact on the overall exploration performance. As expected, the greedy path planner is less effective at finding a path through the clutter, evident by the tangled paths around obstacles. This leads to longer paths with an overall lower rate of improvement. Fig. 3.8 provides a quantitative comparison of the rate of reduction in the map's entropy between the various methods. The initial rate, in the first 10 seconds, is similar in all methods as the robot passes through the unexplored map. However, the rate at which the entropy decreases in the all benchmark methods becomes slower, coinciding with the robot travelling through already explored regions on its way to the goal point. This outcome is independent of the path planner used by the traditional approaches, $A^*$ or greedy. The difference in performance stems from the objective of these methods to find a finite set of viewing points, (a single point for the information-theoretic and frontier planners). In a complex unstructured scenario, there are many potential observation points at every decision. It is clear that by visiting these points, the entire environment will be mapped. However, with fixed-point planners there are no guarantees on the optimality of that process, as there is only limited reasoning about the executed path that joins these points. While these techniques might put a cost or penalty on the driving distance, the gains along a path are not taken into consideration. Even with the SQP-solver, where the set of viewing points is optimised, the utility of the path is not fully considered, leading to similar performance as the traditional planners. CBE, on the other hand, plans in its local neighbourhood taking into account the benefits and risks of potential paths, rather than selecting goal points. The global component pulling the robot toward the nearest frontier only affects the decision when the local information component is negligible.

Table 3.1 provides a quantitative comparison between the exploration tech-

Table 3.1: Comparison of exploration time between CBE, SQP solver (Charrow et al., 2015) and two planners for next observation point; Information-theoretic (Bourgault et al., 2002) and frontier (Yamauchi, 1997), and two smooth safe path planners, greedy and A*. The fastest method is marked in bold font.

| | Exploration Time [s] | | | | | |
|---|---|---|---|---|---|---|
| goal selection | CBE | SQP | Information-theoretic | | Frontier | |
| Path Planner | | | Greedy | $A^*$ | Greedy | $A^*$ |
| | | | | | | |
| World 1 | **63.8** | 132.0 | 136.1 | 68.9 | 139.9 | 85.1 |
| World 2 | 86.2 | 107.0 | 134.5 | **82.7** | 87.2 | 88.5 |
| World 3 | **86.5** | 101.2 | 108.1 | 98.4 | 111.6 | 99.9 |
| World 4 | 67.4 | 109.8 | 106.0 | **64.5** | 115.6 | 98.1 |
| World 5 | 79.6 | 154.3 | 84.4 | 89.8 | **73.5** | 78.3 |
| World 6 | 79.4 | 139.5 | 454.6 | **76.9** | 120.2 | 80.2 |
| World 7 | **62.6** | 133.0 | 87.9 | 65.3 | 99.0 | 86.6 |
| World 8 | **77.1** | 204.5 | 112.4 | 88.7 | 147.4 | 90.7 |
| World 9 | **71.7** | 211.6 | 129.0 | 115.0 | 132.4 | 102.3 |
| World 10 | **76.6** | 126.6 | 115.4 | 98.0 | 111.0 | 95.1 |

niques on several randomly-generated worlds shown in Fig. 3.6. As expected, $A^*$ is a noticeably better path planner than the greedy planner, as it guarantees the shortest traversable path to the goal point. However, it is not immediately clear, which observation point selection method performs best. Although, in some scenarios the performance of both information-theoretic and frontier methods is similar, there are cases where one method outperforms the other by a significant margin. The SQP algorithm exhibits similar properties, where it performs well in some scenarios and poorly in other. The CBE method, on the other hand, consistently maintains good performance. In the majority of the tested scenarios, CBE is the fastest method. In all other cases, it has similar performance as the leading method, whether frontier or information-theoretic. These results show that the CBE planner is less sensitive to the layout of the environment and provides a more consistent and robust method for exploration compared to the other techniques.

As we assume the robot pose is fully known in these simulations, repeatability was tested by changing the initial pose. Figure 3.9 and Table 3.2 present a

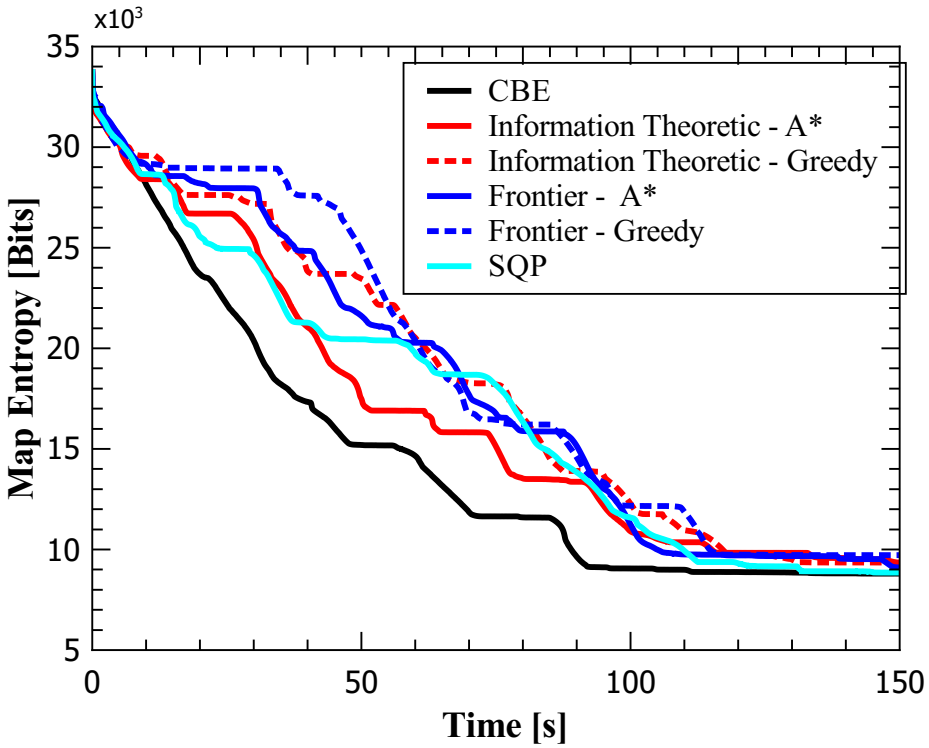



Figure 3.8: Quantitative comparison of the reduction in map entropy (world 3) between exploration methods presented in Fig. 3.7. The initial rate of entropy reduction is similar in all methods. However, the rate slows in both the information-theoretic and frontier methods, as the robot travels through already explored region in route to its next goal point. CBE, avoids previously traversed areas of the map, leading to a faster reduction in entropy.

comparison between CBE and frontier/$A^*$ in world 3 (see in Figs. 3.6 and 3.7) for various starting poses. Once again, we can see that the paths CBE chooses are typically more efficient in covering the environment, which leads to significantly shorter exploration times. The paths produced by frontier/$A^*$ tend to cross areas already traversed before. We note that CBE performs better than frontier even though it uses a relatively simple path representation (quadratic splines), which emphasises the importance of the optimisation step.

### Structured Environments

These experiments test CBE performance in a structured environment scenario. Part of the roads and paths network of Venice and Jerusalem old city were extracted from Google Maps. These complex networks of corridor-like patterns serve as the ground truth in this large-scale exploration experiment. In such a structured system, there is no clear advantage for the constrained BO method. The corridor structure forms an obvious path, which limits the local significance

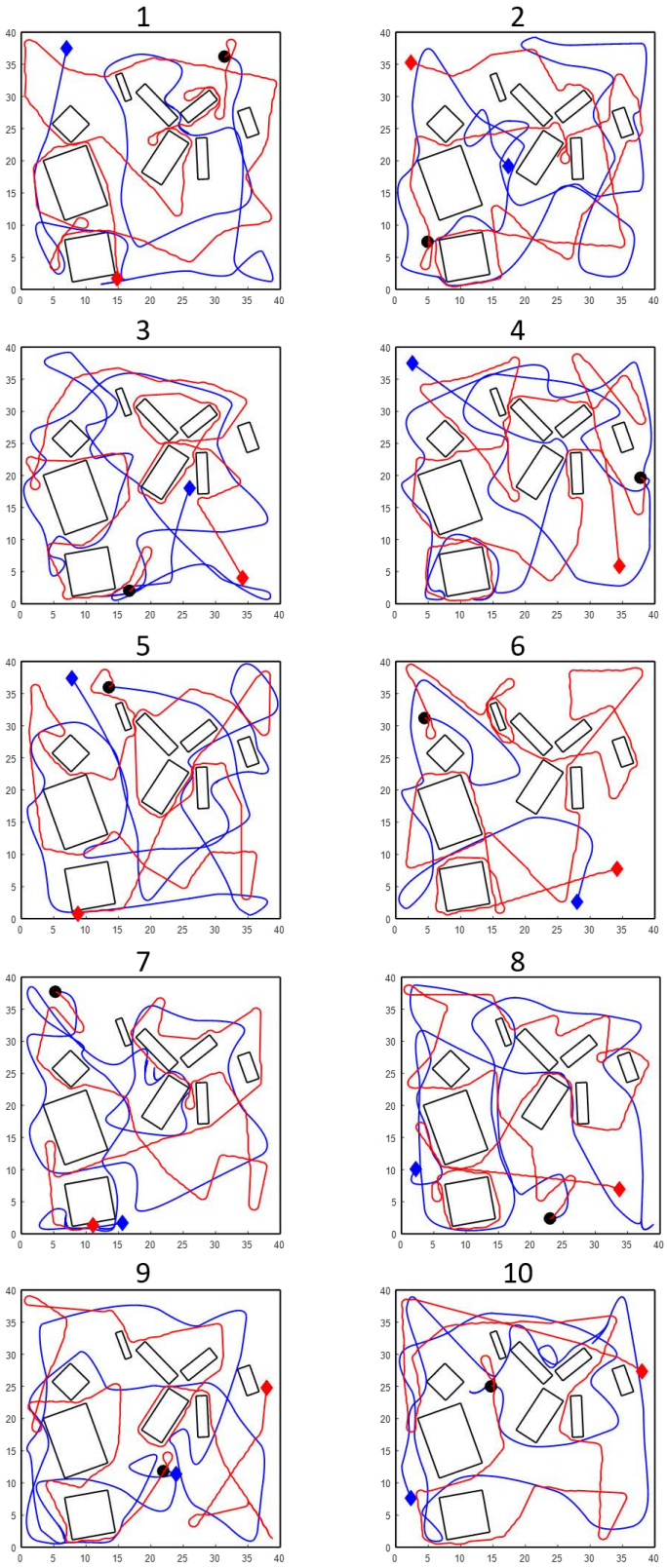



Figure 3.9: Repeatability tests. Each image depicts exploration paths in world 3 (refer to Fig. 3.6) from different starting poses, marked by a black dot. The blue and red lines are the exploration paths, CBE and frontier (Yamauchi, 1997), receptively. Each path ends with a corresponding diamond shaped marker.

Table 3.2: Repeatability - quantitative comparison of exploration paths originating from various starting poses as shown in Fig. 3.9. Comparison is between CBE and frontier (Yamauchi, 1997) exploration methods.

| # | Exploration Time [s] | | Diff [%] |
|---|---|---|---|
| | CBE | Frontier | |
| 1 | 128.9 | 107.3 | -16.8 |
| 2 | 89.8 | 111.9 | 24.5 |
| 3 | 68.7 | 104.6 | 52.2 |
| 4 | 96.9 | 103.9 | 7.3 |
| 5 | 72.7 | 97.4 | 34.1 |
| 6 | 61.1 | 78.7 | 28.8 |
| 7 | 61.8 | 98.6 | 59.7 |
| 8 | 88.4 | 91.4 | 3.4 |
| 9 | 99.5 | 98.5 | -1.0 |
| 10 | 70.0 | 76.9 | 9.8 |
| | | | |
| Average | 83.8 | 96.9 | 20.2 |
| std. dev. | 21.2 | 11.6 | 24.1 |

of path selection. As there are little differences in rewards along the paths, the end goal becomes the most important property of a path. Hence, a CBE planner would be potentially ineffective. By comparison, the frontier based-$A^*$ approach seems to be the most sensible method for such a problem as it moves the robot on the shortest path to the edge of the known space.

Figure 3.10 depicts the executed path of a robot exploring the surroundings of the Piazza San Marco,Venice, while Fig. 3.11 shows exploration around the Church of The Holy Sepulchre, Jerusalem. In both cases, the road network is complex, creating many possibilities for autonomous actions. From these qualitative comparisons, one can see that all techniques cover almost the entire mission area (blue square) with no isolated pockets of unexplored regions. However, a closer examination of the executed paths reveals a significant difference. As the map is *a-priori* unknown, it is reasonable that the robot will have to move occasionally through already mapped roads. However, the paths generated by the information-theoretic heuristics revisits known roads much more than the other planners. Although the search heuristics includes a distance penalty, it is not general enough to be effective in all scenarios. Once again, the reason behind such a sub-optimal performance lies in the basic properties of the global point planners i.e. separating the solution for the goal point from the subsequent path generation. We should note that in these scenarios the frontier planner is not as affected. Choosing the closest frontier as the planner's goal point keeps path planning in the robot's local neighbourhood. However, such an arbitrary goal

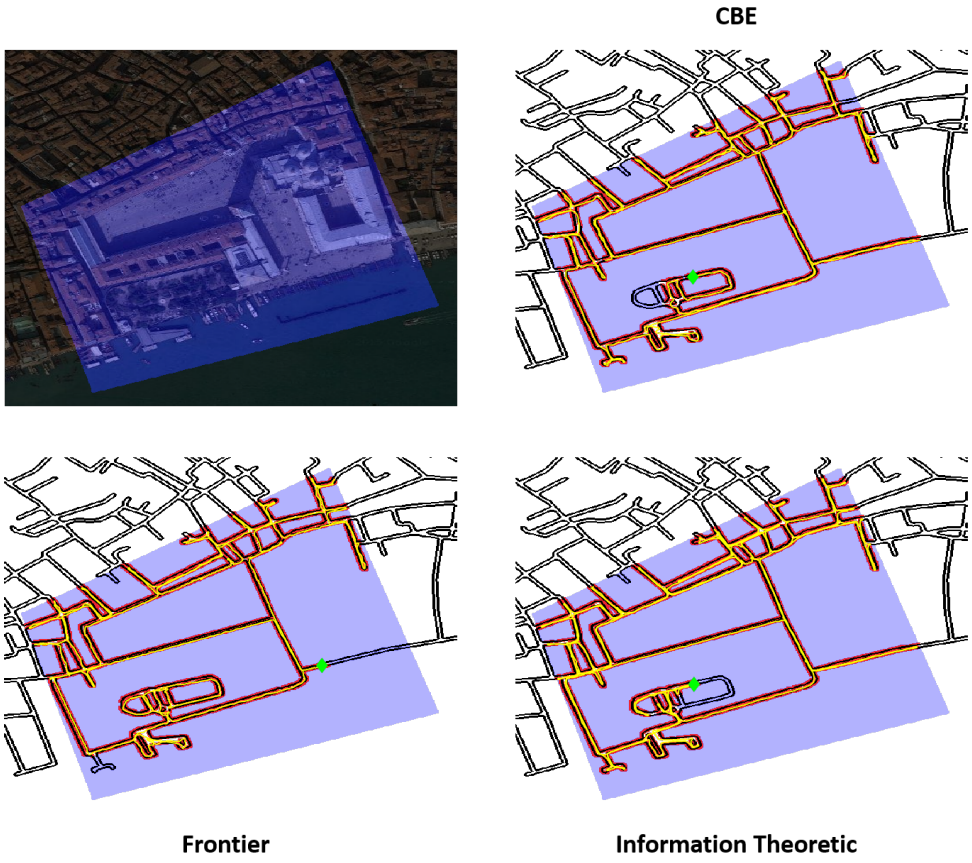



Figure 3.10: Comparison of simulation results of map building in Venice. (Top-left) Satellite image with a blue overlay that marks the mission area. In the remaining images, the black lines mark the ground truth for walls and obstacles. The blue overlay marks unexplored/unobserved regions, whereas explored areas are shown without the blue background. Occupied grid cells are marked with a red overlay. Traversable roads are extracted from Google Maps. The path the robot executed is in yellow, and the last position of the robot is marked with a green diamond. (top-right) CBE, (bottom-left) frontier and (bottom-right) information-theoretic based exploration. As expected, all methods explored almost the entire mission area. The paths generated by the information-theoretic method revisits explored region of the map much more than the other methods, although a distance penalty is incorporated in its reward heuristic. CBE and frontier present similar path structure as both plan mostly in the robot's close neighbourhood.

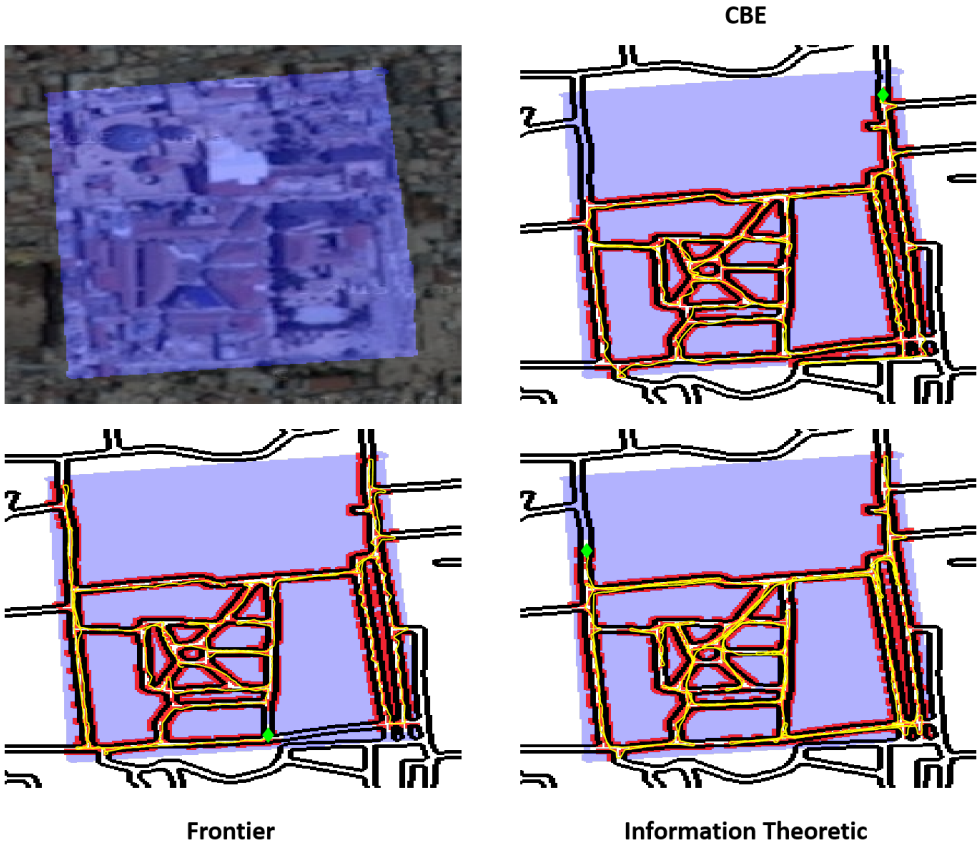



Figure 3.11: Comparison of simulation results of map building in Jerusalem old city. (Top-left) Satellite image with a blue overlay that marks the mission area. In the remaining images, the black lines mark the ground truth for walls and obstacles. The blue overlay marks unexplored/unobserved regions, whereas explored areas are shown without the blue background. Occupied grid cells are marked with a red overlay. Traversable roads are extracted from Google Maps. The path the robot executed is in yellow, and the last position of the robot is marked with a green diamond. (top-right) CBE, (bottom-left) frontier and (bottom-right) information-theoretic based exploration. As expected, all methods explored almost the entire mission area. The paths generated by the information-theoretic method revisits explored region of the map much more than the other methods, although a distance penalty is incorporated in its reward heuristic. CBE and frontier present similar path structure as both plan mostly in the robot's close neighbourhood.

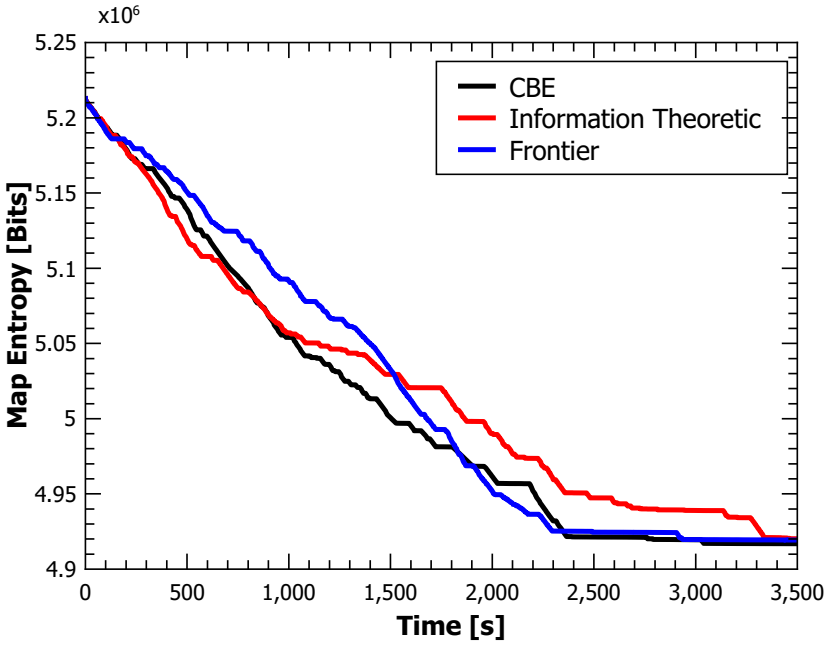



Figure 3.12: Venice - Comparison of reduction in map entropy between exploration methods presented in Fig. 3.10. The overall time to cover the mission area is similar with both CBE and frontier. Both methods outperform the information-theoretic method as the number of paths crossing already explored regions of the map is lower.

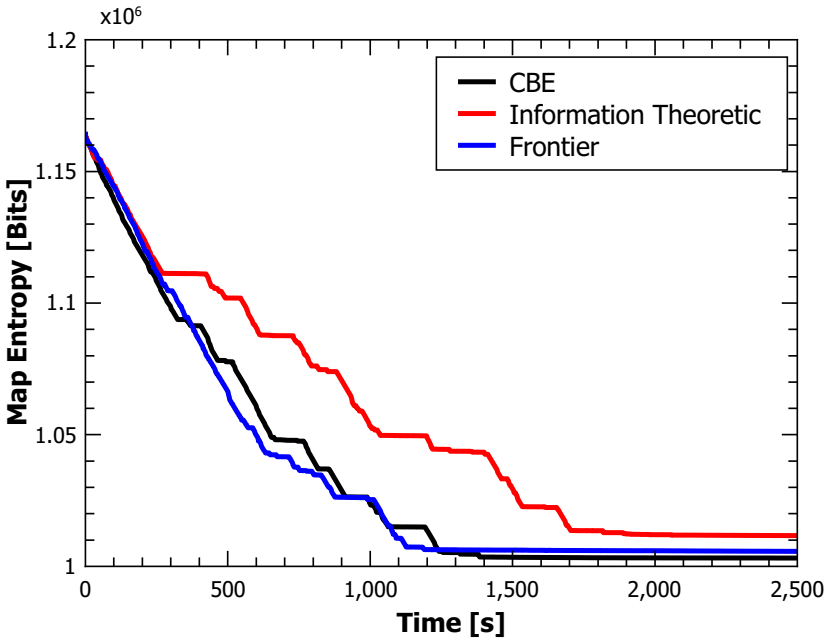



Figure 3.13: Jerusalem - Comparison of reduction in map entropy between exploration methods presented in Fig. 3.11. The rate of entropy reduction using CBE and Frontier is similar and outperforms the information-theoretic method which corresponds to lower instances of crossing already explored regions of the map.

point selection cannot always guarantee an optimal path.

A quantitative comparison for the Venice and Jerusalem exploration simulations are shown in Figs. 3.12 and 3.13, respectively. Most importantly, even in unfavourable conditions, CBE achieves performance as good as frontier. A careful inspection of the results reveals additional insights. Similarly to the unstructured environment simulations, the initial rate in which the map entropy drops is similar in all techniques. As the robot moves through the map, the number of possible actions increases leading to differences in performance. In Venice, frontier is less effective at first, while in Jerusalem the information-theoretic solution is less effective. Yet, in both experiments the CBE algorithm kept a consistent performance comparable or better than the other leading method. These results affirm the conclusion from our previous experiment that constrained BO is a robust exploration method.

The essence of the CBE method lies in its reasoning about the usefulness and safety of the entire path taken. The benefits of reasoning on the overall accumulated reward are more distinct when compared to the information-theoretic single goal point exploration technique (red line). The qualitative results shown in Figs. 3.10 and 3.11 expose the inefficiencies in the single goal point methods. The slower exploration rate in this method corresponds to the revisiting of already explored regions of the map whilst moving to the next goal point. By reasoning on the path utility instead of the end goal point, CBE avoids selecting paths that provide little information. Furthermore, the global component in the constrained BO reward function is found to be very effective in pulling robot away from dead-ends. Yet, a more expressive path option is more desirable in such a case, since it will allow longer term planning.

### 3.5.2 Real Environments

Simulations show the effectiveness of CBE for autonomous exploration. However, to assess performance with pose uncertainty, CBE was evaluated with a real robot mapping a cluttered office environment. We used our in-house robot, the Wombot (see Fig. 3.14), equipped with an i7-4500U 1.8GHz dual-core on-board PC and an Hokuyo UTM-30lx laser range finder, which can travel indoors with a maximum speed of 0.5m/s. As in simulations, we set $W_1 = 100$ and $W_2 = 50$..

We use *Robot Operating System* (ROS) (Quigley et al., 2009) to manage the communication between the various components of the robot i.e. sensors, actuators etc., and software modules. For mapping and localisation, we utilise externally provided ROS package, gMapping (Grisetti et al., 2007). It is worth noting that the aim of these experiments is to assess the performance in the presence of pose uncertainty. Although CBE can encode loop-closing heuristic in its

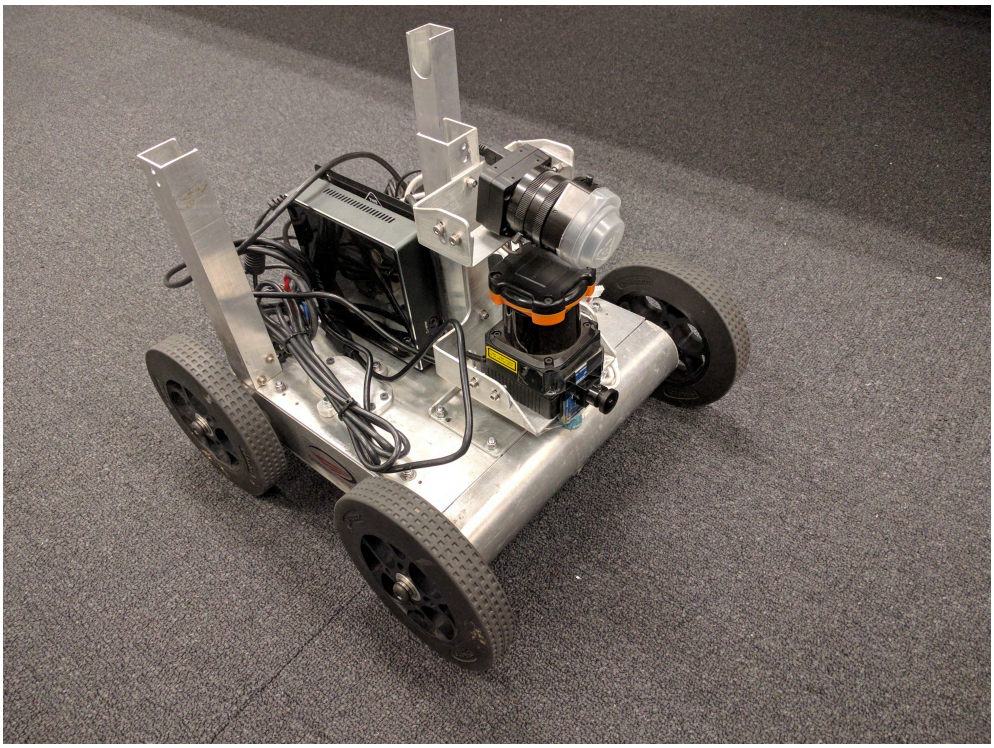


Figure 3.14: The Wombot - a mobile robot equipped with a Hokuyo UTM-30lx laser range finder.

objective function, we only used the objective function defined in Algorithm 5.

Figure 3.15 shows the maps and paths taken at different time stamps for both CBE and frontier. Similarly to experiments in simulations, the frontier planner places a goal point at the closest frontier. Figure 3.15 shows that both methods cover the entire space. However, while both stay clear of obstacles, the path taken by the frontier planner is less efficient. Even at $t = 50s$ CBE covers a larger area of the map and the gap in coverage between the methods widen as exploration continues. The path taken by frontier is longer as the planner takes unnecessary maneuvers. The occluded space behind the various obstacles forms frontiers "traps" that are then visited by the robot. As the robot visits these goal points, the overall path length and exploration time increase. CBE, on the other hand, considers the utility of the entire path as opposed to only considering the utility of the goal point, resulting a shorter and more efficient path. Furthermore, assessing safety using sigma paths that considers the effect of the uncertain pose with robot motion proved to be successful.

Figure 3.16 provides a quantitative comparison between the two methods. In the beginning, both methods perform similarly. After about 50 seconds the two methods start to diverge as the frontier planner pulls the robot towards a goal point behind an obstacle. As the exploration continues, the performance gap between the two methods increase, mainly due to a non-optimal goal point selection

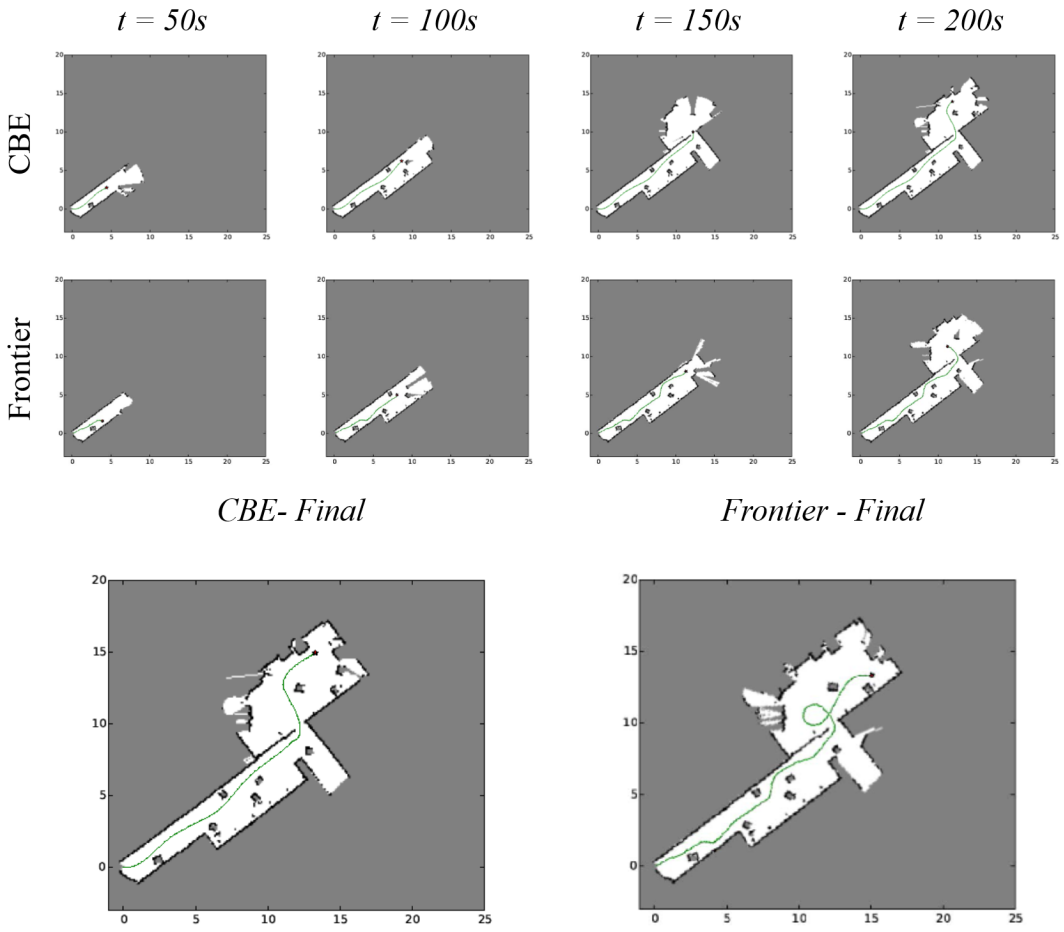



Figure 3.15: Map building comparison in a real environment. Each image represents the available map at a specific time stamp shown on the left. The frontier method produces a map less efficiently as it chooses unnecessary long paths due to frontiers forming in occluded space behind the various obstacles.

by the frontier planner. The final map, and the reduction in overall entropy is the same in both methods, although it took frontier roughly 30 seconds longer to do so.

The advantage of CBE comes with a computational cost as shown in Table 3.3. Finding a frontier and planning a safe path from the robot pose to a specific goal point depends of the size of the map and the distance to the goal point. Typically in our experiments, planning took less than 1 second. However, the search for a safe path took longer when there was no safe path to the selected frontier. In such a case, a safe path to a different goal point was calculated, but only after the first search had exhausted its time budget. CBE is more stable despite the higher computational cost.

As discussed in Section 3.4, updating and querying the GP models are the computational bottlenecks of CBE. As the main goal of this paper was to address

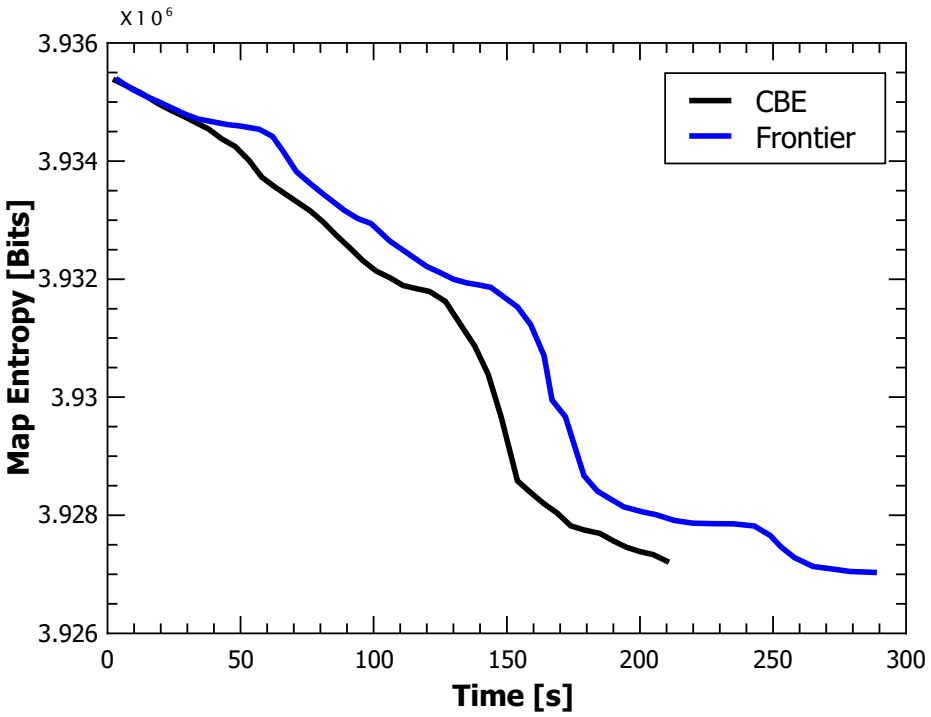



Figure 3.16: Autonomous exploration with a real robot - Comparison of reduction in map entropy between exploration methods presented in Fig. 3.15; The frontier planner visits the occluded space behind the various obstacles leading to longer path and exploration time. CBE maximises the information gain along its path. Therefore, the path avoids unnecessary maneuvers, which results in an efficient exploration.

Table 3.3: Comparison of the average planning and execution time

| | BO | Frontier |
|---|---|---|
| Planning [s] | 58.4 | 9.8 |
| Path Execution [s] | 192 | 236 |

the accuracy and validity of CBE with respect to the exploration problem, the computational complexity was not considered in the algorithm's design. However, an array of approximations for GP regression and classification can be used to address the computational complexity of GP inference. In his work on sparse GP models, Titsias (2009) proposed an approximation method based on variational inference with a complexity of $\mathcal{O}(m^2N)$, where $m$ are inducing points. Big data GP approximations follow an even lower complexity of $\mathcal{O}(m^3)$ for both regression (Hensman et al., 2013) and classification (Hensman et al., 2015). Further reduction in complexity can be achieved by non-linear logistic regression classifiers (Bishop, 2006), with a computational cost that is independent of the number of data points and linear in the number of features.

## 3.6 Summary

This chapter introduces a new strategy for safe autonomous exploration over continuous path. Its novelty lies in the holistic probabilistic approach to robotic exploration. Specifically it presents the following contributions:

(i) Formulation of autonomous exploration as an optimisation problem over constrained continuous space, where the path is evaluated by its accumulated reward and not only by the reward of its goal point. Traditional exploration methods consist of a two-step solution. First, a collection of goal points (typically one) is defined by a set of heuristics followed by a path planning step. As a result, the expected usefulness of the resolved path is based solely on the utility function of the end point and does not consider any potential gains along the way. Our new strategy, on the other hand, does not set goal points. Rather, it optimises the path selection by learning the properties of the objective function and any associated constraints. Consequently, the full potential of the robot trajectory can be exploited and not only that of the end point.

(ii) Constrained Bayesian Exploration as a holistic approach to safe exploration. This method directs the optimisation process in the presence of unknown constraints and risks. Hence, it provides a principled and robust approach for optimising exploration using Bayesian optimisation. As such, CBE guarantees convergence to a local solution and follows well known BO's regret bounds.

Utilising Bayesian inference, the optimiser learns the models of the rewards and constraints. These models are then used to generate a coherent objective function that incorporates gains, costs and risks of any path, allowing efficient identification of potential optimal solutions that satisfy the constraints with high confidence. As the actual model of the objective function and constraints is learned online, incorporating different objectives or constraints is straightforward. Therefore, CBE allows a relatively simple and smooth application of other limitations or objectives such as energy and time budget. limitations, such as energy and time budget, or objectives, e.g. the Cauchy-Schwarz quadratic mutual information (Charrow et al., 2015).

To test the robustness and consistency of our method, we compared its performance with traditional and state-of-the-art exploration techniques. The results show that the performance of the other exploration techniques depends on the layout of the environment. By reasoning about the usefulness of the entire path instead of only its goal point, CBE exhibits a robust and consistent performance

in all tests. Even in unfavourable conditions of structured environments, CBE performs better than or as good as the leading method.

The use of sigma paths to incorporate localisation uncertainty proved successful. A robot travelling through cluttered office space managed to avoid obstacles while still optimising the cost function.

CBE provides a global optimisation framework for exploration along continuous paths. However, it also carries several disadvantages. One major disadvantage is the computational cost, which arises from the cubic cost of updating and querying GPCs. A promising approach to alleviate this restriction is to utilise stochastic variational inference for GPC (Hensman et al., 2015), which has a computational cost independent of the number of data points.

A general disadvantage of Bayesian optimisers is their low-effectiveness for high dimensional problems, also known as the curse of dimensionality. As the dimensionality of the problem increases, the density of observations decreases exponentially (Györfi et al., 2002). As a result, the GP surrogate models can not represent well the objective function. With low fidelity surrogate models, the optimiser is no longer effective. To overcome this problem, we followed practical limitation on BO dimensionality ($<10$) (Snoek et al., 2012) and restricted the family of trajectories used for CBE. However, the use of a predefined family of trajectories, such as quadratic or cubic splines, limits the decision space of the robot. This problem is more pronounced when path planning is constrained, for example near obstacles, as the optimisation space is confined. Examples of more expressive path generation already exists in the literature (Yang et al., 2013; Charrow et al., 2015). However, these methods do not consider the overall reward along the path, or only locally optimise the path selection. Combining exploration with an expressive RKHS motion planning method, such as the method presented by Marinho et al. (2016), may allow for global optimisation of highly expressive paths. This approach is developed in the following chapters of this thesis.

# Chapter 4

# Stochastic Path Planning in Continuous Occupancy Maps

## 4.1 Introduction

Motion planning is a basic building block in autonomous robotics. Essentially, it is a decision making process that ensures safe travel from the robot's current configuration to its goal. As safety is the primary objective, the planned trajectory must avoid collision with obstacles. It is a prolific branch of robotics that has been studied for decades, producing a wide range of planning methods which can be categorically grouped into two main branches; sampling-based planning and trajectory optimisation.

Planning a safe path using an *Occupancy Grid Map* (OGM) is typically done by sampling-based planners (Tsardoulias et al., 2016). Most planners break the planning process into two phases. First, the planner finds a feasible, collision-free, crude path. Then, the following step improves the resulting path by applying certain heuristics.

Trajectory optimisers optimise an objective function such as control cost or execution time. However, there are no trajectory optimiser implementations for path planning using occupancy maps. The main challenge lies in the optimiser's need for contextual information anywhere along the path. Gaps or non-informative gradients will cause the optimiser to converge into a non-optimal and unsafe solution.

In this chapter, we present a new planning paradigm using occupancy maps. We utilise the recently introduced Hilbert maps (Ramos and Ott, 2015) instead of OGMs. Hilbert maps provide a fast and continuous linear discriminative model for occupancy mapping. We take advantage of the fact that spatial gradients of the occupancy can be calculated in closed form, and use them in the functional gradient motion planner update cycle. We present a novel path planner based on a *Gaussian Process* (GP) path representation. Unlike other functional gradient path planning techniques (e.g. (Marinho et al., 2016), (Zucker et al., 2013)), the proposed planner does not commit to a predetermined resolution, whether spatial or parametric. It replaces the regularisation of the step size used in the functional

gradient method with a stochastic gradient approach. This is a key element in the planner's optimisation strategy as it allows a resolution-free gradient update, which is required to ensure convergence.

The contributions presented in this chapter are:

1. A novel path optimisation approach for continuous occupancy maps. This method extends previous work done on discrete cost maps to a continuous environment representation.

2. A stochastic functional gradient motion planner based on GP path representation. The stochastic samples allow flexible support for the path, instead of an a-priori fixed set as used in prior work.

.

The remainder of this chapter is organised as follows. Section 4.2 reviews the literature on path planning using occupancy maps. Section 4.3 extends the general functional optimisation framework in Section 2.3 for path planning in cost maps. The modifications required for planning in occupancy maps and the algorithm for FGD path planning in Hilbert maps are detailed in Section 4.4. The experimental results, in simulation and with real laser data, are described in Section 4.5. Section 4.6 summarises the proposed method and findings.

## 4.2 Related Work

Path planning using occupancy maps is commonly approached by sampling-based methods with several very successful algorithmic families such as: *Rapidly exploring Random Trees* (RRT), *Probabilistic RoadMap* (PRM), *Visibility Graphs* (VG) and Space Skeletonisation (see review by Tsardoulias et al. (2016)). Sampling-based methods typically work by first building a graph representation of the configuration space, where edges represent valid connections. After the graph is built, a valid path is obtained using a search algorithm on the graph structure. The visibility graphs method builds a graph where the nodes are the vertexes of the obstacles (Lozano-Pérez and Wesley, 1979). Space skeletonisation uses *Generalised Voronoi Diagram* (GVD) to compute safe paths (Bhattacharya and Gavrilova, 2007; Garrido et al., 2006). PRM randomly samples the configuration space for free space configuration and then uses a local planner to find edges to connect these configurations to existing nodes (Kavraki et al., 1996). Next, a tree search method is used to determine the path. Another successful and prolific method is RRT, which randomly grows a tree rooted at the start configuration

The theory and results in this chapter were published in (Francis et al., 2017)

(Lavalle, 1998). The main drawback of sampling based methods is that while they are very successful in finding safe paths, there is no explicit optimisation of an objective function, such as length or smoothness.

Optimisation is a widely used approach for finding feasible paths, where the planned path is the local extrema of a pre-defined arbitrary cost function. Loosely speaking, the cost function captures the costs and penalties associated with a configuration-space state, e.g. distance from obstacles. Khatib (1986) pioneered the use of artificial potential field for collision avoidance. *Covariant Hamiltonian Optimisation for Motion Planning* (CHOMP) utilises covariate gradients from a precomputed obstacle cost to minimise the trajectory's obstacle and smoothness functionals (Zucker et al., 2013). The *Stochastic Trajectory Optimisation for Motion Planning* (STOMP) planner uses noisy perturbations to perform optimisation under constraints where the cost functional is non-differentiable (Kalakrishnan et al., 2011). Both CHOMP and STOMP commit to a waypoint representation which require to trade-off expressiveness with computational costs. Mukadam et al. (2016) proposed the Gaussian process motion planner which uses a Gaussian process generated by linear time varying stochastic differential equations for path representation. Marinho et al. (2016) perform trajectory optimisation in a RKHS. However, all these methods fall short when planning using occupancy maps as discussed in section 4.4.2.

## 4.3 Functional Gradient Path Planning

In this section, we extend the general functional optimisation framework in Section 2.3 for path planning. We first re-define the notation of Section 2.3. A path, $\xi : [0,1] \rightarrow \mathcal{C} \in \mathbb{R}^D$, is a function that maps time, $t \in [0,1]$, into configuration space $\mathcal{C}$. We define an objective functional, $\mathcal{U}(\xi) : \Xi \rightarrow \mathbb{R}$, that returns a real number for each path $\xi \in \Xi$. The objective functional captures the path optimisation criteria, such as path safety and kinematic costs.

The objective functional, $\mathcal{U}(\xi)$, varies between the different planning methods. However, it is typically a weighted sum of two penalties:

$$\mathcal{U}(\xi) = \mathcal{U}_{obs}(\xi) + \lambda \mathcal{U}_{dyn}(\xi) m \tag{4.1}$$

where;

(i) $\mathcal{U}_{obs}(\xi)$ penalises proximity to obstacles;

(ii) $\mathcal{U}_{dyn}(\xi)$ regulates either the curve shape or motion dynamics:

As obstacles are defined in the robot's working space $\mathcal{W} \in \mathbb{R}^3$, estimating the

obstacle cost functional is done by mapping a path from configuration space into workspace using a forward kinematic map, $x$. Given $\mathcal{B} \in \mathbb{R}^3$, a set of points on the robot, $x\left(\xi(t), u\right)$ maps a robot configuration, $\xi(t)$ and body point $u \in \mathcal{B}$ to a point in the workspace $x : \mathcal{C} \times \mathcal{B} \rightarrow \mathcal{W}$. Then, the obstacle cost functional is estimated by aggregating the workspace cost function, $c : \mathbb{R}^3 \rightarrow \mathbb{R}$, along the trajectory and robot body points using a *reduce* operator such as an integral or a maximum. The only requirement is that the *reduce* operator can be approximately represented by a sum over a finite set, $\mathcal{T}(\xi) = \{t, u\}_i$ of time $t_i$ and body points $u_i$:

$$\mathcal{U}_{obs}(\xi) \approx \sum_{(t,u)\in\mathcal{T}(\xi)} c\left(x\left(\xi(t), u\right)\right). \tag{4.2}$$

$\mathcal{U}_{dyn}(\xi)$ is a secondary objective functional, which typically penalises a path based on its kinematic costs or curve properties. The exact choice depends on the implementation and path representation used. In most cases the penalty deals with derivatives of the path, for example in (Zucker et al., 2013) the squared velocity norm was used as the dynamic penalty:

$$\mathcal{U}_{dyn}(\xi) = \frac{1}{2}\int_0^1 \left\| \frac{d}{dt}\xi(t) \right\|^2 dt. \tag{4.3}$$

In (Marinho et al., 2016), the optimisation regulariser was the $L_2$ norm of $\xi$ which implicitly assumes the zero-line, connecting starting point to the goal point, as the preferable solution. We will show in Section 4.5 that such a regulariser is not suitable when planning using occupancy maps.

Given the cost functional in Eq. (4.1), the functional gradient is given by:

$$\nabla_\xi \mathcal{U} = \nabla_\xi \mathcal{U}_{obs}(\xi_n(\cdot)) + \lambda \nabla_\xi \mathcal{U}_{dyn}(\xi_n(\cdot)). \tag{4.4}$$

Optimisation of $\xi$ can be performed by an iterative approach as described in Eq. (2.37). Applying the functional gradient of Eq. (4.4) to Eq. (2.37) leads to the overall update rule;

$$\xi_{n+1}(\cdot) = \xi_n(\cdot) - \eta_n A^{-1}\left[\nabla_\xi \mathcal{U}_{obs}(\xi_n(\cdot)) + \lambda \nabla_\xi \mathcal{U}_{dyn}(\xi_n(\cdot))\right]. \tag{4.5}$$

## 4.4 FGD Using Hilbert Maps

In this section we discuss the shortcoming of the current functional gradient methods when planning paths in occupancy maps. We then propose a method that utilises a continuous GP path representation combined with stochastic sampling to solve this problem using Hilbert maps.

Occupancy maps create several challenges for gradient based path planners. Most importantly, gradient information in an occupancy map is not necessarily useful. The obstacle cost functional defined in Eq. (4.2) requires a workspace cost function, $c(x(\xi(t), u))$. Most trajectory optimisers work with a precomputed distance field as a cost map, which produces noiseless and informative gradients anywhere in the map. Estimating the obstacle cost in an occupancy map is more challenging. While the occupancy map (or blurred map if using OGM) can act as $c(x(\xi(t), u))$, the obstacle cost is only well defined in observed areas of the map. Even when OGM is converted to an SDF using a *Truncated Signed Distance Function* (TSDF) (Curless and Levoy, 1996), it is unable to provide informative gradients in unobserved regions (Oleynikova et al., 2017).

Figure 4.1 illustrates the differences between the precomputed cost used by most FGD motion planners and a standard occupancy map (Elfes, 1989). Figure 4.1a shows a cost map with complete knowledge of the obstacle. The cost, given in closed-form, and its spatial gradient are defined everywhere in the map as indicated schematically by the arrows. Figure 4.1b illustrates the equivalent occupancy grid map, which is inferred from laser observations. As expected, The grid map only holds information about observed locations. In those regions, the cost follows the occupancy. However, in the unobserved regions of the map, the two approaches generate different outputs. While the precomputed cost still produces a well-behaved function and the desired repulsive gradient, the occupancy returns to the map's prior occupancy probability of 0.5 and generates inconsistent gradients.

The main challenge of using FGD on occupancy maps lies in the inability to define a usable gradient everywhere in the map. Cross sectional data of the precomputed cost and occupancy maps, as depicted in Fig. 4.1c, summarises this. In observed areas, the occupancy can act as the obstacle cost, as spatial gradients of both maps "pushes" away from obstacles. However, in occluded regions the behaviour is entirely different. While the precomputed gradient still returns a repulsive gradient, pushing away from the obstacle, the spatial gradient of the occupancy map pulls inward, toward unobserved and unsafe regions of the map. Conversion of an OGM to DSF using TSDF (Curless and Levoy, 1996) does not alleviate this problem, as TSDF cannot interpolate over unobserved regions of the map. While Oleynikova et al. (2017) suggest TSDF for planning, they acknowledge TSDF inability to produce useful gradients everywhere in the map, requiring multiple re-planning during execution.

The other challenge arising from planning with trajectory optimisers using occupancy maps is that all planners commit to a specific path parametrisation to solve Eq. (2.37). Waypoint parametrisations, as used by (Zucker et al., 2013;

Kalakrishnan et al., 2011; Park et al., 2012), have to find a balance between expressiveness and computational complexity. More recent work on functional gradient motion planning in RKHS (Marinho et al., 2016) and the Gaussian process motion planner (Mukadam et al., 2016; Dong et al., 2016) define a parametric support with finite resolution for their path representation. These methods produce highly expressive trajectories. However, given the finite resolution of the support, the spatial density of the support points changes, as the optimisation process deforms the trajectory. Consequently, some areas in the workspace have low support density, which results in a low update rate. This problem is exacerbated when using occupancy maps as some regions in the map have no informative gradients. To prevent the optimisation process from becoming corrupted, such uninformative updates are rejected, reducing even further the effective density of the support.

The following sections address these issues. The resulting path planning algorithm computes in closed-form the obstacle functional gradients from a Hilbert map. It then uses a stochastic sampling schedule and an iterative GP representation to generate an expressive path over a flexible stochastic support.

### 4.4.1 Occupancy Gradient in Hilbert Maps

In our approach, the spatial cost function $c(x(\xi(t), u))$ is the Hilbert occupancy map, which is estimated by Eq. (2.45) along the trajectory and robot body points. Although Hilbert maps do not form a tangible grid as OGM, querying the spatial occupancy gradient is as straightforward as querying an occupancy grid. By applying the chain rule, the Euclidean space gradient of Eq. (2.45) around a query point, $\mathbf{x}^*$ becomes:

$$\begin{aligned} \frac{\partial}{\partial \mathbf{x}^*} p(y^* = +1 | \mathbf{x}^*, \mathbf{w}) &= \frac{\partial \sigma(\mathbf{z_{NL}}(\mathbf{x}^*, \mathbf{w}))}{\partial \mathbf{z}_{NL}} \frac{\partial \mathbf{z}_{NL}}{\partial \mathbf{x}^*} \\ &\approx \sigma(\mathbf{z}_{NL})(1 - \sigma(\mathbf{z}_{NL}))\mathbf{w}^T \frac{\partial}{\partial \mathbf{x}^*} \hat{\Phi}(\mathbf{x}^*). \end{aligned} \tag{4.6}$$

Here we used the fact the derivative of LR is given by $\frac{\partial}{\partial \mathbf{x}} \sigma(\mathbf{x}) = \sigma(\mathbf{x})(1 - \sigma(\mathbf{x}))$.

### 4.4.2 GP Paths using Hilbert maps

GPs provide a principled way to represent smooth trajectories. The GP model requires a small set of support points, which define waypoints in configuration space that the path should follow. GP regression provides us with a complete solution that allows querying the model at any given time and handles boundary conditions we wish to impose. Unlike other GP-based motion planners (Mukadam et al., 2016; Dong et al., 2016), the method presented here does not require a

predefined support, but rather learns and builds the trajectory support while optimising the path.

A GP path is defined as a vector-valued (multiple output) GP (Alvarez et al., 2012):

$$\xi(t) \sim \mathcal{GP}(\mu(t), k(t, t')), \qquad t, t' \in [0, 1]. \tag{4.7}$$

Here, $\mu(t) \in \mathbb{R}^D$ is the vector-valued mean function of $t$ and $k(t, t') \in \mathbb{R}^D \times \mathbb{R}^D$ is a positive matrix-valued kernel between $\xi(t)$ and $\xi(t')$ with a corresponding kernel matrix for two different time instances, $K(t, t')$:

$$K(t, t') = \begin{bmatrix} k_{1,1}(t, t') & k_{1,2}(t, t') & \dots & k_{1,D}(t, t') \\ \vdots & \ddots & & \vdots \\ k_{D,1}(t, t') & k_{D,2}(t, t') & \dots & k_{D,D}(t, t') \end{bmatrix}. \tag{4.8}$$

As $\xi(t) \in \mathbb{R}^D$ defines a state in configuration space, each element in $K$, $k_{d,d'}(t, t')$ $d, d' = 1, ..., D$ represents the effect joint $[\xi(t)]_d$ at time $t$ has on joint $[\xi(t')]_{d'}$ at $t'$.

Updating the model requires conditioning the GP model with waypoint observations. The term observations here is used loosely, and means the states, $\xi^o(t^o)$ at time $t^o$ that the trajectory must pass through. By conditioning the GP with these observations we can compute the *maximum a posteriori* (MAP) configuration at any query time, $t^*$:

$$\bar{\xi}(t^*) = \mu(t^*) + K(t^*, t^o)K(t^o, t^o)^{-1}(\xi^o(t^o) - \mu(t^o)). \tag{4.9}$$

The choice of $t^o$ differentiates this work from other GP path representations, such as (Mukadam et al., 2016; Dong et al., 2016). In prior work, gradient observations were taken on a fixed set of observation points. Meaning that $t^o$ was fixed a-priori and as a result the path support was fixed a-priori. In this work, on the other hand, $t^o$ is learned online, by accepting and rejecting stochastic samples from the occupancy map.

There are several advantages of using GPs to represent the path. First, we do not need to discretise the path. Instead a finite set of $N$ points, $\{t_i^o, \xi_i^o(t_i^o)\}_{i=1}^N$, serve as the curve support, which can be queried for its MAP value at any given time $t^*$ using Eq. (4.9). Second, the mean function $\mu$ provides an explicit prior, which can be exploited by initialising the optimisation with a rough path from a fast path planning method. Finally, boundary conditions can be imposed by treating them explicitly as observations. Besides the obvious boundary conditions at the start and goal points one can define must-visit waypoints along the trajectory or define the robot's direction by including derivative observations (Solak

et al., 2002).

### 4.4.3 Stochastic Gradient

A drawback of other functional gradient motion planners is that they either utilise a spatial parametrisation or commit to a finite parametric resolution to represent and update the path. In both cases, this may lead to gaps in the sampling of the objective functional. To overcome this we adopt a resolution-free sampling method. Since the functional objective of Eq. (4.1) can be approximated by a sum of individual points along the path, optimisation of the objective can be performed using *stochastic gradient descent* (SGD) (Bottou et al., 2016). From Eqs. (4.1)-(4.3), we define an empirical objective functional that approximates the real objective:

$$\hat{\mathcal{U}}(\xi) = \sum_{t,u \in \mathcal{T}} \mathcal{U}_{obs}\left(\xi(t,u)\right) + \lambda \mathcal{U}_{dyn}\left(\xi(t,u)\right) \xrightarrow[n\to\infty]{} \mathcal{U}(\xi). \tag{4.10}$$

A consequence of Eq. (4.10) is that minimising the objective requires reasoning over many points all along the curve. Such a process, which effectively resembles batch optimisation, is computationally infeasible. The approach taken by other trajectory optimisation methods (e.g. (Marinho et al., 2016; Mukadam et al., 2016)) is to estimate $\hat{\mathcal{U}}(\xi)$ with a finite resolution support. However, it is clear from Eq. (4.10) that while this is computationally attractive, such an approach cannot guarantee convergence to the optimum.

The stochastic functional gradient path planner utilises SGD to ensure convergence under the SGD guarantees (Bottou et al., 2016). Generally, SGD randomly selects a mini-batch from a dataset and then updates the solution in small steps, based on the gradient computed from that mini-batch. In the path planning problem, there is no dataset per se. Instead, the dataset is built during optimisation by samples taken over the entire $[0, 1]$ domain. Consequently, our method utilises SGD's update rule and uses random samples $[t^*, u^*, \xi_n(t^*)]$ as stochastic training points. The update rule of Eq. 2.37 is then replaced by:

$$\xi_{n+1}(\cdot) = \xi_n(\cdot) - \eta_n A^{-1} \nabla_\xi \mathcal{U}(\xi_n(t^*), u^*) \tag{4.11}$$

where $\eta_n > 0$ is the step size parameter and can be either constant or asymptotically decaying. Matrix $A$ can be seen as a preconditioner that may help accelerate convergence rate, and in many cases is set to the identity matrix (Bottou et al., 2016).

By conditioning the step size $\eta_n$, SGD ensures the convergence, in expectation, of the empirical objective in Eq. (4.10) to the optimal objective, while keeping

computational cost per iteration low (Bottou et al., 2016).

### 4.4.4 Planning on Hilbert Maps Algorithm

To finalise the stochastic functional gradient path planning algorithm, we need to define the functional gradient of $\mathcal{U}_{obs}$ and $\mathcal{U}_{dyn}$. Applying Eq. (2.38) to $\mathcal{U}_{obs}$ at a sampled time $t^*$ and for robot body point $u^*$ yields;

$$\nabla \mathcal{U}_{obs}(\xi(t^*), u^*) = \frac{\partial}{\partial \xi(t)} x(\xi(t^*), u^*) \nabla_x c\left(x\left(\xi(t^*), u^*\right)\right). \tag{4.12}$$

Here, $\mathbf{J}(t^*, u^*) \equiv \frac{\partial}{\partial \xi(t)} x(\xi(t^*), u^*)$ is the workspace Jacobian. $\nabla_x$ emphasises that this is a Euclidean gradient of the cost function $c$.

We opted to use the squared velocity norm integral, as shown in Eq. (4.3), as the dynamic penalty $\mathcal{U}_{dyn}$. Using the $L_2$ norm such as in (Marinho et al., 2016) is less attractive as it implicitly defines a favourable simple mean solution which requires tuning of regularisation coefficients for different scenarios. With Eq. (4.3), the functional gradient can be easily computed as:

$$\nabla \mathcal{U}_{dyn}(\xi(t^*)) = -\frac{d^2}{dt^2} \xi(t^*). \tag{4.13}$$

Again, as $\xi(t)$ is represented by a GP, computing the derivative is straightforward. The update rule at time $t^*$ and robot body point $u^*$ can now be summarised from Eqs. (4.11), (4.12), and (4.13) as:

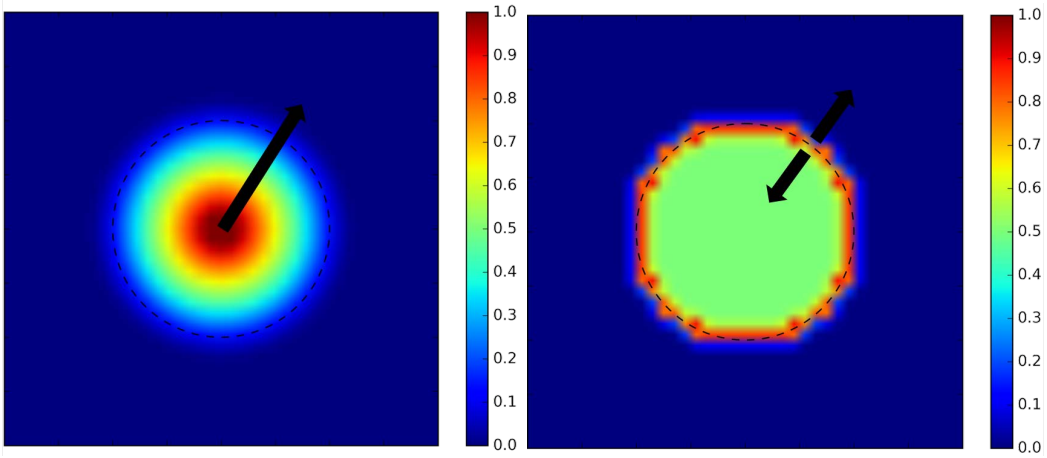
$$\xi_{n+1}(t^*) = \xi_n(t^*) - \eta_n A^{-1} \left( \mathbf{J}(t^*, u^*)^T \nabla_x c\left(x\left(\xi(t^*), u^*\right)\right) + \lambda \frac{d^2}{dt^2} \xi(t^*) \right). \tag{4.14}$$

The algorithm for path planning using Hilbert maps is shown in Algorithm 6. The algorithm accepts an optional initial solution to start the optimisation with, otherwise a straight line trajectory is used. This initial path is then used as the mean function of the GP path.

At each iteration, a mini-batch $(t_{sup}, u_{sup})$ is drawn randomly. For each point $t^* \in t_{sup}$ and $u^* \in u_{sup}$ the corresponding state $\xi_n(t^*)$ and workspace pose $x(\xi(t^*), u^*)$ are computed from the current GP path model, $\xi_n(t)$. To perform the functional update, the Hilbert map is queried at $x(\xi(t^*), u^*)$, and the probability of occupancy $P_{occ}$ is obtained. Functional updates may only happen if the occupancy is within safe limits, i.e. free from obstacles. Using Eq. (4.14) new states $\xi_{n+1}(t^*)$ are computed and the path model $\xi_n(t)$ is updated with the new path observations. To enforce a valid path, the boundary conditions are then incorporated into the GP model as additional observations. Finally, a new path model is initialised with the previous model as its mean function $\xi_{n+1}(t) \sim \mathcal{GP}(\xi_n(t), k)$.

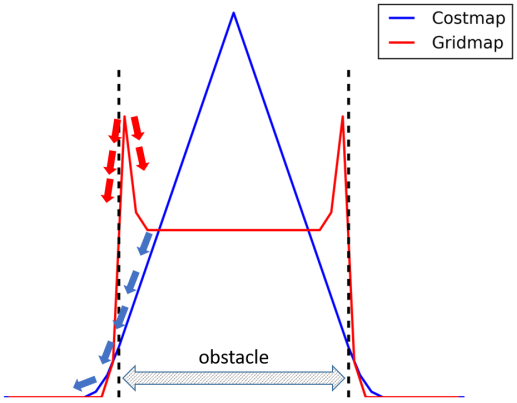
**Algorithm 6:** Functional gradient path planning using Hilbert maps

**Input:** $\mathcal{H}$: Hilbert Occupancy Map.
$\xi(0), \xi(1)$: Start and Goal states.
$P_{safe}$: No obstacle Threshold.
$\xi_{initial}(t)$: Initial solution (optional).
$k$: covariance function for GP path.
**Output:** $\xi_{min}(t)$
`// Use prior guess/solution if available`
**if** $\xi_{initial}$ **then**
  $\mu_0 \leftarrow \xi_{initial}$
**else**
  $\mu_0 \leftarrow (\xi(1) - \xi(0))\, t + \xi(0)$
**end**
$\xi_0 \leftarrow \mathcal{GP}_0 \sim \mathcal{GP}(\mu_0, k)$
$n = 0$
**while** $\xi$ *not converged* **do**
  `// Stochastic sampling`
  $(t_{sup}, u_{sup}) \leftarrow$ Draw mini-batch randomly
  **foreach** $(t^*, u^*) \in (t_{sup}, u_{sup})$ **do**
    $P_{occ} \leftarrow \mathcal{H}(x(\xi_n(t^*), u^*))$ Eq. (2.45)
    **if** $P_{occ} \leq P_{Safe}$ **then**
      $\xi_{n+1}(t^*) \leftarrow$ update rule Eq. 4.14
      $\xi_n(t) \leftarrow$ update GP with $(t^*, \xi_{n+1}(t^*))$
    **end**
  **end**
  `// Update boundary conditions`
  $\xi_n(t) \leftarrow$ update: $(0, \xi(0)), (1, \xi(1))$
  $\xi_{n+1}(t) \leftarrow \mathcal{GP}(\xi_n(t), k)$
  $n = n + 1$
**end**




(a) Precomputed cost map

(b) Occupancy Grid Map




(c) Cross section

Figure 4.1: Comparison of cost maps used in path planning. Dashed lines illustrate the obstacle boundaries. Arrows indicate the direction of gradient. (a) precomputed cost field as specified in (Zucker et al., 2013). (b) shows an occupancy map based on sensors observations.(c) depicts cross sections of the map in 4.1a and 4.1b, which emphasise the difference in spatial gradients, indicated by corresponding coloured arrows. In the cost map, cost is defined everywhere in the map and always generates repulsive gradient. The direction of gradient in occupancy map is inconsistent around the obstacle borders, with attractive potential in unsafe areas of the map.

## 4.5 Experiments

In this section, we evaluate the performance of the stochastic functional gradient path planner in simulation and with real data. We show that other functional gradient methods such as (Marinho et al., 2016) fail when planning using occupancy maps. We also provide comparisons to other planning methods for occupancy maps such as RRT* (Karaman, 2010) and PRM* (Karaman and Frazzoli, 2011) .

### 4.5.1 Simulations

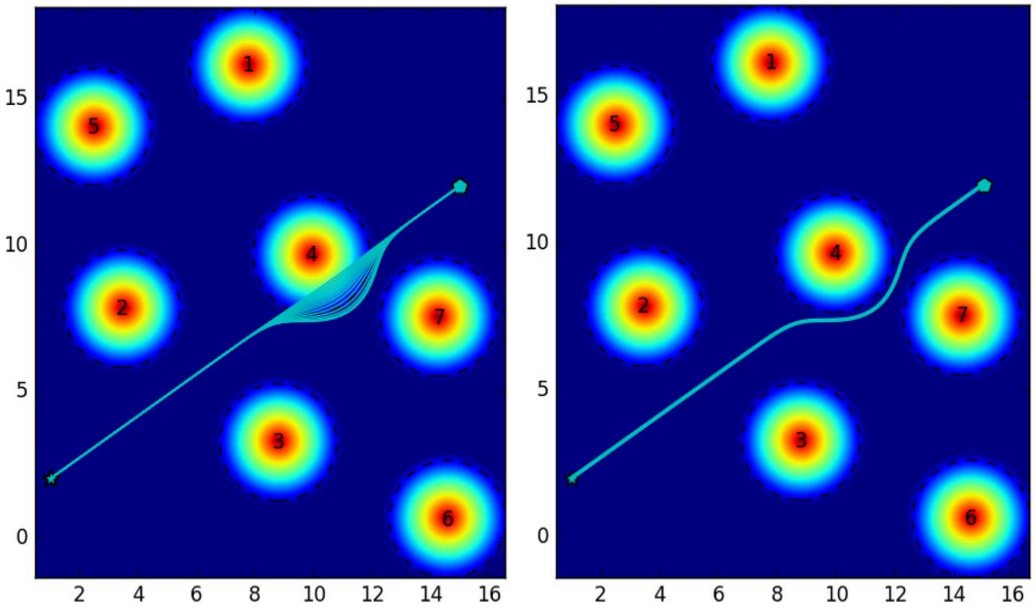
In this section we compare the proposed method with the functional gradient motion planner in RKHS (Marinho et al., 2016), as both methods are related. We show that while (Marinho et al., 2016) provides a flexible path representation, changes are needed in order to perform optimisation in occupancy maps.

Most trajectory optimisers assume full knowledge about the location of obstacles and precompute offline a cost field, $c(\mathbf{x})$ in workspace $\mathcal{W}$ which penalises the proximity to obstacles, for example (Ratliff et al., 2009):

$$c(\mathbf{x}) = \begin{cases} -d(\mathbf{x}) + \frac{1}{2}\epsilon & d(\mathbf{x}) < 0 \\ \frac{1}{2\epsilon}(d(\mathbf{x}) - \epsilon)^2 & 0 \leq d(\mathbf{x}) \leq \epsilon \\ 0 & \text{otherwise} \end{cases}, \tag{4.15}$$

where $d(\mathbf{x})$ is the distance of $\mathbf{x}$ to the boundary of the nearest obstacle and $\epsilon$ is a minimal safety buffer from obstacles. An example of path planning with a precomputed cost field based on (Marinho et al., 2016) is shown in Fig. 4.2: 4.2a depicts the iterative optimisation process and 4.2b shows the optimal path. These noiseless obstacles result in a cost gradient that is well-defined anywhere in the workspace, including inside obstacles. As the gradient is precomputed, evaluating the update rule is also computationally efficient, leading to fast convergence to a local minima. However, planning using occupancy maps adds several challenges to this optimisation process. The cost field, and more importantly its spatial gradients, are not necessarily informative. In addition, as the map is generated by laser observations, all predictions are noisy. As a result, the assumptions at the core of the functional gradient-based planner are no longer valid and a change to the planning process is required.

Constructing a Hilbert map for a simulated environment of randomly placed obstacles is achieved by generating an occupancy dataset. The dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_1^N$ is created by randomly placing occupied observations, $y_i = +1$, on the boundaries of all obstacles and free observations $y_i = -1$ outside obstacles. There are no laser points inside obstacles. After the dataset is created, the map model




(a) Optimisation Process (b) Final Path

Figure 4.2: RKHS motion planning (Marinho et al., 2016) in a precomputed cost field calculated according to (4.15). Obstacle potential costs and their spatial gradients are well-defined anywhere in the workspace. Dashed lines illustrate the obstacles boundaries. Star and pentagon mark the start and goal points, respectively. (a) paths generated during the RKHS optimisation process. (b) shows the optimal path.

is fitted. Figure 4.3 depicts a Hilbert continuous occupancy map generated for the environment shown is Fig 4.2. Using the continuous map representation, the occupancy and its gradient can be queried at any location.

Figure 4.4 shows an attempt to plan a path in an occupancy map using the functional gradient method described in (Marinho et al., 2016) with various support sizes ($N = 5, 50, 100, 200$). The cyan line depicts the optimal path after the algorithm has converged. Clearly, the resulting path is unsafe regardless of the size of the support $N$ used in the optimisation process. As expected, increasing the size of support leads to a more expressive path. Yet, even with $N = 200$ the resulting path was not safe. The reason lies in the lack of informative gradient in the occupancy map and the finite parametric resolution of the path representation. As the optimisation process deforms the curve, the spatial density of the support changes too. Consequently, there is less support around critical areas of the map such as the boundaries of obstacles. Increasing resolution even further will not alleviate this problem since we have no *a-priori* knowledge of the number of required support points. Additionally, increasing the number of support points create unnecessary jerks in the curve, as a response to noisy occupancy gradients, as shown in Fig 4.4.

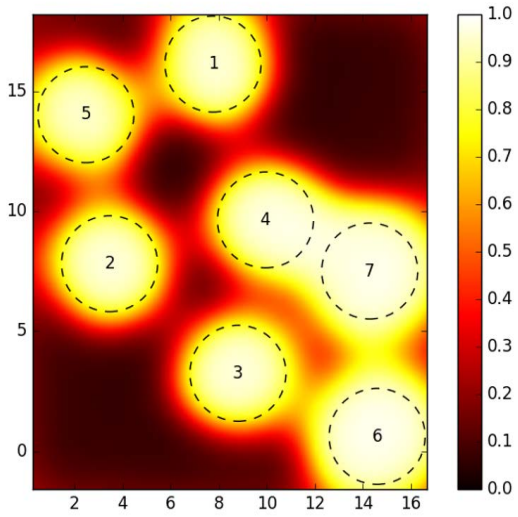
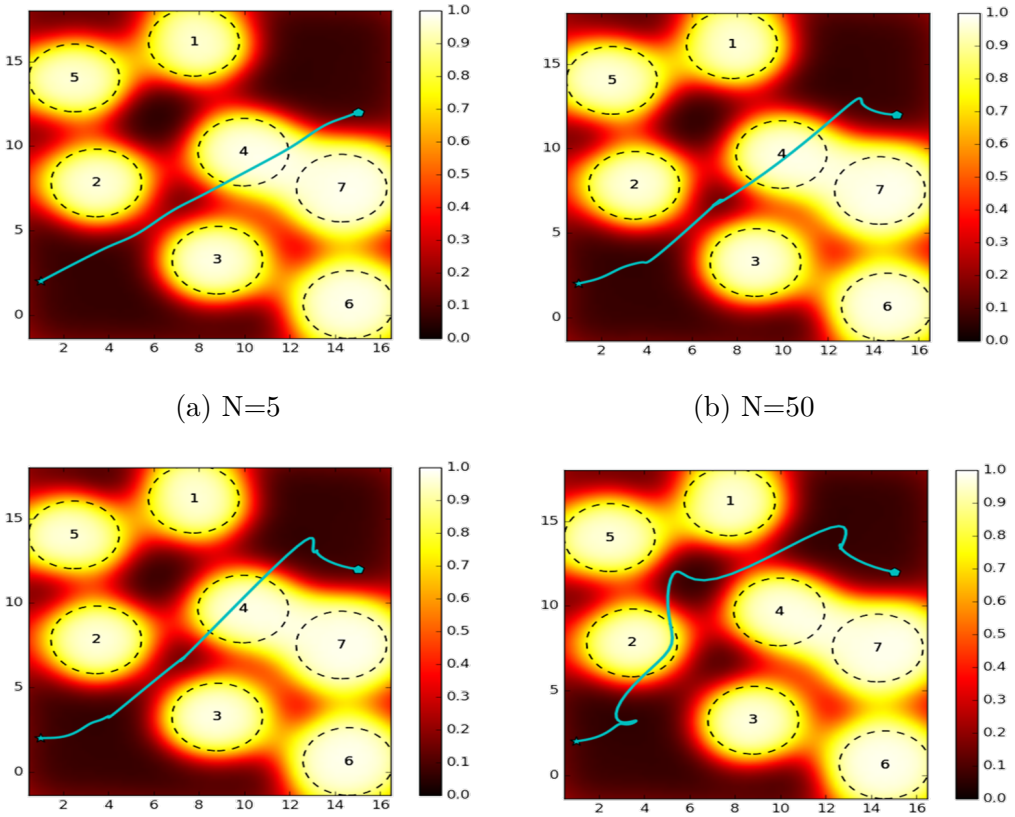



Figure 4.3: Continuous occupancy Hilbert map for the environment shown in Fig 4.2. The map shows the probability of occupancy, $p(y^* = +1|\mathbf{x}^*, \mathbf{w})$ as in (2.45). Note that the optimal path of in Fig. 4.2 passes through the gap between obstacles 4 and 7. In the Hilbert map, such a path is considered invalid as it passes occupied or unsafe area.

Our stochastic functional gradient method does not commit to a specific parametric resolution, rather it randomly selects points along the curve during the optimisation. Figure 4.5a shows in cyan the average path planned using the occupancy map of Fig 4.3. Since the optimisation objective balances an obstacle cost with a penalty on the trajectory shape, the optimal path is collision-free and smooth. Finding a path relies on generating enough samples to instantiate a gradient update, especially in areas of high importance such as obstacles boundaries. Figure 4.5b shows the convergence of the optimisation process to the minima of the objective, by plotting the maximum occupancy along the trajectory at each iteration. The maximum occupancy along the path drops steadily as the optimisation progresses. However, it cannot drop below 0.4, since the predictive occupancy between obstacles 2 and 4 is approximately 0.4. Yet, Fig. 4.5b provides empirical evidence for the expected optimality of the stochastic process.

Figure 4.6 provides a qualitative comparison between the stochastic GP planner and RRT* (Karaman, 2010) on the random worlds of Fig. 3.6, where Hilbert maps were generated from simulated laser data. Table 4.1 lists complementary information about the paths' length and safety. At first glance, the difference between the two planning methods is the path smoothness. The waypoints-based path produced by the RRT* planner, is jagged by construction, which is in contrast to the smooth path of the proposed method. Closer inspection reveals that the RRT* planner tends to place the path close to the obstacles' edge. This




(a) N=5 (b) N=50

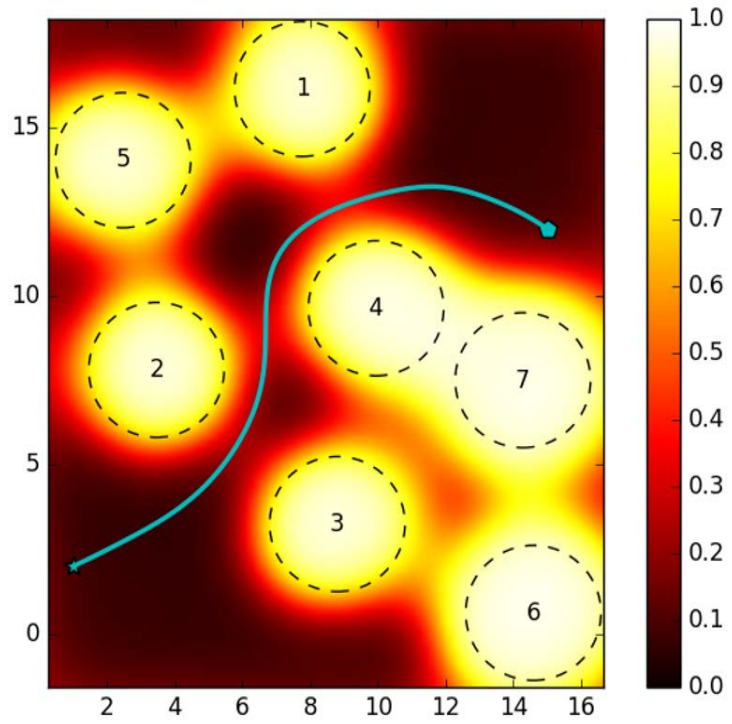
(c) N=100 (d) N=200

Figure 4.4: RKHS motion planning (Marinho et al., 2016) fails to plan using Hilbert maps. The individual plots compare the effect the size of the path support $N$ has. The final path is depicted in cyan, while the cyan star and pentagon mark the start and goal points, respectively. The RKHS motion planner does not find a valid solution since it represents the path with a parametric resolution, which leads to gaps in the sampling of the objective functional.

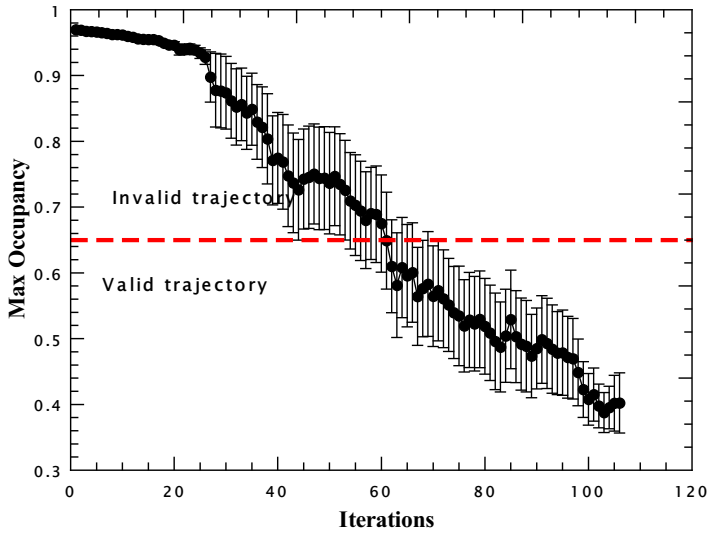
is a result of the search heuristic used in selecting the waypoint of the final RRT*, which does not consider proximity to obstacles. The stochastic GP planner, on the other hand steers path away from obstacles, maintaining a safe distance. Consequently, the path safety, as shown in Table 4.1, is borderline or even unsafe for RRT*, compared with a large safety margin for the GP planner. Moreover, although the GP planner safety margin is bigger, most resulting paths are slightly shorter than the paths produced by RRT*.

A major disadvantage of RRT* is evident from the paths generated for worlds 4 and 8. In these scenarios, the shortest path must traverse a narrow passage, where the probability of sampling a node for the RRT* is low. With no samples in the narrow corridor, the planned path takes a detour around the obstacles. resulting in significantly longer trajectories.

Figure 4.7 reveals the main drawback of the proposed method. It plots the




(a) Average optimal path




(b) Convergence

Figure 4.5: Stochastic functional gradient motion planner results. (a) Average path over 10 repetitions. The trajectory is depicted in cyan while the star and pentagon marker indicate the start and goal points, respectively. The average path follows the mid line between obstacles, which reduces the obstacle cost. (b) Shows the convergence of the stochastic functional gradient motion planner. The maximum occupancy along the trajectory is plotted as a function of the iterations. Data shown is the average over 10 repetitions. The $P = 0.5$ dashed red line marks the threshold for a valid trajectory. Note that the maximum occupancy does not reduce to zero, as the continuous occupancy map predictions for the gap between obstacles 1,2 and 4 are approximately 0.4.

Table 4.1: Comparison of path length and safety time between the stochastic GP planner and an RRT* planner. Path safety is measured by the maximum occupancy along the entire path, where unsafe path are marked in red..

| Algorithm | Stochastic GP Planner | | RRT* | |
|---|---|---|---|---|
| | Length [m] | Max. Occ. | Length [m] | Max. Occ. |
| World 1 | 39.1 | 0.26 | 39.4 | 0.32 |
| World 2 | 27.9 | 0.25 | 27.1 | 0.50 |
| World 3 | 37.0 | 0.33 | 37.1 | 0.47 |
| World 4 | 28.1 | 0.38 | 44.0 | 0.50 |
| World 5 | 31.2 | 0.28 | 31.5 | 0.49 |
| World 6 | 27.7 | 0.36 | 27.2 | 0.49 |
| World 7 | 36.0 | 0.38 | 39.0 | 0.50 |
| World 8 | 36.2 | 0.37 | 63.5 | 0.50 |
| World 9 | 35.4 | 0.28 | 35.4 | 0.49 |
| World 10 | 36.4 | 0.43 | 36.1 | 0.49 |

planning runtime as a function of the number of samples. While the stochastic GP planner requires approximately an order of magnitude less samples in order to optimise path selection, its runtime is significantly higher. The cubic dependence on the number of samples points out that the computational bottleneck of the stochastic planning paradigm is the GP path representation.

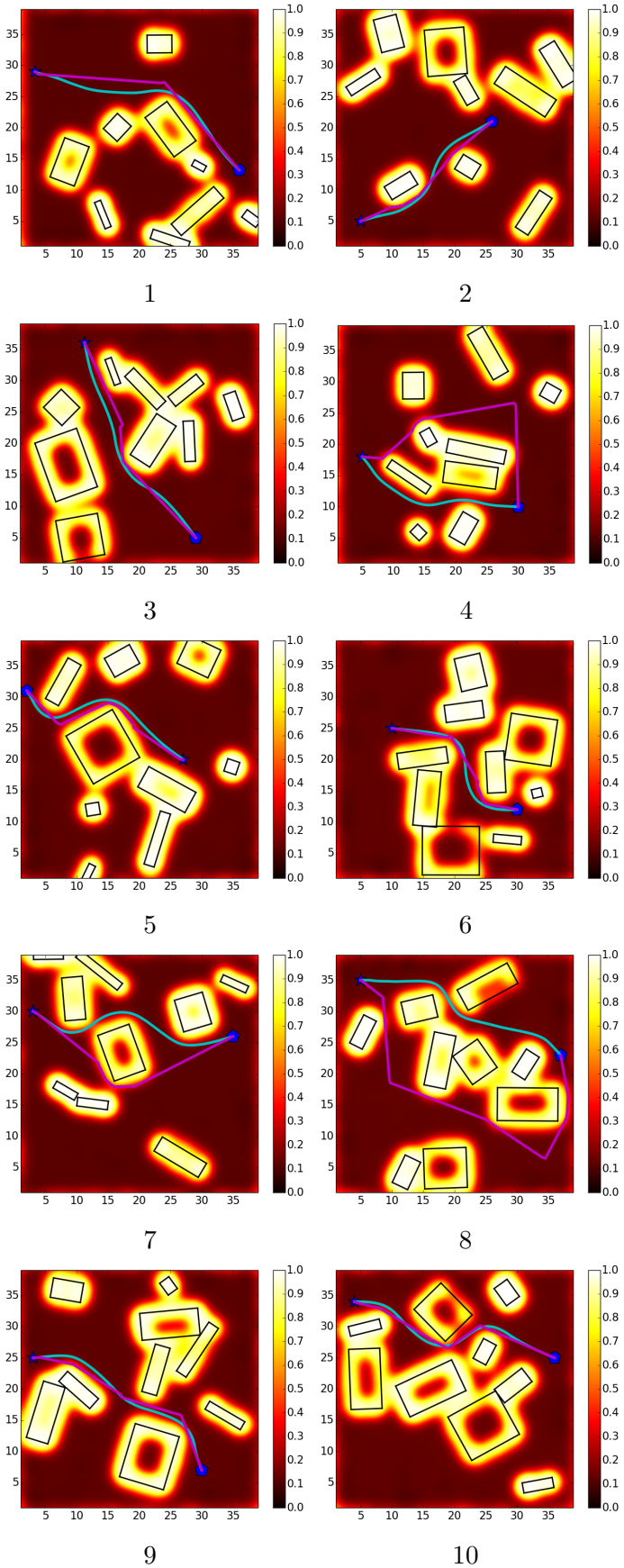



Figure 4.6: Comparison of path planning methods on a continuous occupancy map of randomly generated scenarios (refer to Fig. 3.6). The resulting paths for the proposed method and RRT* are in cyan and magenta, respectively.

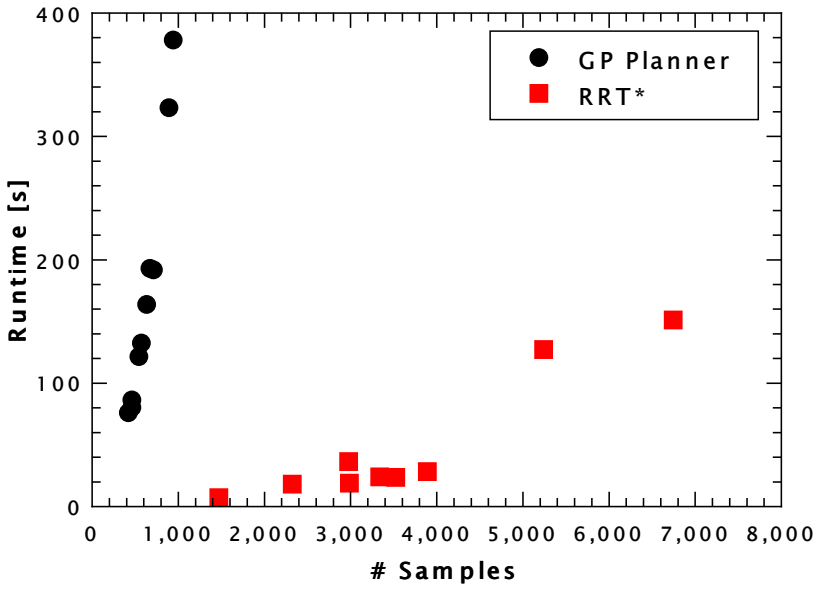
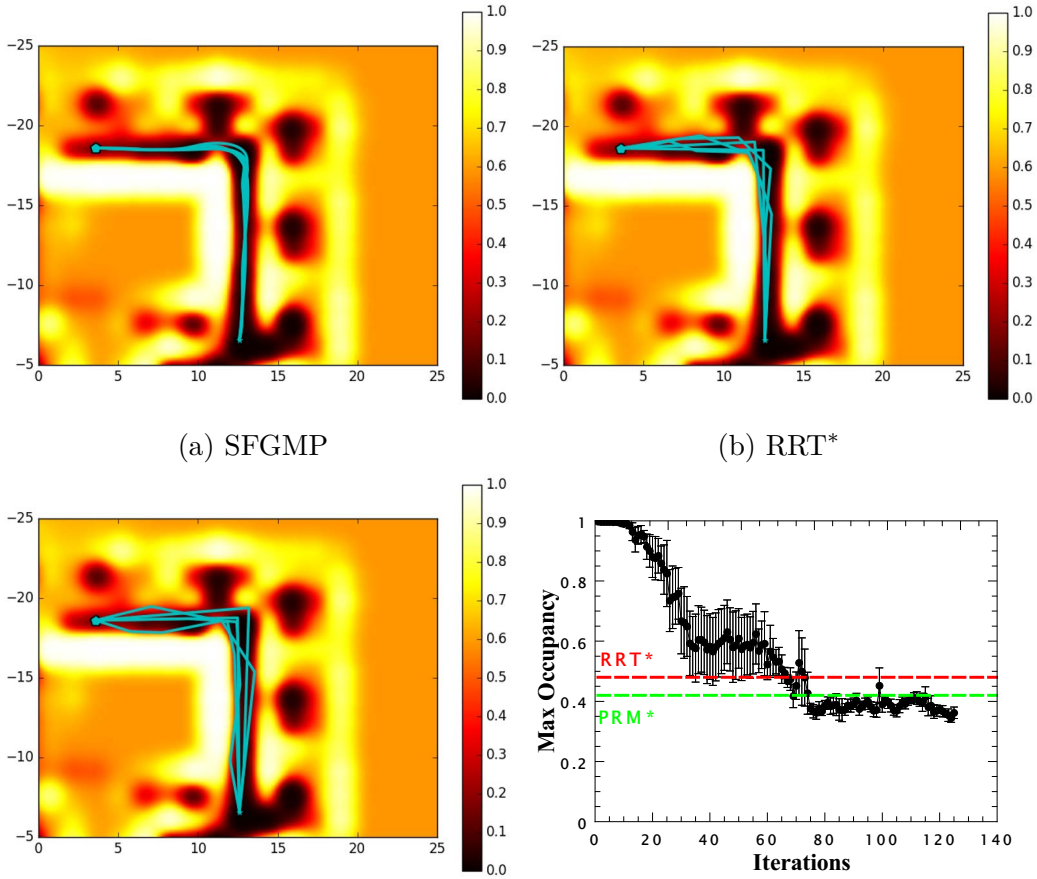



Figure 4.7: Runtime comparison between the proposed method and RRT*. RRT*, shown in red, requires more occupancy samples, however it keeps a simpler path model. The stochastic path planner, in black, uses a GP path representation. Although it require less samples to represent a path, the underlying GP model dictates a cubic computational complexity.

### 4.5.2 Real Laser-scan Data

The map for this experiment was generated using the Intel-Lab dataset (available at http://radish.sourceforge.net/). This map contains many rooms and dead-ends that might challenge the optimiser. We compared the optimal trajectory of our proposed method with two other standard planning methods; RRT* (Karaman, 2010) and PRM* (Karaman and Frazzoli, 2011) using implementations from the *Open Motion Planning Library* (OMPL) (Sucan et al., 2012). Figure 4.8 shows a comparison between the different methods. Both RRT* and PRM*, using a planning budget of 60s, succeed in generating a path from start to goal. The path consists of waypoints (states) the robot should pass through. The list of waypoints provides a very sparse representation of the path that requires additional resources in order to transform into robot actions. In contrast, the proposed stochastic functional motion planner provides a detailed and smooth path represented by a function of $t$. Furthermore, the paths generated by RRT* and PRM* might follow close to walls or overshoot the corner leading to a higher risk of collision, and longer paths. The paths of our stochastic planner tend to follow the mid line between obstacles and perform smooth turns resulting in shorter and safer trajectories.

Table 4.2 and Fig. 4.8d provide a quantitative comparison between our stochastic functional gradient motion planner (SFGMP), RRT* and PRM*. The objective of the optimisation is to minimise the obstacle cost. Figure 4.8d shows the reduction of the maximum occupancy along trajectory as the optimisation progresses. After converging to the optimal solution, the safety of the path of the proposed method is significantly better than that of the other two methods. The results in Table 4.2 summarise the expected performance. The maximum occupancy along the path, which indicates the path safety, is 0.36 for the proposed method and 0.42 and 0.48 for PRM* and RRT*, respectively. Furthermore, the penalty term $\mathcal{U}_{dyn}$ in the objective in Eq. (4.10) leads the optimisation to prefer shorter paths. We note that PRM* and RRT* employ a huerstic that is aimed at finding a shortest path on a generated graph. However, this often results in a path that cuts corners, potentially through unsafe space. Consequently, the path generated by our method outperforms the other sampling-based methods.




(a) SFGMP (b) RRT*

(c) PRM* (d) Convergence

Figure 4.8: Comparison of path planning methods on a continuous occupancy map of Intel-Lab; (a) stochastic functional gradient motion planner (SFGMP), (b) RRT* and (c) PRM*. Each image shows five repetitions of path planning with each method. SFGMP paths are smooth and follow the mid lines between walls. RRT* and PRM*, using a planning budget of 60s, produce paths that move the robot dangerously close to walls at times. (d) shows convergence of the stochastic functional gradient motion planner. The maximum occupancy along the trajectory is plotted as a function of the iteration. Data shown is the average over 5 repetitions. Red and green dashed lines mark the average performance of RRT* and PRM* respectively. Our proposed stochastic planner significantly outperforms the other methods. Note that the maximum occupancy does not reduce to zero, as the continuous occupancy map predictions for the end point is approximately 0.35.

Table 4.2: Performance comparison

| | SFGMP | RRT*[1] | PRM*[2] |
|---|---|---|---|
| Maximum occupancy | $0.36 \pm 0.02$ | $0.44 \pm 0.03$ | $0.46 \pm 0.03$ |
| Path length [m] | $20.90 \pm 0.10$ | $21.50 \pm 0.10$ | $22.50 \pm 0.50$ |

[1] Karaman (2010)
[2] Karaman and Frazzoli (2011)

## 4.6 Summary

This chapter introduced a novel method for path optimisation using occupancy maps. Sampling-based techniques are the prevalent method for path planning using occupancy maps. Although these techniques are flexible and have a high success rate in finding safe paths, optimising additional properties of the path such as length and execution time are not part of their reasoning. Trajectory optimisers, on the other hand, are designed for that purpose. Yet, the current implementations of trajectory optimisers require a finite resolution in the trajectory support and rely on having access to a well defined cost potential field. However, neither of these requirements are met by occupancy maps. Gradients obtained from the map's occupancy are noisy and not necessarily informative, which limits the choice of trajectory support, especially when the resolution is finite.

The planning method used in this chapter employs stochastic optimisation to enhance the expressiveness of the basic functional gradient motion planner. It removes the need to commit to an a-priori parametric resolution, which allows our planner to better handle obstacles. The GP paths used in the planner provide a structured and flexible representation that can easily incorporate prior knowledge or initial solutions, such as coarse paths generated by a sampling based method.

The main limitation of the stochastic algorithm presented in this chapter is its computational complexity. The main culprit is the GP path representation, which carries a cubic update cost and a quadratic query cost. These costs limit performance once the number of samples reach several hundreds. To tackle this limitation we present in the following chapter, a modification of the stochastic planning algorithm based on kernel approximations. We also introduce an adaptive sampling scheme that targets under-sampled areas of the curve and biasing the stochastic sampling.

# Chapter 5

# Stochastic Scalable Path Planning

## 5.1 Introduction

In the previous chapter, we presented a novel trajectory optimisation method for planning in occupancy maps. Trajectory optimisers use a variational planning paradigm, which enables optimisation of a variety of objective functions, such as safety or control cost, directly in the space of trajectories. However, aside from the method described in Chapter 4 and in (Francis et al., 2017), there are no implementations of trajectory optimisation for occupancy maps. The main impediment lies in the optimiser's assumption the objective cost and gradient can be computed anywhere. Such an assumption is generally not valid in occupancy maps, as the map might have gaps or non-informative gradients. Consequently, there are no guarantees for an optimal or even safe solution.

In this chapter, we present a novel approach for trajectory optimisation using occupancy maps. We utilise kernel approximation techniques to form an expressive and tractable non-linear path model. This representation can be considered as a generalisation of the motion planning in *reproducing kernel Hilbert spaces* (RKHSs) paradigm of Marinho et al. (2016). While (Marinho et al., 2016) implicitly employed a reduced rank approximation of the kernel matrix (Schölkopf et al., 1998), the proposed approach can utilise any kernel matrix approximation technique. Moreover, the approximate kernel representation is naturally updated by randomly drawn samples, which replace the predetermined sampling schedule of (Marinho et al., 2016; Mukadam et al., 2016). Consequently, optimisation can employ *Stochastic Gradient Descent* (SGD) (Bottou, 2010) with its convergence guarantees. Finally, with a finite path representation, updating and querying the path model has a fixed cost, which alleviates the main computational restriction of the non-parametric representation discussed in Chapter 4.

The technical contributions of this chapter are:

1. An expressive and tractable path model based on kernel approximations,

Part of this chapter has been presented in the Third Machine Learning in Planning and Control of Robot Motion Workshop, *IEEE International Conference on Robotics and Automation*, 2018, and will also appear in *IEEE International Conference on Robotics and Automation*, 2019.

which can be considered a generalisation of the Gaussian Process Motion Planner (Mukadam et al., 2016). This is a critical building block of the path planner, as it is allows for fast and low computational cost update procedures using stochastic samples.

2. A path planning method based on SGD (Bottou, 2010) which ensures efficient convergence to an optimal solution under the guarantees of SGD.

3. Introducing an adaptive stochastic sampler to accelerate functional updates. The sampler uses only the generated samples to reconstruct the objective costs and adapts its proposal distribution to target areas of the curve that require more samples. The same sampler also provides a probabilistic indicator for convergence. This sample-based indicator replaces the exhaustive convergence checks done by other FGD planners.

The remainder of this chapter is organised as follows. We rely on the general method developed in Chapter 4 and introduce the changes required to form a scalable functional path planner and the adaptive stochastic sampler in Section 5.2. Experimental results obtained in simulation and with real data scenario are shown in Section 5.3. Finally, Section 5.4 summarises the proposed method and contributions.

## 5.2 Scalable Functional Regression

In this section, we extend the FGD path planning paradigm, developed in Chapter 4, into a scalable stochastic trajectory optimisation framework based on kernel approximations. Similar to Chapter 4, the main objective of the planner is to produce a safe, collision-free path. A secondary objective may incorporate other costs such as smoothness of the trajectory or time of travel. The notation used in this chapter follows the notation introduced in Section 4.3.

### 5.2.1 Stochastic Functional Regression

Any FGD planner optimises an objective function, such as in Eq. (4.1). As the objective is uncountable, it is estimated via samples. Therefore, the choice of sampling schedule is paramount for a successful and efficient planner. The importance of the sampling schedule is exacerbated in occupancy maps, where not every sample can generate an informative gradient, as discussed in Section 4.4. Consequently, sampling everywhere along the curve is most desired, as this increases the chance of identifying transition areas in the map. Yet, with a fixed resolution sampling defining a sufficient resolution a-priori is difficult. Hence most methods limit the sampling resolution according to their computational resources.

GP-based planners (Mukadam et al., 2016; Dong et al., 2016) use GPs for a smooth path representation. However, as the path is updated only at fixed support points, a dense representation is needed to ensure sufficient expressivity. Similar limitations also hold for the non-parametric approach used in (Marinho et al., 2016), where the support is taken from fixed resolution samples of the objective function. CHOMP and STOMP perform batch optimisation by exploring the solution space using either Hamiltonian Monte Carlo or by estimating the probability density of the objective using noisy path perturbations. As the path is waypoint based the solution space exploration is performed in the robot's workspace. Consequently, the optimisation process is highly sensitive to the choice of the exploration hyperparameters. For example, STOMP's update rule fails if the variance of perturbation is smaller than the size of obstacles, which in occupancy maps is unknown a-priori. The stochastic non-parametric approach of Chapter 4 addresses this problem by using continuous sampling in the trajectory domain. However, as the path is represented by a GP the computational costs are high, i.e. of the order $\mathcal{O}(N^3)$ where N is the number of samples.

The approach taken in this work, alleviates the limitations present in the previous chapter. Namely, it allows stochastic updates from continuous samples. To keep the computational cost low, while maintaining a highly expressive representation, a parametric and thus concise path representation based on kernel approximation is employed.

In kernel machines, $\Upsilon(t)$ defines a mapping from $t \in [0, 1]$ into a potentially infinite-dimensional RKHS [1] $\mathcal{H}$ (Schölkopf and Smola, 2001). The kernel function $k(t, t')$ defines the inner product $\left\langle \cdot, \cdot \right\rangle_{\mathcal{H}}$ between two points in that space. In the approximate kernel approach we denote $\hat{\Upsilon}$ as a finite set of features that approximate the RKHS inner product by a dot product;

$$k(t, t') = \left\langle \Upsilon(t), \Upsilon(t') \right\rangle_{\mathcal{H}} \approx \hat{\Upsilon}(t)^T \cdot \hat{\Upsilon}(t'). \tag{5.1}$$

We note that the set of features only approximates the selected kernel in expectation, indicated by $\hat{\Upsilon}$ notation. There are several methods to generate features to approximate a kernel. For the *radial basis function* (RBF) kernel defined by $k(t, t') = \exp(-\gamma \parallel t - t' \parallel^2)$, with $\gamma$ a free parameter and $\parallel \cdot \parallel$ is the Euclidean norm, we employed two different approximations[2]:

1. *Random Fourier features* (RFF) (Rahimi and Recht, 2009)

[1] The path RKHS is different to the one used by the Hilbert maps.
[2] Other approximations such as (Melkumyan and Ramos, 2009) were not used in this work

This approximation requires $m$ random samples of two variables;

$$\begin{aligned} s_i &\sim \mathcal{N}(0, 2\gamma I) \\ b_i &\sim \text{uniform}[-\pi, \pi] \end{aligned} \qquad i = 1...m \tag{5.2}$$

The features vector is then given by

$$\hat{\Upsilon}^{RFF}(\mathbf{t}) = \frac{1}{\sqrt{m}}[\cos(s_1\mathbf{t} + b_1), ..., \cos(s_m\mathbf{t} + b_m)] \tag{5.3}$$

2. RBF features

   A kernel matrix $K$ with rank $n$ can be approximated by projecting it into a lower rank matrix using a set of $m$ inducing points denoted by $\hat{t}_1, ..., \hat{t}_m$ (Schölkopf et al., 1998). Then, $K \approx K_{n,m}\hat{K}_m^{\dagger}K_{m,n}$, where the elements of matrices $K_{n,m}$ and $\hat{K}_m$ are defined as:

$$\begin{aligned} (K_{n,m})_{(i,j)} &= K(t_i, \hat{t}_j)\ i = 1, ..., n \text{ and } j = 1, ..., m, \\ (\hat{K}_m)_{(i,j)} &= K(\hat{t}_i, \hat{t}_j)\ i, j = 1, ..., m. \end{aligned}$$

   $\hat{K}_m^{\dagger}$ is the pseudo inverse of $\hat{K}_m$. Using these $m$ inducing points, the approximation features vector is given by Schölkopf et al. (1998):

$$\hat{\Upsilon}^{RBF}(t) = \hat{D}^{1/2}\hat{V}^T[k(t, \hat{t}_1), ..., k(t, \hat{t}_m))]^T \tag{5.4}$$

   Here, $\hat{D}$ is the diagonal matrix of eigenvalues of $\hat{K}_m$ and $\hat{V}$ are the corresponding eigenvectors. The $m$ inducing points can be modified during the optimisation, similar to (Williams and Seeger, 2001). However, in the this work we employed evenly distributed fixed features. We note that since the planner of Marinho et al. (2016) also uses a fixed number of support points, it implicitly employed this approximation, though without stochastic updates.

Using a weight vector $\mathbf{w}$ we can now express the robot configuration $\xi(t)$ at $t$, as a function of the finite set of approximating features, $\hat{\Upsilon}(t)$:

$$\xi(t; \mathbf{w}) = \xi_o(t) + \xi_b(t) + \mathbf{w}^T\hat{\Upsilon}(t). \tag{5.5}$$

Here, $\xi_o$ is an offset path, which may be used to bias solution and can be computed by a crude and fast planner. $\xi_b$ is a term used to enforce boundary conditions. Both $\xi_o$ and $\xi_b$ are represented by an approximated kernel representation with the same curve properties as $\xi$ (continuity, differentiability, etc.), although the feature set can be different.

Once the path representation has been defined, we can treat path planning as a regression problem, i.e., optimising the weight vector $\mathbf{w}$:

$$\mathbf{w}_{optimal} = \underset{\mathbf{w}}{\operatorname{argmin}} \quad \mathcal{U}(\mathbf{w}). \tag{5.6}$$

The advantage of using this approach is that the model can be learned through stochastic sequential updates by samples from the entire domain.

As with any kernel method, the choice of $\gamma$ and $m$ is critical. The inverse lengthscale parameter $\gamma$ depends on the desired smoothness of the trajectory. Non-smooth paths or sharp transitions require a shorter lengthscale (larger $\gamma$). Alternatively, a different kernel function, e.g. Matérn, and an appropriate kernel approximation should be used. Choosing the number of features $m$ is a trade-off between the approximation accuracy and computational complexity. With a larger features set, the trajectory model can be more expressive, however the model will require additional computational resources.

In the following sections we discuss how to implement FGD using the approximated kernel regression model. We revise the general update rule of Eq. (2.37) into a practical gradient update based on the choice of path representation. Then, we present the full algorithm of the stochastic approximate kernel path planner.

### 5.2.2 Approximate Kernel Update Rule

Equation (5.5) expresses the path as a weighted sum of features, therefore the iterative update rule of Eq. (2.37) must be performed with respect to the weight vector $\mathbf{w}$. Following Eq. (2.37), we sample the functional gradient of the objective function at time $t_i$. We refer to these samples as stochastic, since $t_i$ can be drawn at random from anywhere along the curve domain, $t_i \in [0, 1]$, and is not limited by a predefined sampling resolution. The sampled gradient $\nabla_\xi \mathcal{U}\Big(\xi_n(t_i, \mathbf{w}_n)\Big) \in \mathcal{H}$ can be viewed as a path perturbation, which is defined in $\mathcal{H}$ and thus must be projected onto the finite representation spanned by $\mathbf{w}$ using the appropriate inner product, which is approximated using Eq. (5.1). This approximation leads to the following iterative update rule:

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \eta_n A^{-1} \hat{\Upsilon}(t_i)^T \hat{\Upsilon}(t_i) \nabla_\xi \mathcal{U}\Big(\xi_n(t_i, \mathbf{w}_n)\Big). \tag{5.7}$$

$\xi_o$ and $\xi_b$ are removed from Eq. (5.7) for brevity. Note that given a convex problem, to guarantee convergence of SGD, the learning rate $\eta_n$ must satisfy the Robbins-Monro conditions (Robbins and Monro, 1951); $\sum_{n=1} \eta_n^2 < \infty$ and $\sum_{n=1} \eta_n = \infty$.

Boundary conditions are handled in a similar fashion. We employ an additional

path $\xi_b = \mathbf{w}_b^T \hat{\Upsilon}_b$ to compensate for the difference around the boundary conditions. The boundary features $\hat{\Upsilon}_b$ are not necessarily identical to $\hat{\Upsilon}$. The update rule for $\mathbf{w}_b$ then has the following form:

$$\mathbf{w}_{bn+1} = \mathbf{w}_{bn} - A^{-1}\hat{\Upsilon}_b(t_b)^T \hat{\Upsilon}_b(t_b)\Delta x_b(t_b). \tag{5.8}$$

Here $t_b$ are time points where boundary conditions are defined and $\Delta x_b(t_b)$ is the corresponding difference between the current value of $\xi$ at $t_b$ and the desired value at the boundary. Note that this is similar to Eq. (5.7), except $\eta_n$ was omitted and the gradient was replaced by the difference to the desired boundary value.

### 5.2.3 Targeted Sampling

The trajectory optimisation process requires valid functional gradient samples. It is clear, as discussed in Chapter 4, that only stochastic sampling schedules can guarantee optimisation convergence. However, in an occupancy map, these samples are only present around the borders of obstacles. Moreover, as the optimiser deforms the path, the location of these samples changes too. In order to speed-up convergence, we introduce a dynamic proposal distribution $\mathcal{Q}$, that proposes samples based on their expected effectiveness. $\mathcal{Q}$ replaces the uniform distribution used in Chapter 4. Such a sampling schedule increase the chance of sampling around areas where previous updates were effective while still maintaining samples over the entire path domain $t \in [0, 1]$. We note that $\mathcal{Q}$ serves a similar purpose to *importance sampling* (Bishop, 2006), however in the functional optimisation framework, $\mathcal{Q}$ must dynamically adapt to changes in the functional.

An effective $\mathcal{Q}$ needs to be proportional to the objective functional gradient, i.e. higher probability where there is gradient to deform the path. In addition, sampling and updating $\mathcal{Q}$ should not incur high computational costs. These two properties are achieved by representing $\mathcal{Q}$ as a mixture model:

$$\mathcal{Q}(t) = \sum_{l \in L} p(l)p(t|l), \tag{5.9}$$

where $L$ is a set of intervals $l \in L$ such that $\cup_{l \in L} l = [0, 1]$. $p(t|l)$ is a uniform distribution defined over the interval $l$, i.e., $p(t|l) = U[l]; \forall l \in L$ . $p(l)$ is the probability of selecting interval $l$, and we require $\sum_{l \in L} p(l) = 1$.

Generating samples from $\mathcal{Q}$ is a two-step process, as shown in algorithm 7. First, $N$ interval samples are drawn according to $p(l)$. Each sample $s_l$ corresponds to an interval $l \in L$. Given a sampled interval $s_l$, a corresponding time sample is drawn according to $p(t|s_l)$. In this work, $p(t|s_l)$ is a uniform distribution in interval $s_l$, however more elaborate distribution can be used.

**Algorithm 7:** Sampling from proposal distribution $\mathcal{Q}$

**Input:** $\mathcal{Q}(t) = \sum_{l \in L} p(l)p(t|l)$: proposal distribution.
$N$: number of samples required.
**Output:** $\mathcal{T}$: samples in range $[0, 1]$

$\mathcal{T} \leftarrow \varnothing$
**for** $i = 1 : N$ **do**
  //Sample and interval from $p(l)$:
  $s_l \sim p(l)$
  //Draw time samples, $t_s$ from interval $s_l$:
  $t_s \sim p(t|s_l)$
  $\mathcal{T} \leftarrow \mathcal{T} \cup t_s$
**end**

An effective proposal distribution draws samples in high-impact areas, which corresponds to higher objective functional gradients $\nabla\mathcal{U}\Big(\xi(t)\Big); \forall t \in [0, 1]$. Moreover, as the path deforms during optimisation, $\mathcal{U}\Big(\xi(t)\Big)$ and its gradients changes, which requires $\mathcal{Q}$ to dynamically follow. However, since our knowledge of $\nabla\mathcal{U}\Big(\xi(t)\Big)$ is based on randomly drawn samples, we are also interested in sampling over the entire $[0, 1]$ domain. Therefore updating $\mathcal{Q}$ must balance the exploration-exploitation trade-off. This approach will guarantee that intervals with higher objective content will be sampled more, while still sufficiently sampling all over the $[0, 1]$ domain.

Practically, updating $\mathcal{Q}$ depends on the choice of $p(l)$. To maintain efficient computation, we employ a multinomial distribution $p(l) \sim Mult(|L|)$, where $|L|$ is the number of intervals in $L$. Algorithm 8 details the update process. Since $\mathcal{Q}$ has to dynamically follow $\nabla\mathcal{U}\Big(\xi(t^*)\Big)$, the update process collects samples $S = \nabla\mathcal{U}(t_1), ..., \nabla\mathcal{U}(t_N)$ in $|L|$ *first in first out* (FIFO) queues. Based on the time $t_i$ of a sample, the relevant queue is updated with $\nabla\mathcal{U}(t_i)$. As the queue has a finite depth $D_Q$, new samples push older samples out. Once the queues has been updated, $p(l)$ is derived of the normalised sum of queue weights. The choice of queue depth $D_Q$ defines the reactiveness of the update process. A high $D_Q$ means $p(l)$ will responds slowly to new samples. On the other hand, $D_Q = 1$ will only maintain the last samples from each interval.

As $\mathcal{Q}$ changes during optimisation, it can be used as an indicator of convergence. $\mathcal{Q}$ biases sampling according to the impact a sample had on the path update. Meaning, that around $\xi_{optimal}$, new samples should have little effect on the path. As a result, $\mathcal{Q}$ should return to maximum entropy sampling, i.e. $U[0, 1]$. Hence, by monitoring the entropy of $\mathcal{Q}$, one can identify whether sampling follows a uniform distribution which indicates a solution has been found. We note that maximum entropy is not a sufficient condition for convergence as it does not

**Algorithm 8:** Updating the proposal distribution $\mathcal{Q}$

**Input:** $S = \mathcal{U}(t_1), ..., \mathcal{U}(t_N)$: $N$ Samples of the objective functional.
$L$: intervals.
$D_Q$: depth of queue.
**Output:** $p(l) \forall l \in L$: samples in range $[0, 1]$

//First update instance: initialise queue $Qu(l) \forall l \in L$:
**if** *not* $Qu$ **then**
  **foreach** $l \in L$ **do**
    $Qu(l) \leftarrow Queue(D_Q)$
  **end**
**end**

//Update Queues:
**foreach** $(t_i, U_i) \in S$ **do**
  $l_i = l \in L | t_i \in l_i$ //Find interval that contains $t_i$.
  $Qu(l_i) \leftarrow \nabla U_i$ //push sample $\nabla U_i$ to queue.
**end**
//Computing $p(l)$:
**foreach** $l \in L$ **do**
  $w(l) = \sum_{D_Q} Qu(l)$
**end**
$p(l) = \frac{w(l)}{\sum_{l \in L} w(l)}$

guarantee safety. To ensure safe convergence, two condition should be met:

(i) All samples must be valid, i.e. no collision. To ensure safety, we require that this condition will hold over several iterations.

(ii) The entropy of sampler must be above a threshold which is proportional to the maximum attainable entropy of a uniform distribution.

### 5.2.4 Approximate Kernel Path Planning Algorithm

The pseudo-code of the stochastic approximate kernel path planner is shown in Algorithm (9). The output of this algorithm is an optimised path $\xi_{min}(\cdot)$, parametrised by the weight vector $\mathbf{w}_{min}$.

At each iteration, a mini-batch $(t_s, u_s)$ is drawn according to the proposal distribution $\mathcal{Q}$. The occupancy of each sample $t^* \in t_s$ and $u^* \in u_s$ is assessed by querying the map model in the corresponding state $\xi_n(t^*)$. If the probability of occupancy at $\xi_n(t^*)$, $P_{occ}$, is within the safety limits, i.e. clear of obstacles, a functional gradient update is invoked. Following Eq. (5.7), the weight vector $\mathbf{w}$ is updated with the stochastically sampled gradient observations, leading to a

**Algorithm 9:** Stochastic approximate kernel FGD path planner

**Input:** $\mathcal{H}$: Occupancy Map.
$\xi(0), \xi(1)$: Start and Goal states.
$P_{safe}$: No obstacle threshold.
$\Upsilon(\hat{t}', \cdot)$: Approximated feature vector.
$\mathcal{Q}$: Proposal distribution (optional).
**Output:** $\mathbf{w}_{min}, \xi_{min}(\cdot)$

//If $\mathcal{Q}$ is not provided, use uniform distribution as default
**if** *not* $\mathcal{Q}$ **then**
  $\mathcal{Q} = \mathcal{U}[0, 1]$
**end**

$n = 0$
**while** *solution not converged* **do**
  $(t_s, u_s) \sim \mathcal{Q} \leftarrow$ Draw mini-batch from $\mathcal{Q}$ - Algo. 7
  **foreach** $(t^*, u^*) \in (t_s, u_s)$ **do**
    $P_{occ} \leftarrow \mathcal{H}(x(\xi_n(t^*), u^*))$, Eq. 2.45
    **if** $P_{occ} \leq P_{Safe}$ **then**
      //Update rule Eq. 5.7
      $\mathbf{w}_{n+1} \leftarrow \mathbf{w}_n - \eta_n A^{-1} \hat{\Upsilon}(t_i)^T \hat{\Upsilon}(t_i) \nabla_\xi \mathcal{U}\Big(\xi_n(t_i, \mathbf{w}_n)\Big)$
    **end**
  **end**
  //Fix boundary conditions, Eq. (5.8):
  $\mathbf{w}_{bn+1} \leftarrow \mathbf{w}_{bn} - A^{-1} \hat{\Upsilon}_b(t_b)^T \hat{\Upsilon}_b(t_b) \Delta x_b(t_b)$
  Update $\mathcal{Q}$ - Algo. 8
  $n = n + 1$
**end**

new path representation $\xi_{n+1}(\cdot)$. Finally, the boundary conditions are enforced using Eq. (5.8).

This leads to an algorithms with low computational complexity which stems from the concise path representation and update rule. Using $m$ approximated features, the computational cost of updating and querying the path model is $\mathcal{O}(m)$ which is constant regardless of the number of samples drawn. This is in contrast to the computational cost of the stochastic GP path planner used in Chapter 4, which is cubic in the number of samples, i.e. $\mathcal{O}(N^3)$. Meaning that the time per iteration increases as more samples are collected.

## 5.3 Experiments

In this section, we evaluate the performance of our method and compare it with other related path planning techniques in simulation and with real data. We

show that stochastic sampling is a critical aspect of path planning in occupancy maps, which is complimented by the scalable model of the approximate kernel path representation. We compare our proposed method with two other planning methods for occupancy maps; RRT* (Karaman, 2010) and the stochastic GP path planner introduced in Chapter 4. We conclude this section by comparing the effectiveness of the targeted sampler $\mathcal{Q}$ with a uniform sampling approach. All methods, including Hilbert maps, are implemented in Python and tested on an Intel Core i5-4570 with 8GB RAM.

### 5.3.1 Simulations

Most trajectory optimisers assume full knowledge of obstacle properties and compute a cost function and its spatial gradient for the entire workspace, e.g. (Zucker et al., 2013). This is not attainable when working with occupancy grid maps.

Fig. 5.1 compares planning using cost maps and occupancy map using two leading methods, STOMP (Kalakrishnan et al., 2011) and an RKHS planner (Marinho et al., 2016). Occupancy is represented by a Hilbert map (Ramos and Ott, 2015), which was computed using simulated laser readings of the environment. While planning in a cost map both methods successfully find a safe path from start to goal. However, when the same algorithms are used with the occupancy map, they both fail. The reason lies in the deterministic sampling schedule both methods use, where the path is updated only around its predetermined support. As there are no valid gradients inside obstacles, gaps are formed in the support of both curves and the path can not be updated. As stated in Chapter 4, using TSDF to convert an OGM to an SDF will do little to alleviate this problem, since TSDF cannot produce valid gradients in unobserved parts of the map, as been also observed by Oleynikova et al. (2017).

The performance of our proposed stochastic planner differs from that of other planners. The RBF planner employed $m = 50$ inducing points with $\gamma = 7$, while the RFF planner used $m = 100$ with $\gamma = 10$. The trade-off parameter $\lambda = 0.0075$ and the per iteration learning rate $\eta_n = \eta_0^{-1}(n+n_0)$, where $\eta_0 = 50$ and $n_0 = 100$.

An overviews of the iterative process of the stochastic FGD path planner is shown in Fig. 5.2 where each column shows

(i) the current path overlaid on the occupancy map;

(ii) the accumulated samples, both valid and invalid;

(iii) the underlying cos which is used for presentation purposes only as it is available to the planner only through stochastic samples.

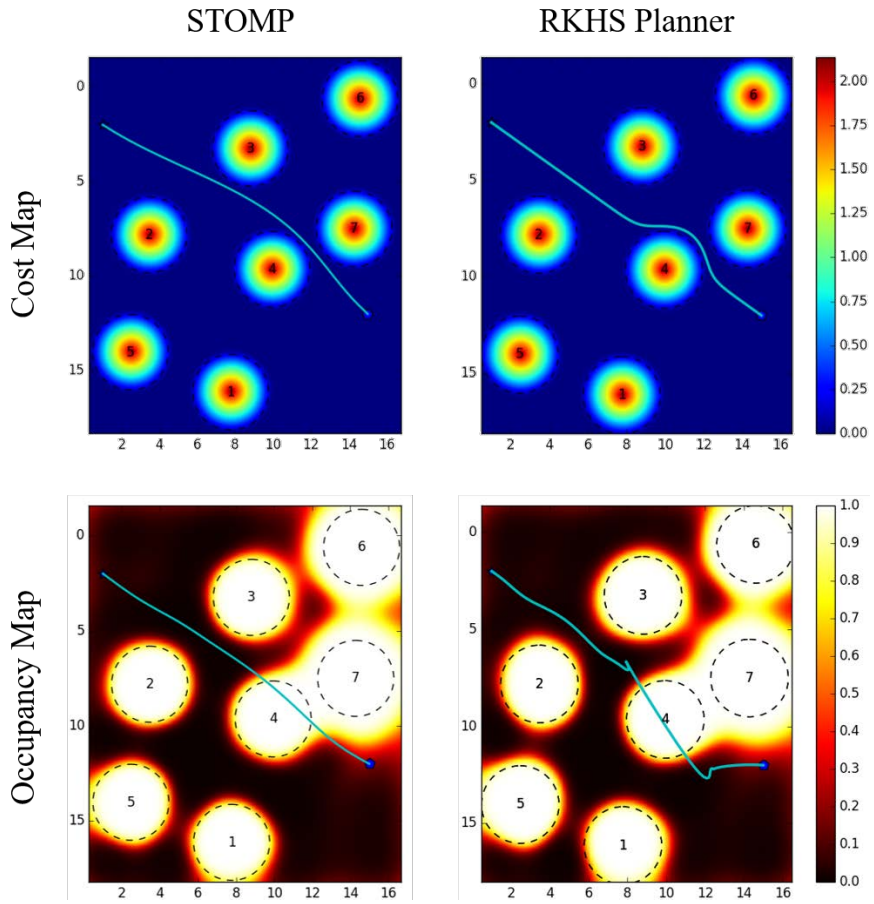



Figure 5.1: Comparison of motion planning in cost maps vs. occupancy maps using state-of-the-art planners. In all maps, dashed black lines indicate the borders of obstacles. Cost map is computed according to (Zucker et al., 2013). Hilbert maps (Ramos and Ott, 2015) are used to model occupancy, which is computed from simulated laser readings. Left, STOMP (Kalakrishnan et al., 2011) and on the right, RKHS non-parametric planner (Marinho et al., 2016). Both planners are successful when planning in a well-defined cost map. However, both fail when planning using occupancy maps.

The iterative update process starts from an initial guess $\xi_o$, a line connecting the start and goal points. The overall cost is determined solely by on the obstacle cost as the dynamic cost for a straight line is $\mathcal{U}_{dyn} = 0$. After 30 iterations the path deforms around the edges of the obstacles. Samples are drawn from the entire domain $[0, 1]$ and those inside obstacles are rejected. However, samples on the edges with valid occupancy update the path and push it away from the obstacles. At $n = 50$, the path clears all obstacles, however, the optimisation has not yet converged since opposing objective functions, motion and obstacles, have not equalised yet. After 59 iterations the algorithm has converged to its final solution. With a mini-batch of 20 samples per iteration about 1200 samples were used in order to reach convergence. Deterministic sampling methods such

Table 5.1: Simulation comparison

| | Max. occupancy | Path length [m] | Samples | Runtime [s] |
|---|---|---|---|---|
| Approx. Kernel: RBF | $0.40 \pm 0.02$ | $21.2 \pm 0.4$ | $1418 \pm 350$ | $10.4 \pm 2.6$ |
| Approx. Kernel: RFF | $0.40 \pm 0.02$ | $21.4 \pm 0.2$ | $915 \pm 212$ | $12.3 \pm 3.0$ |
| GP Paths[1] | $0.35 \pm 0.05$ | $20.5 \pm 0.2$ | $775 \pm 372$ | $315 \pm 293$ |
| RRT*[2] | $0.46 \pm 0.05$ | $19.6 \pm 0.3$ | $1397 \pm 219$ | $7.1 \pm 2.2$ |

[1] Introduced in Chapter 4
[2] Karaman (2010)

as (Marinho et al., 2016; Mukadam et al., 2016; Zucker et al., 2013) require dense sampling, in the order of 100s of samples per iteration, of the objective function to decide on the best update location. As such the stochastic path planner offers significant reduction in computational costs.

A quantitative comparison between the different planners is shown in Table 5.1. The RRT* planner produces non-smooth piecewise-linear paths, which are inherently shorter than the other method. However, as it does not minimise the collision risk, the resulting paths might be unsafe as the maximum occupancy exceeds the safety threshold. As expected, its runtime is shorter than both methods, though comparable to the approximate kernel method. The GP planner requires half the iterations of the approximate kernel for convergence. However, its runtime is an order of magnitude higher due to the cubic complexity with the number of samples. The proposed method's complexity, on the other hand, is fixed regardless of the number of samples observed, which leads to a runtime that depends mainly on the sampling cost, similar to RRT. The small difference in runtime between RBF and RFF features, mainly depends on the number of approximating features used.

Being a member of the functional gradient planners family (Zucker et al., 2013), our method is able to plane trajectories in high-dimensional environments. Figure 5.3 shows a planning instance in a 3D occupancy map, where the planner finds a safe path through an opening in a wall. The gray surface represents the 0.5 occupancy iso-surface derived from the volumetric 3D map.

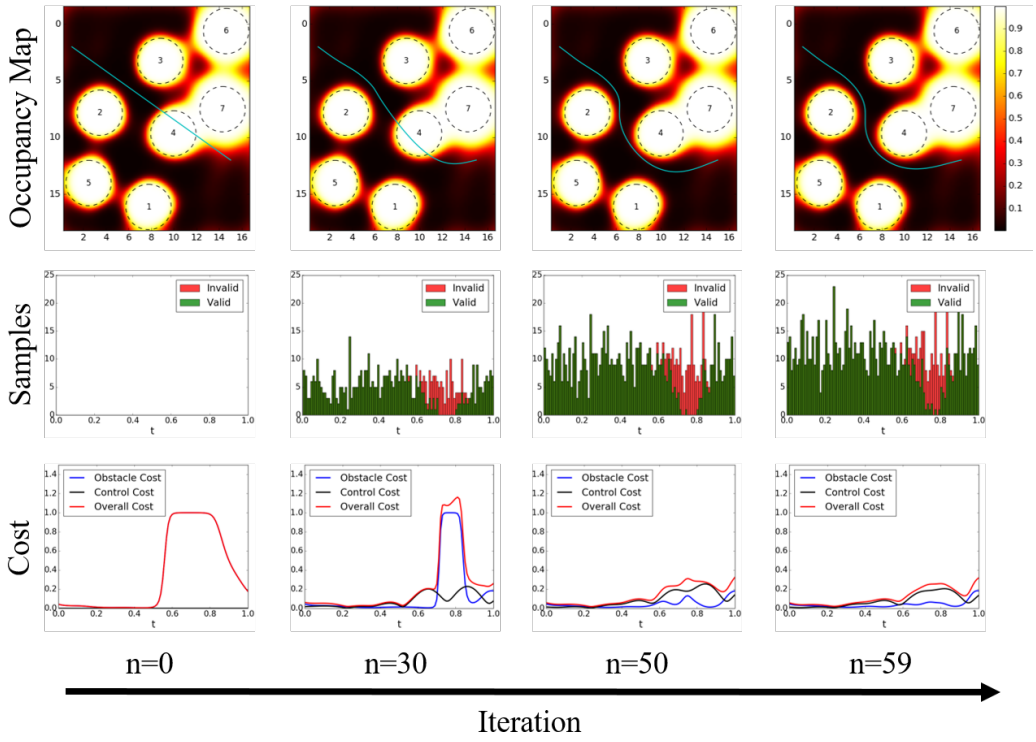
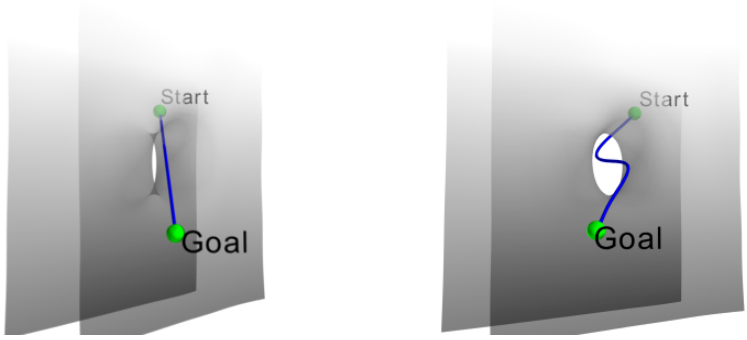



Figure 5.2: The image is divided into four segments, depicting the state of our planner at different iterations $n = 0, 30, 50, 59$. Each segment's column consists of three images: (top) shows the current path superimposed on the occupancy map, (middle) shows the accumulated samples over all previous iterations, where green and red indicate valid and invalid samples respectively, (bottom) depicts the objective functions for obstacle, motion-related and the overall cost. This view of the cost is only for presentation purposes and is not available to the planner, as it relies only on stochastic samples. $n = 0$: The planner starts from an initial guess, $\xi_o$, which in this example is the line connecting the start and goal points. $n = 30$: The path deforms around the edges of the obstacles, while samples are drawn across the entire domain $[0, 1]$. $n = 50$: Overall cost has reduced and path clears all obstacles. $n = 59$: Solution has converged.




(a) Initial (b) Final

Figure 5.3: Planning in a 3D map. Gray surface represents 0.5 occupancy isosurface. Planner resolves a safe path through the narrow passage. (a) Initial solution (b) Final solution.

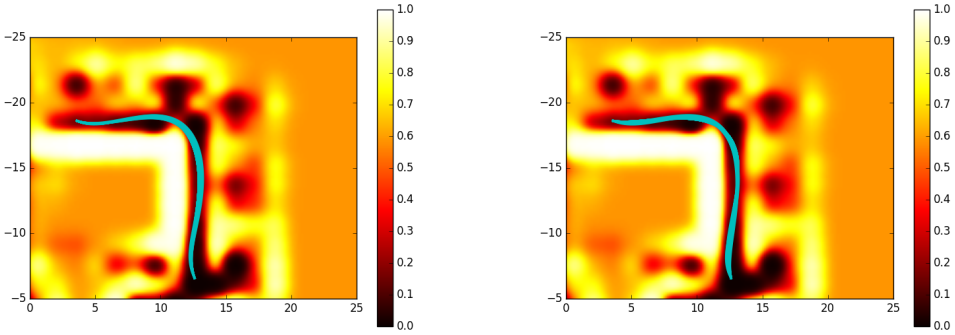
### 5.3.2 Real Data

The map for this experiment was fitted according to laser observations from the Intel-Lab dataset[3]. Both RFF and RBF planners used $m = 50$ features and $\gamma = 4$. $\lambda$ and $\eta_n$ are as in the simulations.

Figure 5.4 and Table 5.2 show a comparison between the different methods. RRT$^*$ forms a path based on several waypoints (states) the robot should pass from start to goal. As a result, the path typically is jagged, with short jerks. In contrast, the paths generated by our method are continuous and smooth. In addition, unless using inflated obstacles, RRT$^*$ paths tend to move close to the walls or cutting corners, as indicated by the relative high, and unsafe, occupancy of RRT$^*$ in Table 5.2. The stochastic planner follows the mid line between obstacles and perform smooth turns resulting in shorter and safer trajectories.

Table 5.2 provides quantitative comparisons between these planners. All planners obtain similar path length, which corresponds to the length of the corridor. Runtime results reveal the main advantage of the approximate kernel planner. While the stochastic non-parametric GP planner introduced in Chapter 4 requires less samples to converge, its actual runtime is almost 25 slower. Meaning that while the non-parametric GP path representation is highly expressive, it is not scalable due the cubic computational complexity. The RRT$^*$ runtime performance is quite poor too. Although RRT$^*$ samples occupancy and not the occupancy spatial gradient, it have similar per iteration cost as the approximate kernel approach. Nonetheless, its sampling efficiency in this scenario is low since most of the map is either occupied or unknown. In contrast to both methods, our method uses an approximate kernel path representation, which has a fixed linear complexity. Consequently, the properties of the path are similar to that of the non-parametric path, however adding more observations does not change the computational performance of the model. As a result, the time needed to obtain a solution by the proposed method is much shorter compared with the GP planner and RRT$^*$.

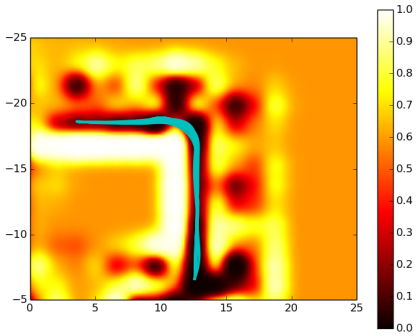
[3]Available at `http://radish.sourceforge.net/`



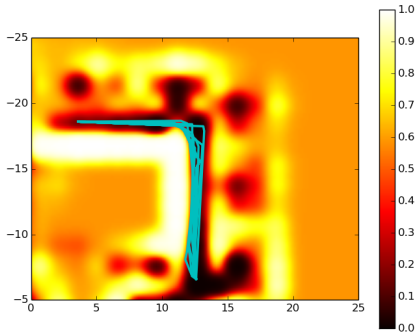
(a) Approx. Kernel RBF



(b) Approx. Kernel RFF



(c) GP Paths



(d) RRT*

Figure 5.4: Comparison of path planning methods on a continuous occupancy map of the Intel-Lab (partially shown); (a) RBF approximation, (b) RFF approximation, (c) stochastic non-parametric GP paths, which were introduced in Chapter 4 and (d) RRT* (Karaman, 2010). Each image shows ten paths generated by the planning algorithm, to indicate repeated performance.The planning methods in (a), (b) and (c) produce smooth paths which follow the mid lines between walls. RRT* paths are typically not smooth, and some have small jerks. In addition, RRT* paths pass dangerously close to walls.

Table 5.2: Intel dataset comparison

| | | Max. occupancy | Path length [m] | Samples | Runtime [s] |
|---|---|---|---|---|---|
| Approx. RBF | Kernel: | $0.34 \pm 0.05$ | $20.3 \pm 0.2$ | $1629 \pm 287$ | $3.2 \pm 0.6$ |
| Approx. RFF | Kernel: | $0.34 \pm 0.03$ | $20.3 \pm 0.1$ | $1861 \pm 60$ | $5.6 \pm 1.8$ |
| Gp Paths | | $0.38 \pm 0.08$ | $20.4 \pm 0.2$ | $398 \pm 132$ | $78.3 \pm 47.5$ |
| RRT* | | $0.49 \pm 0.05$ | $20.3 \pm 0.4$ | $10788 \pm 1462$ | $34.5 \pm 10.0$ |

### 5.3.3 Targeted Sampling

Figure 5.5 depicts planning using an adaptive proposal distribution $\mathcal{Q}$. This process is similar to the example shown in Fig. 5.2, however the uniform proposal distribution in Fig. 5.2 is replaced with a multinomial distribution with 10 intervals evenly distributed over the entire $[0, 1]$ domain. To dynamically update $\mathcal{Q}$, the sampler uses 10 queues, one for each interval, with a depth $D_Q = 5$. These queues are updated by samples of the objective function as discussed in Section 5.2.3.

Comparing Fig. 5.5 with Fig. 5.2 reveals a difference in the distribution of samples. As expected, the uniform proposal distribution used in Fig. 5.2 results in samples distributed uniformly over the $[0, 1]$ domain. In contrast, the majority of samples drawn by the adaptive proposal distribution, Fig. 5.5, are located around obstacles, although samples are still drawn over the entire domain.

The proposal distribution $\mathcal{Q}$ is derived solely from stochastic samples of the functional gradients. Therefore, samples are drawn where it is most effective. In Fig. 5.5, for $n < 100$, $\mathcal{Q}$ draws samples mainly in valid regions of the path, indicated by the low probability for $t > 0.6$, where the path crosses an obstacle. For $n = 110$, $\mathcal{Q}$ has a high probability around $t = 0.5$ (indicated by the red circle), which corresponds with samples on the border of the obstacle. These samples result in major changes to the path. Close to the stationary solution, there are only small changes in the trajectory. As a result, $\mathcal{Q}$ approximately returns to maximum entropy sampling, i.e, uniform sampling over the entire domain. However, as this process relies on stochastic samples, the final solution is only in the neighbourhood of the locally optimal solution (Bottou et al., 2016). Hence, by checking the entropy of $\mathcal{Q}$, in conjunction with a check of their validity, we can estimate whether $\mathcal{Q}$ is approximately uniform, which indicates that the solution has been found.

A performance comparison over 100 simulations of planning using an adaptive distribution as opposed to a uniform scheme is shown in Fig. 5.6 and Table 5.3. Figure 5.6 compares the number of iterations required to reach convergence. The adaptive sampling presents faster optimisation with smaller variance in the number of iterations required for convergence. Table 5.3 provides a quantitative comparison which reaffirms the advantages of targeted sampling, faster and more robust optimisation process. Moreover, there is no degradation in performance, as length is almost unaffected by the change in sampling procedures.

$\mathcal{Q}$ is also used as a convergence indicator. Path planning with uniform sampling converges when all samples are valid, i.e. safe. An adaptive sampling method, on the other hand, can reason on other convergence conditions, taking into consider-

Table 5.3: Adaptive sampling - Performance comparison

| | Adaptive Sampling | Uniform Fixed Sampling |
|---|---|---|
| Iterations to converge | $132 \pm 43$ | $202 \pm 98$ |
| Path length [m] | $21.07 \pm 0.36$ | $21.02 \pm 0.35$ |
| Converged solution [%] | 85 | 65 |

ation all the various components of the objective function. Fig. 5.7 depicts path safety and $\mathcal{Q}$'s entropy during optimisation.To emphasise this process, $\mathcal{Q}$ was set with 50 intervals. Initially, the maximum occupancy along the path, marked in red, is 1, a certain collision. As the optimisation progresses, the path clears from obstacles, indicated by a maximum occupancy below 0.5. At that point, a uniform sampling scheme would indicate converges. Although safe, this solution is not necessarily optimal with regards to the overall objective. The entropy of $\mathcal{Q}$, in blue, provides the complimentary condition for convergence. In the beginning of the optimisation, part of the path is invalid. This leads to a decrease in the sampling probability for those regions, and to a significant drop in the overall entropy. As the path deforms during optimisation, a greater part of it clears from obstacles, which results in a more balanced sampling distribution. Consequently, the entropy gradually increases.

The dips in the entropy graph are caused by samples that led to major shift in the path, which correspond to high functional gradient update. In the initial state of the optimisation, these updates originates from obstacle-related costs, that steer the path away from collision. The influence of the dynamic cost becomes more significant as the path deforms. The motion-cost gradient update, in most cases, counteract the effect of the obstacles' gradient. In Fig. 5.7, these dips continue on even after the path is considered safe. Using the entropy of $\mathcal{Q}$, one can reason on confidence of convergence, albeit to guarantee convergence, the entropy condition must be in conjunction with a safety validity check.

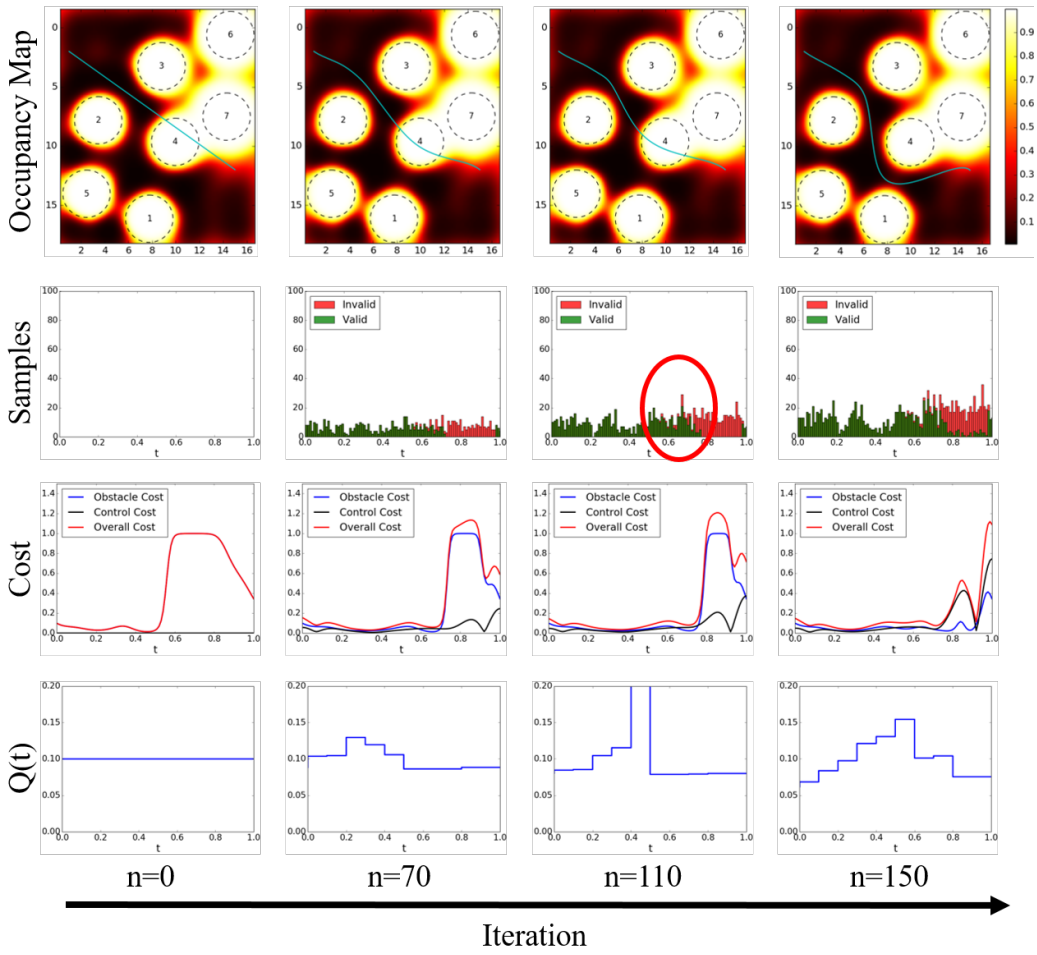



Figure 5.5: Planning using adaptive sampling. This figure is similar in its structure to Fig. 5.2, with the addition of an additional layer which depicts the proposal distribution $\mathcal{Q}$. The proposal distribution increases the probability of sampling where the objective is high, which is shown by the higher density of sampling around the obstacle. A good proposal distribution draws samples in high-impact areas, where the effect on the trajectory is more profound. On the final iteration, $\mathcal{Q}$ returns to an approximate uniform distribution, indicating convergence.

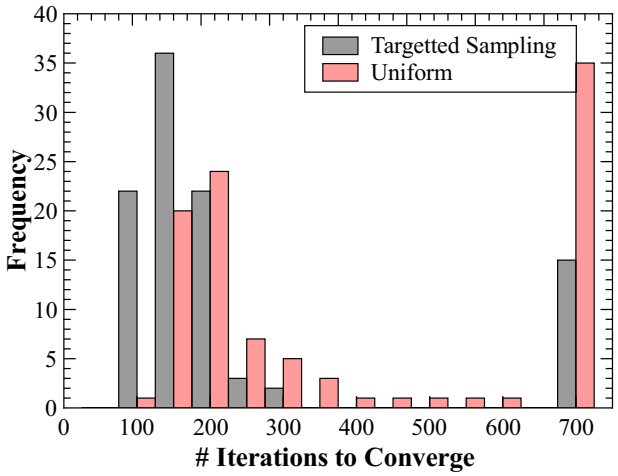



Figure 5.6: Convergence comparison between a dynamic proposal distribution and fixed uniform sampling distribution. The comparison was done using 100 simulations in as environment as shown in Fig. 5.6. Histogram depicts the distribution of the number of iterations required to reach convergence. An attempt is considered to have failed once more than 700 iteration have passed. Convergence of solution means a safe trajectory in case a uniform sampler is used. In the case a proposal distribution $\mathcal{Q}$ is used, convergence is a safe trajectory has been obtained and the entropy of $\mathcal{Q}$ has reach the required threshold (98% of a uniform distribution).

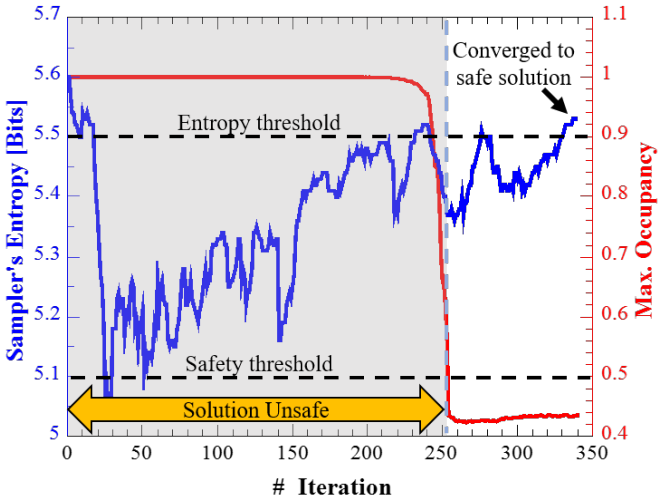



Figure 5.7: Indicting convergence using the entropy of $\mathcal{Q}$. The two conditions for convergence are safety and maximum entropy. A path is considered safe when the maximum occupancy along the path is below a safety threshold, i.e. no collision. The shaded area marks iterations during the optimisation where the path is not safe. The second condition for convergence requires that the sampler's entropy would be approximately the maximum attainable entropy.

## 5.4 Summary

The planning method proposed in this chapter employs SGD to optimise a path represented by a set of approximate kernel features. This model provides a highly expressive path representation which is computationally efficient. SGD combines the approximate kernel path model with a stochastic sampling schedule to form a computationally efficient optimisation process with convergence guarantees.

Planning in occupancy maps is a challenge for trajectory optimisers. Maps are a product of sensor observations, leading to contextual information gaps due to corrupt measurement or occlusions. As a result, the path can only be optimised around narrow transition areas on the edges of obstacles. Using a-priori sampling resolution of the objective function, as done in most trajectory optimisers, can not guarantee sampling in those areas. Employing stochastic samples across the entire path domain avoids the need to commit to an a-priori sampling resolution, which enables the optimiser to identify effective transition areas and overcome the gaps formed by the obstacles.

Experimental results, in simulation and with real data, demonstrate the importance of stochastic sampling for planning in occupancy maps. Combined with an approximate kernel path representation, our method offers a scalable and efficient method for trajectory optimisation in occupancy maps. The resulting trajectories are similar to the the non-parametric path representation, though with an order of magnitude faster convergence.

The introduction of an adaptive sampling procedure provides a useful mechanism to speed up convergence. More importantly, the proposal distribution acts as an indicator of convergence. Identifying convergence in a functional optimisation framework typically relies on testing whether the solution is safe or not. Using the proposal distribution to indicate convergence replaces this exhaustive check with a sample-based indicator. Such an indicator can also take into consideration an objective functional which consists of several sub-objectives.

In the next chapter we will show how this cost-effective path optimisation framework can be used to develop an FGD-based exploration method for continuous occupancy map. This approach offers a new take on exploration over continuous path, as it optimises using functional gradients instead of explicitly computing the MI reward.

# Chapter 6

# Functional Exploration

## 6.1 Introduction

Autonomous exploration is a complex task where the robot moves through an unknown environment with the goal of mapping it. The desired output of such a process is a sequence of paths that efficiently and safely minimise the uncertainty of the resulting map. However, optimising over the entire space of possible paths is computationally intractable. Therefore, most exploration methods relax the general problem by optimising a simpler one, for example finding the single next best view.

In Chapter 3, we developed a black-box optimisation process for exploration under constraints based on Bayesian optimisation. This approach allows global optimisation of the MI objective over continuous path. However, in practice, BO is limited in the dimensionality of the solution space. Therefore, optimisation is performed over a limited family of trajectories, e.g. cubic splines.

In this chapter, we formalise exploration as a variational problem. We present a novel approach based on *functional gradient descent* (FGD) to efficiently optimise exploratory paths over continuous occupancy maps. We use stochastic FGD to overcome the limitations of standard FGD methods in order to ensure convergence. This process enables optimisation over the entire path, resulting in continuous smooth paths that maximise the overall map quality while keeping the robot safe from collisions. Our contributions are:

1. A *Next Best Path* method. An information-driven variational framework for safe autonomous exploration in continuous occupancy maps. Path optimisation is performed directly in the space of trajectories using a combined objective; which considers safety, efficiency and information. The method is invariant to the choice of path representation as it uses stochastic functional gradient descent to optimise the objective along the entire path.

2. Developing a *mutual information* (MI) variational objective for continuous occupancy maps. This replaces the common and expensive approach of computing MI explicitly over the entire map for each evaluated path. Instead, our method modifies the path using MI functional gradients without

the need to compute MI explicitly. These gradients are obtained from local perturbations on the map model which are derived in closed-form.

The remainder of this chapter is organised as follows. Literature on autonomous exploration in continuous occupancy maps is surveyed in Section 6.2. Section 6.3 describes in detail the functional exploration algorithm. Experimental results, analysis and comparison with other exploration methods are presented in Section 6.4. Finally, Section 6.5 summarises the contributions of this chapter.

## 6.2 Related work

The goal of autonomous exploration is to produce a consistent environment model by minimising any uncertainties. In a mapping context, exploration is the process of producing high-fidelity maps (Stachniss, 2009). This is a complex problem mainly due the dimensionality of the solution space. Most exploration methods plan exploration paths over occupancy grid maps (Thrun et al., 2005), rather than continuous continuous occupancy maps. Regardless of the type of occupancy map, exploration methods take one of two forms; frontier-driven or information-theoretic. Juliá et al. (2012) provide a quantitative comparison between these exploration methods.

Frontier-based exploration methods drive the robot toward the borders of the known space (Yamauchi, 1997). In a grid map, frontiers are clusters of free cells neighbouring unknown cells. Once frontiers are identified, a separate path planner finds a safe path toward a selected frontier. Various utility functions can be used when choosing the most desirable frontier. The simplest form considers only the travelling costs (Yamauchi, 1997). González-Baños and Latombe (2002) choose a goal point based on a score of expected coverage penalised by the travelling costs. A generalised approach for goal point selection given several criteria were suggested by Holz et al. (2011).

Information-theoretic exploration methods optimise a utility function associated with the uncertainty of the map. Early work optimised the selection of a discrete goal point rather than optimising an entire path. Elfes (1996) suggested MI as an information metric for exploration, while Whaite and Ferrie (1997) proposed a *next best view* (NBV) approach using the entire map entropy. Vallvé and Andrade-Cetto (2015) use a potential field computed over the entire configuration space to find exploration candidates. However, this method assumes discrete steps, disregarding the reduction of entropy between consecutive robot poses. Charrow et al. (2015) combined the frontier and information-based

Part of this chapter is to appear in the 2017 *International Symposium on Robotics Research*.

methods. While they optimised the information heuristic over a continuous control input space, in effect the path consisted of a fixed number of time steps. Lauri and Ritala (2015) formulate the exploration problem as *partially observable Markov decision process* (POMDP) and used sample-based approach to solve the POMDP. Similarly to the work of Charrow et al., the action space is continuous, however, the path consists of a finite set of time steps, which is in contrast to the proposed method where optimisation is performed in the space of trajectories.

Only a handful of algorithms tackle exploration in continuous occupancy maps. Yang et al. (2013) employed *rapidly-exploring random tree* (RRT) to sample a set of feasible path candidates for a *Gaussian Processes* (GPs) maps. An adaptation of the frontier method for continuous occupancy maps was introduced by Jadidi et al. (2014), where a discretised frontier map was built from the continuous map. More recently, Jadidi et al. (2015) employed MI to rank frontiers. *Bayesian optimisation* (BO) has also been used for exploration. Marchant and Ramos (2014) utilised BO to optimise the selection of continuous informative paths over a continuous environmental model. Chapter 3 utilised constrained BO for safe exploration to learn an MI objective. While BO optimises continuous paths, in practice it uses only a limited parametrisation such as quadratic or cubic splines.

In summary, the exploration method proposed in this paper uses an information-based utility to optimise path selection. Employing calculus of variations, the optimisation procedure is invariant to the path representation. This is a major difference from existing methods that typically depend on a finite path parametrisation, such as finite sets of time steps or waypoints, or employ simple representations such as quadratic or cubic splines. Instead, our method can utilise a highly expressive path representation; such as non-parametric (Chapter 4) or approximate kernel paths (Chapter 5). As our method uses continuous occupancy maps, the MI utility can be derived directly from the map model in closed form, which simplifies computations.

## 6.3 Exploration Functional

The following section introduces our proposed functional exploration method. The use of functional gradient descent on an information-based objective results in trajectories which are highly expressive and safe while optimising the amount of information gained along the path.

### 6.3.1 Notation

We first re-introduce the notation used throughout the following sections. The notation is similar to Section 4.3, although some changes of notation are required

as we add additional objective functionals.

The workspace of the robot $W \in \mathbb{R}^3$ defines the space where obstacles lie and the map is queried. In addition, to account for the robot's finite size or its pose uncertainty, a set of body points, $B \in \mathbb{R}^3$ are defined. As the trajectory $\xi$ lies in configuration space $C$, we set a forward kinematic transform $g$ that maps a robot configuration, $\xi(t)$ and body point $b \in B$ to a point in the workspace $g : C \times B \to W$. To simplify notation, we define $x_t = g(\xi(t), b)$ as the workspace location for the pair $(t, b)$. We assume that the robot is equipped with a sensor, such as laser range finder, with a maximum range $R_{max}$ and an angular field of view $\Omega$.

For a given function $\xi$, a functional returns a single value $\mathcal{U} : \mathbb{R}^D \to \mathbb{R}$. Functionals are usually represented by an integral. However, whenever we compute the safety or MI functionals, the cost is computed from samples taken over $W$. As a result, the functional must be approximated by a *reduce* operator, e.g., average or maximum, that aggregates the cost along $\xi(\cdot)$. Given a workspace cost function $c\Big(g\big(\xi(t), b\big)\Big) : \mathbb{R}^3 \to \mathbb{R}$, we can approximate the functional by a sum over a finite set $\mathcal{T}(\xi) = \{t, b\}_i$ of time and body points:

$$\mathcal{U}_{obs,MI}(\xi) \approx \sum_{(t,b)\in\mathcal{T}(\xi)} c_{obs,MI}\Big(g\big(\xi(t), b\big)\Big) \equiv \sum_{(t,b)\in\mathcal{T}(\xi)} c_{obs,MI}(x_t) \tag{6.1}$$

In addition, we note the difference between gradient operators. We define $\nabla$ as a gradient with respect to $\xi$, while $\nabla_x$ is the workspace gradient.

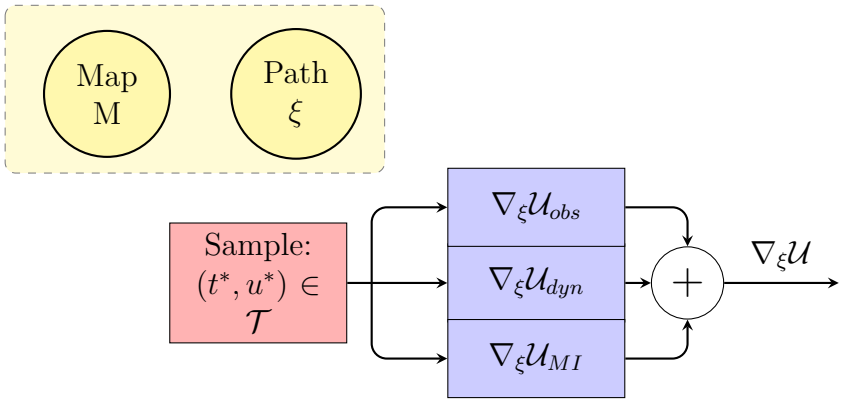
### 6.3.2 Exploration Functional Objective

The goal of autonomous exploration is to safely reduce the uncertainty of the environment model. Some exploration methods compute a finite set of go-to points that locally maximise information gain. Other methods optimise an information-based objective over the entire path, however, these are highly dependent on the path parametrisation. Formulating exploration as a variational problem and solving it using functional gradient descent, provides a general optimisation framework which is invariant to the choice of path parametrisation and can even take a non-parametric form as shown in Chapter 4.

The approach taken in our work relies on maximising mutual information along the entire path while keeping the trajectory safe. This is attained by constructing an objective functional $\mathcal{U}$ which is a weighted sum of three components:

- $\mathcal{U}_{obs}$ which maintains path safety by penalising proximity to obstacles as defined in Eq. (4.2),

Figure 6.1: Given a sample $(t, b)$, the functional $\mathcal{U}$ is the weighted sum of the various objectives; obstacle, dynamic and MI. $M$ and $\xi$ are global variables used to compute these objectives.




- $\mathcal{U}_{dyn}$ which penalises based on the shape of the trajectory, keeping path smooth and short as in Eq. (4.3) and,
- $\mathcal{U}_{MI}$ which rewards the mutual information gained along the path.

The overall objective takes a form of a weighted sum, as is also shown schematically in Fig. 6.1;

$$U(\xi) = \beta_{obs} U_{obs}(\xi) + \beta_{dyn} U_{dyn}(\xi) + \beta_{MI} U_{MI}(\xi). \tag{6.2}$$

Here $\beta_{obs}$,$\beta_{dyn}$,$\beta_{MI}$ are user-defined coefficients. In the following sections, we will introduce the different components of the functional objective; $U_{obs}(\xi)$, $U_{dyn}(\xi)$ and $U_{MI}(\xi)$. For each component we will derive its functional gradient assuming a Hilbert map as the environment model.

### 6.3.3 Mutual Information Functional $\mathcal{U}_{MI}(\xi)$

*Mutual information* (MI) is used in many autonomous exploration algorithms as an information-based objective function (Julian et al., 2014). In this context, MI is defined as the reduction in entropy conditioned on expected observations. Given an occupancy map $M$ and a set of expected observations $\hat{\mathbf{z}}$, we can define MI as:

$$MI(M; \hat{\mathbf{z}}) = H(M) - H(M|\hat{\mathbf{z}}) \tag{6.3}$$

where $H$ denotes Shannon's entropy. The main computational challenge of Eq. (6.3) is resolving the expected observations $\hat{\mathbf{z}}$. These are emulated observations that are produced by ray casting based on the sensor model. Determining $\hat{\mathbf{z}}$ and MI over an entire path is computationally intensive, leading most exploration methods to solve a relaxed MI optimisation problem, by either discretising or

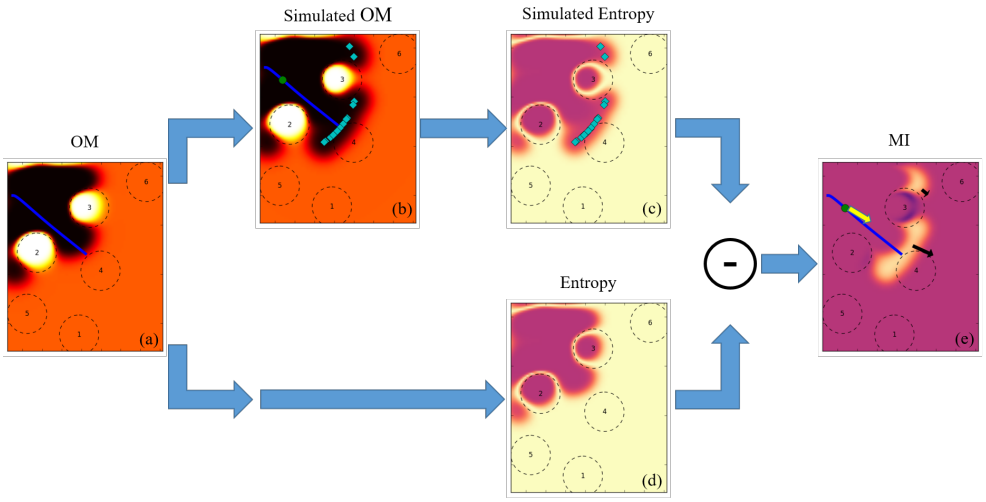



Figure 6.2: MI Functional gradient. The MI gradient is computed for a time sample $t$ given a continuous occupancy map $OM$ (a). The current path estimate $\xi$ is depicted in blue, while the expected pose at time $t$ is shown in green. The modified OM (b) is generated using hypothetical observations, shown as cyan diamonds, based on a robot's configuration at time $t$. Only observations at the edge of sensor range are considered for $t$. The entropy of the OM (c) and the modified OM (d) can be computed from the occupancy values (high entropy shown in white), which produces MI (e). The MI gradient is estimated by samples around observations (shown as black arrows). Accumulating gradient samples results in the overall MI gradient for $t$, shown by the yellow arrow on the robot. Note that the images of occupancy, entropy and MI are only given here for presentation purposes and full maps are never computed. The planner only accesses occupancy, entropy and gradient values through stochastic samples.

parameterising the paths.

The approach taken in this work uses the MI functional $\mathcal{U}_{MI}$ and its gradient to maximise MI efficiently over the entire trajectory. To compute $\mathcal{U}_{MI}$, an MI reward function is computed over the robot's workspace $W$, in a similar fashion to the computation of $\mathcal{U}_{obs}$:

$$c_{MI}(x_t) \approx MI(M; \hat{\mathbf{z}}|x_t). \tag{6.4}$$

However, computation of the conditional entropy $H(M|\hat{\mathbf{z}})$ entails changes to the Hilbert map model.

In the following section, we will describe the stages involved in computing the MI functional gradient. The three stages executed when computing $\nabla U_{MI}$ are:

- Simulating expected observations
- Creating a perturbed Hilbert map model $M|\hat{\mathbf{z}}$
- Obtaining MI functional gradient $\nabla U_{MI}$

Figure 6.2 shows the steps required for MI gradient computations at a given time $t$.

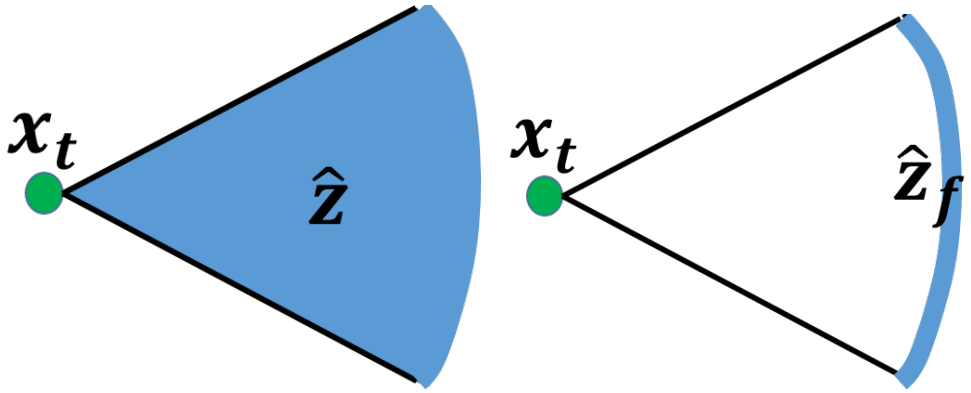
### Simulating expected observations

Similarly to the obstacle functional, the MI functional is approximated by a sum over a finite set of points $x_t$;

$$\mathcal{U}_{MI}(\xi) \approx \sum_{x_t \in \mathcal{T}(\xi)} c_{MI}(x_t). \tag{6.5}$$

$c_{MI}$ is chosen in a way that will estimate the infinitesimal change in MI at $x_t$. To compute $c_{MI}$ we emulate observations by ray casting, as done in other information-driven exploration techniques (Elfes, 1996; Jadidi et al., 2015). However, while other methods evaluate the MI reward over the entire field of view of the sensor, our method is only interested in the expected observations at the sensor's limits (maximum range). The rationale behind this approach is that new information about the environment will be mainly obtained at the sensor's sensing limits $R_{max}$. Figure 6.3 illustrates the difference between the two approach to compute MI.

The output of the ray casting process is a set of unoccupied expected observations $\hat{\mathbf{z}}_f$ for any time and body point $x_t$. $\hat{\mathbf{z}}_f$ differs from the expected observations of Eq. (6.3), as we are only interested in unoccupied (no obstacle) observation at the sensor maximum range. Figure 6.2b depicts $\hat{\mathbf{z}}_f$ as cyan diamonds.




(a) Standard Exploration (b) Functional Exploration

Figure 6.3: Difference in MI calculation. $x_t$ is the robot's pose for which MI is computed, the black lines depict the sensor field of view and the blue area is the region where MI is computed. (a) MI is computed over the entire field of view of sensor, producing $\hat{\mathbf{z}}$. (b) MI is computed only at the sensor's range limits, producing $\hat{\mathbf{z}}_f$

**Creating a perturbed Hilbert map model $M|\hat{\mathbf{z}}$**

In this step, we generate $M|\hat{\mathbf{z}}_f$, the modified Hilbert map conditioned on the expected observations. The straightforward approach is to train a new map model based on $\hat{\mathbf{z}}_f$. This approach is commonly used in exploration methods for occupancy grid maps. However the computational costs of such an approach are high, as new maps must be generated along the entire trajectory during optimisation. Instead, we propose the use of a perturbed Hilbert map model.

The perturbed Hilbert map model replaces the predictive map model $\sigma\Big(\mathbf{w}^T\hat{\Phi}(\mathbf{x})\Big)$, given in Eq. (2.45), with a perturbed model $\sigma\Big(\Psi(\mathbf{x},\hat{\mathbf{z}}_f)\Big)$. This model uses the expected observations as a dataset $\hat{\mathbf{z}}_f = \Big\{x_i, r_i\Big\}_{i=1}^{N}$, where $x_i \in \mathbb{R}^D$ is a point in 2D or 3D space and $r_i$ is the log-odds of the desired predictive occupancy posterior at $x_i$, to fit a GP:

$$\Psi(\mathbf{x},\hat{\mathbf{z}}_f) \sim GP\Big(\mathbf{w}^T\hat{\Phi}(\mathbf{x}), k(x_i,x_i)\Big). \tag{6.6}$$

We note that $\mathbf{w}^T\hat{\Phi}(\mathbf{x})$ of the current occupancy map is the mean function of the GP. The kernel function $k$ is the same function approximated by the Hilbert map features $\hat{\Phi}(\cdot)$. The predictive probability of the perturbed map $\hat{p}$ is given by:

$$\hat{p}(\mathbf{x},\hat{\mathbf{z}}_f) = \sigma\Big(\Psi(\mathbf{x},\hat{\mathbf{z}}_f)\Big) \tag{6.7}$$

Figure 6.2 depicts the resulting occupancy map following the embedding of the expected observations in an existing map.

The computational cost of the perturbed Hilbert map is cubic in the number

of expected observations $|\hat{\mathbf{z}}_f|$. As we are only modifying the map in a small region, a small set of observations is required to generate perturbation, keeping the computation load low.

### Obtaining MI functional gradient $\nabla\mathcal{U}_{MI}$

The workspace MI cost function for $x_t$ is defined as the MI summed over the entire map:

$$c_{MI}(x_t) = \int_M MI\Big(M(m); \hat{\mathbf{z}}_f(x_t)\Big)dm. \tag{6.8}$$

However, as we are only interested in the change at the limits of the sensor range, $c_{MI}$ can be replaced by;

$$c_{MI}(x_t) = \int_{m\in m_{max}(x_t)} MI\Big(M(m); \hat{\mathbf{z}}_f(x_t)\Big)dm, \tag{6.9}$$

where $m_{max}(x_t)$ denotes workspace locations which lie on the arc given by the maximum sensing range $R_{max}$ and the sensor's field of view $\Omega$, as shown in Fig. 6.3 . $c_{MI}$ can be approximated with a sum using either a deterministic or a Monte-Carlo schedule:

$$c_{MI}(x_t) \approx \sum_{m\in m_{max}(x_t)} MI\Big(M(m); \hat{\mathbf{z}}_f(x_t)\Big). \tag{6.10}$$

Given the approximations in Eqs. (6.1) and (6.10), the MI functional can be represented as

$$\mathcal{U}_{MI}(\xi) \approx \sum_{x_t\in\mathcal{T}(\xi)} \sum_{m\in m_{max}(x_t)} MI\Big(M(m); \hat{\mathbf{z}}_f(x_t)\Big). \tag{6.11}$$

The MI functional gradient follows the same form as Eq. (4.12):

$$\nabla\mathcal{U}_{MI}(x_t) = \sum_{x_t\in\mathcal{T}(\xi)} \sum_{m\in m_{max}(x_t)} \mathbf{J}(x_t)\nabla_x MI\Big(M(m); \hat{\mathbf{z}}_f(x_t)\Big). \tag{6.12}$$

The spatial gradient of MI, $\nabla_x MI\Big(M(m)$, can be expressed in closed-form using the Hilbert maps' continuous model. Using the MI definition from (6.3), the gradient is defined as

$$\nabla_x MI(M; \hat{\mathbf{z}}) = \nabla_x H(M) - \nabla_x H(M|\hat{\mathbf{z}}), \tag{6.13}$$

where $\nabla_x H$ is the spatial gradient of the entropy. Using the chain rule we rewrite

the spatial gradient of $H$ around a query point $x$ as:

$$\nabla_x H(x) = \frac{dH}{dp} \nabla_x p(x), \tag{6.14}$$

where $p$ is the probability of occupancy at $x$ given by Eq. (2.45). As $p$ is a Bernoulli random variable, $\frac{dH}{dp}$ is simply; $\frac{dH}{dp} = log_2 \frac{1-p}{p}$. The occupancy gradient $\nabla_x p(x)$ depends on the occupancy map used. For the unperturbed map, $\nabla_x p(x)$ is given by Eq. (4.6). The spatial gradient of the perturbed map, Eq. (6.7), can be computed similarly to the unperturbed map;

$$\begin{aligned} \nabla_x \hat{p}(\mathbf{x}, \hat{\mathbf{z}}_f) &= \nabla_x \sigma\Big(\Psi(\mathbf{x}, \hat{\mathbf{z}}_f)\Big) \\ &= \sigma\Big(\Psi(\mathbf{x}, \hat{\mathbf{z}}_f)\Big)\Big(1 - \sigma\Big(\Psi(\mathbf{x}, \hat{\mathbf{z}}_f)\Big)\Big) \nabla_x \Psi(\mathbf{x}, \hat{\mathbf{z}}_f). \end{aligned} \tag{6.15}$$

As $\Psi$ is a GP, its gradient can be computed in closed form.

Figure 6.2 shows the computation of MI gradient by sampling along the arc defined by the sensor range. Each sample produces an MI gradient, schematically shown by the black arrows. The overall MI gradient, which pushes optimisation toward exploratory trajectories is computed from the sum of all samples and is shown in Fig. 6.2e as a yellow arrow.

### 6.3.4 Functional Exploration Algorithm

In this section, we describe the functional exploration algorithm, which aims to find a safe path that maximises MI over its entire course.

Functional exploration is a general optimisation framework, meaning it is invariant to the choice of path representation. Functional optimisation methods have used waypoint parameterisation (Zucker et al., 2013), Gaussian process representation (Mukadam et al., 2016; Dong et al., 2016), or defined trajectories as RKHS (Marinho et al., 2016). However, in all methods the objective is sampled via a deterministic schedule. Such approaches have proved unsatisfactory for planning using occupancy maps (see Chapter 4). To ensure convergence to a safe solution, the path is stochastically sampled in the entire $t \in [0, 1]$ domain and any uninformative gradient updates, such as those coming from unsafe parts of the map, are rejected.

The path representation used in this work is based on kernel matrix approximations introduced in Chapter 5. The path is essentially a weighted sum of nonlinear features, similar to a nonlinear regression. This path model is highly expressive, yet concise. Most importantly, the path can be optimised by SGD (Robbins and Monro, 1951). Following Chapter 5, the general functional optimisation is transformed into an iterative optimisation process of the weights of the

path,

$$\mathbf{w}^{\xi}_{optimal} = \underset{\mathbf{w}^{\xi}}{\operatorname{argmin}} \mathcal{U}(\xi). \tag{6.16}$$

The algorithm for functional exploration is given in Algorithm 10. The essential inputs are the initial robot state and the occupancy map. Boundary conditions or an initial solution are optional inputs. In each iteration, a mini-batch is drawn. The safety of each sample is checked, and if it is below $P_{safe}$, the sample is used to update the path. The gradient of the various components (obs, dyn, MI) of the objective functional are computed, and summed according to Eq. (4.1). Once the overall functional gradient is computed, the weights $w^{\xi}_{n+1}$ are updated according to Eq. (5.7).

**Algorithm 10:** Stochastic Functional Exploration

**Input:** $M$: Occupancy Map.
$\xi(0)$: Start state.
$P_{safe}$: No obstacle threshold.
$\mathcal{Q}$: Optional proposal sampling distribution (use uniform $\mathcal{U}[0,1]$ is unspecified.
Optional: boundary conditions $\xi_b$, initial guess $\xi_o$.
**Output:** $\mathbf{w}^{\xi}_{optimal}$
$n = 0$
**while** *solution not converged* **do**
  //Stochastic sampling:
  $S \sim \mathcal{Q}$ Draw mini-batch from proposal distribution
  **foreach** $x_t \in S$ **do**
    $P_{occ} \leftarrow M(x_t))$ Eq. (2.45)
    **if** $P_{occ} \leq P_{Safe}$ **then**
      $\hat{\mathbf{z}}_f(x_t) \leftarrow$ simulated observations
      $\nabla \mathcal{U}_{obs} \leftarrow \frac{\partial}{\partial \xi} x_t \Big(\sigma\Big(\mathbf{w}^T \hat{\Phi}(x_t)\Big)\Big(1 - \sigma\Big(\mathbf{w}^T \hat{\Phi}(x_t)\Big)\Big)\mathbf{w}^T \nabla_x \hat{\Phi}(x_t)\Big)$ Eq.(4.12)
      $\nabla \mathcal{U}_{dyn} \leftarrow -\frac{d^2}{dt^2}\xi(t)$ Eq.(4.13)
      $\nabla \mathcal{U}_{MI} \leftarrow \sum_{m \in m_{max}(x_t)} \frac{\partial}{\partial \xi} x_t \nabla_x MI\Big(M(m); \hat{\mathbf{z}}_f(x_t)\Big)$ Eq.(6.12)
      $\nabla \mathcal{U} \leftarrow$ Combine using Eq.(6.2)
      $\mathbf{w}^{\xi}_{n+1} \leftarrow \mathbf{w}^{\xi}_{n} - \eta_n A^{-1} \hat{\Upsilon}(t_i)^T \hat{\Upsilon}(t_i) \nabla_{\xi} U(\xi_n)(t_i)$; update rule Eq. 5.7
    **end**
  **end**
  Update $\mathcal{Q}$ - Algo. 8
  $n = n + 1$
**end**

Figure 6.4 provides an insight into the optimisation process of a single planning iteration. Each iteration starts with an initial guess, which is depicted in blue. The obstacle functional gradient repels the path from obstacles and unknown

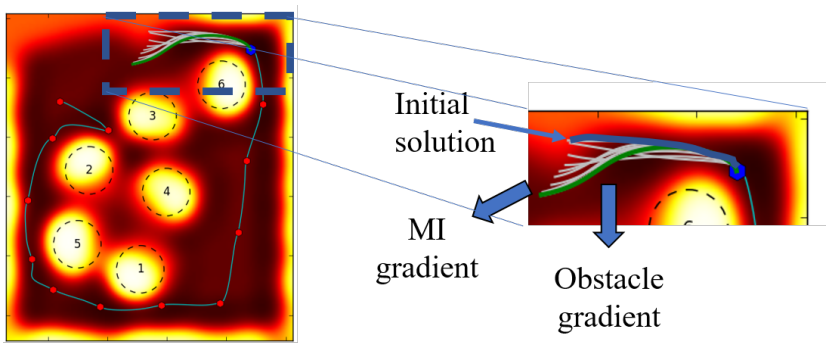



Figure 6.4: A functional planning iteration. Functional gradients deform the initial solution in blue. The obstacles gradient repels path from obstacles and unknown space. The MI gradient pulls path toward unexplored space. The resulting path, in green, is the optimised solution.

space. The MI objective pulls the path toward unexplored space. The intermediate path updates, following the functional gradient, are shown in grey. The final optimal path is shown in green.

## 6.4 Experimental Results

In this section, we evaluate the performance of the stochastic functional exploration planner and compare it to other exploration methods for continuous occupancy maps. As a benchmark, we chose to compare our method to the RRT-based exploration method of Yang et al. (2013) which we modified to use Hilbert maps. This method optimises MI during path selection, and while its bottleneck lies in computational complexity, it takes advantage of the fact the RRTs are probabilistically complete. Another method used for comparison is based on frontier exploration (Yamauchi, 1997), which takes a grid approximation of the continuous map in a similar approach to Jadidi et al. (2014). The path is then constructed by a smooth RRT planner which reasons only about the path safety. All methods, including Hilbert maps, are implemented in Python and tested on an Intel i5-6200U with 8GB RAM.

### 6.4.1 Simulations

Figure 6.5 shows a qualitative comparison at various planning iterations. While all exploration methods successfully build the map, there are some clear differences. The frontier-based method produces jerky paths and tends to move towards the edges and corners of the map. Path optimisation based methods, on the other hand, stay closer to obstacles. As the RRT-based method maximises MI only, its path moves closer to the edges of obstacles, while the proposed method keeps a bigger distance, as its objective includes obstacle safety explicitly. We

Table 6.1: Exploration performance comparison

| | Proposed method | RRT [1] | Frontier [2] |
|---|---|---|---|
| Avg. occupancy [%] | 1.2 | 5.1 | 3.0 |
| Max. occupancy [%] | 26.3 | 75.2 | 47.9 |
| Median Iter. Plan Time [s] | 65.4 | 200.6 | 13.4 |
| Avg. Iter. Plan Time [s] | 73.7 | 196.6 | 84.3 |
| Max. Iter. Plan Time [s] | 174.4 | 239.6 | 1345.2 |

[1] Yang et al. (2013)
[2] Yamauchi (1997)

note that increasing the safety margin by applying a blurring filter, as done with grid maps, is not applicable in continuous maps, as it requires expensive discretisation of the map.

Quantitative comparisons are shown in Fig. 6.6 and in Table 6.1. Figure 6.6 depicts the reduction in the entropy of the map. The rate of reduction is similar for both our method and the RRT-based exploration, albeit slightly better for the latter, mainly due the fact that the paths generated by the RRT move closer to obstacles. As a result, the RRT planner covers the map faster, however with a higher probability of collision, as shown in Table 6.1 by the maximum occupancy values exceeding the 50% occupancy threshold. The convergence of the frontier planner is slower, as a result of the over-estimation of the path utility by the choice of a single goal at each iteration. In addition, paths are jerky, leading to longer time to cover the same area. As the frontier planner does not explicitly minimise collision risk, the maximum occupancy over the path is high. In contrast, the maximum occupancy of the proposed method along the path is significantly below the 50% occupied threshold.

Comparing the median runtime results, shown in Table 6.1, reveals that the RRT planner is significantly slower than our method, mainly because MI is computed over entire paths. Since frontier only queries the map for occupancy, its runtime is significantly shorter. However, its average runtime is similar to our method as occasionally resolving longer path is required. It is worth noting that the runtime results are limited by the Python implementation of the Hilbert map. C++ implementation of the Hilbert map proved to be two to three orders of magnitude faster, which makes it suitable for online applications.

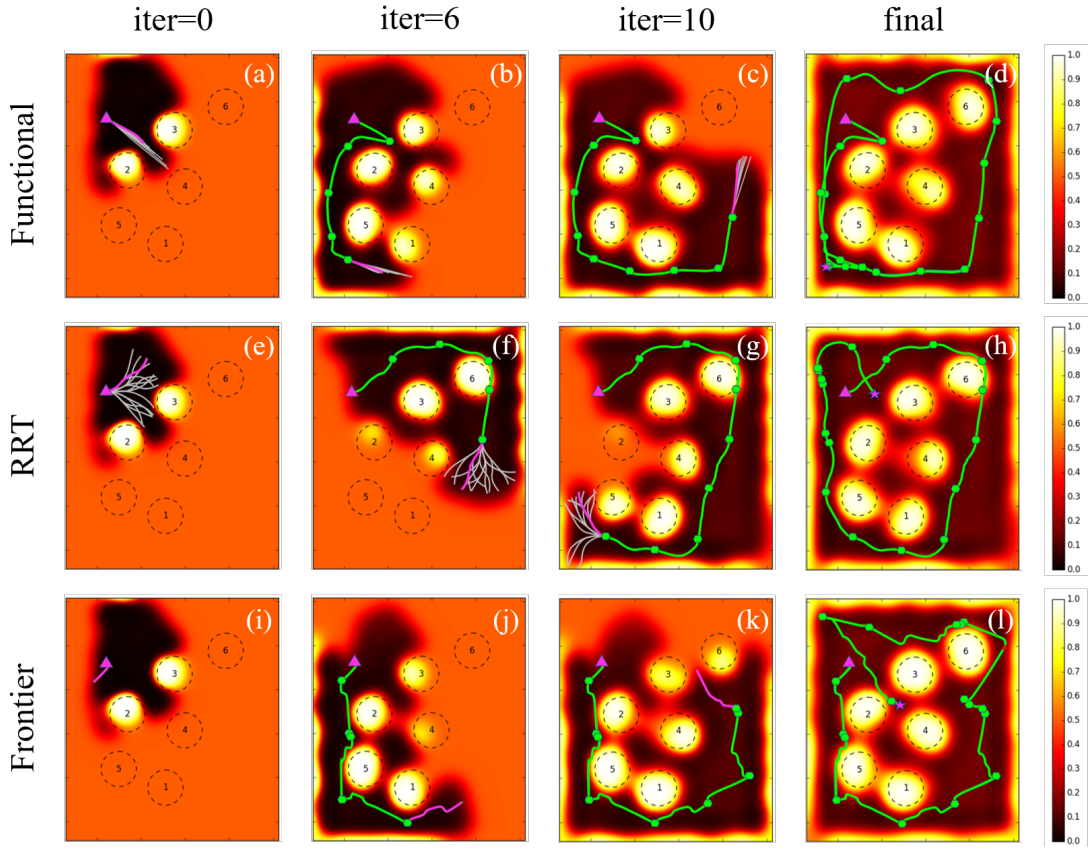



Figure 6.5: Comparison of exploration methods at various planning iterations using continuous occupancy maps. The hexagonal markers are the planning poses. Triangle and star represent start and end points, respectively. The grey lines are either candidate paths (RRT) or intermediate solutions (functional exploration). The pink line depicts the optimised path and the green line is the traversed path. The path is assessed during execution, and a re-plan step is invoked if a path is no longer safe.

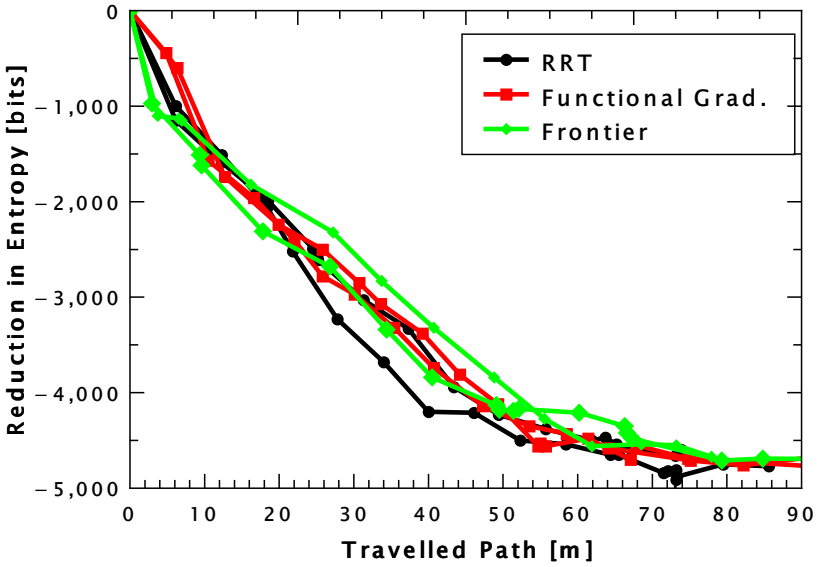



Figure 6.6: Comparison of exploration methods - 2 repetitions; the proposed functional exploration method (red), RRT-based exploration (black) and frontier-based exploration (green). RRT and the propose method converge in a similar fashion as both optimise MI over entire path. The goal-based approach of frontier converges slower as it does not explicitly optimise on MI.

### 6.4.2 Real World Scenario

To evaluate the performance of the functional exploration algorithm in a real world scenario, we simulated a robot exploring the Intel-Lab. We used the Intel-Lab dataset (available at http://radish.sourceforge.net/), which we also used in Chapter 4 and Chapter 5, to generate a ground truth, shown in Fig. 6.7, from which we can emulate range observations. However, the exploring robot does not have direct access to the ground truth map. The map in Fig. 6.7 reveals a relatively simple structure. Yet, the small rooms and narrow corridors pose a difficulty to a robot with limited manoeuvrability. To prevent a situation where the robot is stuck in a room, we added a reverse-on-path option. Meaning, if the robot identifies a dead-end, it may reverse on the path that took it to that spot.

Figure 6.8 shows the exploration process at various planning iterations. The generated Hilbert map is overlaid with the ground truth map of 6.7 as a reference to the map accuracy. The robot successfully explores the majority of the map, moving mainly in the main corridor. It enters only some of the rooms, only where there is enough clearance at the entrance. We note that the robot only relies on occupancy around the entrance to assess safety.

Figure 6.8 provides an insight to the path optimisation process. This is shown by the intermediate paths (in grey), which reveals how the functional objective in

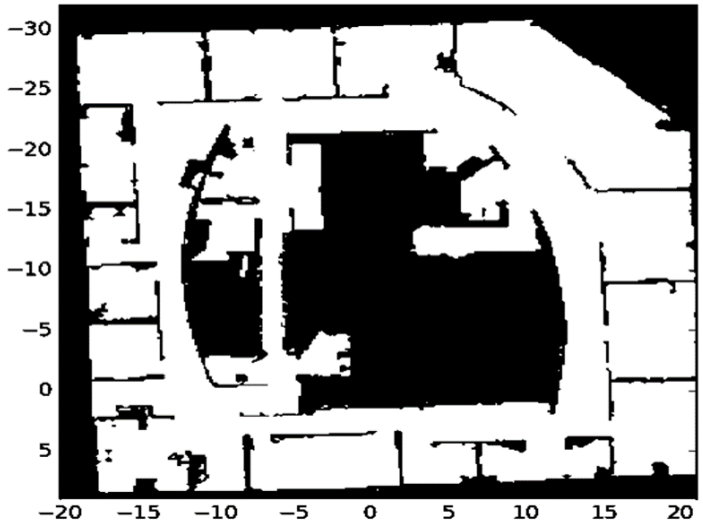
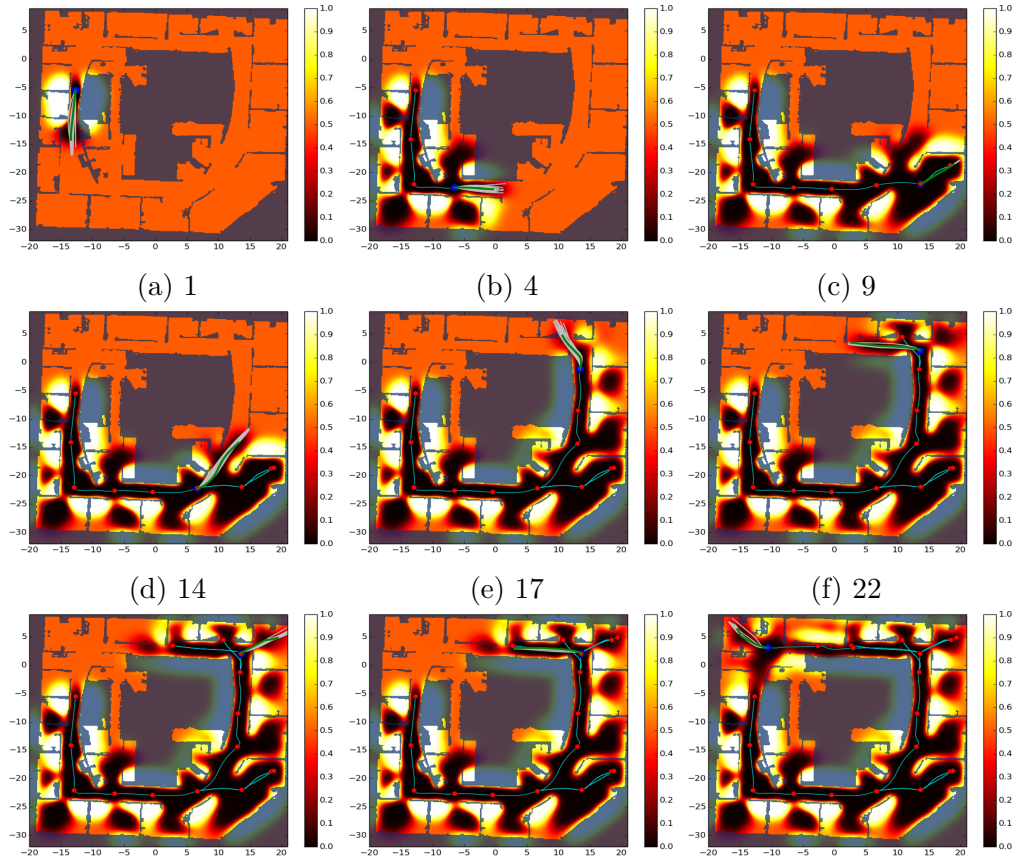



Figure 6.7: Intel-Lab - Ground truth map based on the Intel-Lab dataset. The robot does not have access to this map. It is only used to emulate range observations.

Eq. (6.2) balances safety with exploration during the path selection process. The MI term in Eq. (6.2) pulls paths towards the border between known and unknown space. The safety functional, on the other hand, maintains a safe distance from obstacles and unknown space. Consequently, paths tend to move in the middle of the corridors and and end close, but within some margin, to a frontier.

The main limitation of the functional exploration approach is the lack of global context during the optimisation. As FGD in a local optimisation process, its outcome depends on the starting point of the optimisation. This make FGD sensitive to dead-ends. An exploration dead-end scenario is shown in Fig. 6.8c, where a robot is inside a room unable to find its way out. When the robot is inside a room, it can not identify any planning horizon in its local neighbourhood. Meaning, the MI functional's contributions during optimisation are negligible, which results in non-exploring paths. To somewhat resolve this problem, we added a reverse-on-path option when the algorithm identifies a dead-end. However, a more robust solution may include a global exploration initial guess, such as a frontier, to start the optimisation. This will require to develop a frontier detection method for continuous occupancy maps, as current frontier exploration methods require to discretise the occupancy map, which is computationally intensive.



(a) 1 (b) 4 (c) 9

(d) 14 (e) 17 (f) 22

(g) 25 (h) 29 (i) 32

Figure 6.8: Functional exploration in the Intel-Lab at various planning iterations using continuous occupancy maps.The blue overlay depicts the ground truth, as shown in Fig. 6.7. The grey lines are intermediate solutions and the green line is the optimal path. The path is assessed during execution, and a re-plan step is invoked if a path is no longer safe. The traversed path is plotted in cyan, the red hexagons are the planning poses and the blue hexagon is the current pose.

## 6.5 Summary

This chapter introduces a novel method for exploration over continuous occupancy maps using stochastic functional gradient descent. This approach formalises exploration as a variational problem, where optimisation is performed directly in the space of trajectories. The functional objective of the proposed method explicitly optimises both safety and information collection over the entire path, finding the *Next Best Path*. While this approach can be used with any type of occupancy map, it is highly effective with Hilbert maps, where the introduced MI objective and its gradient can be computed from a perturbed model of the map. Our proposed approach eliminates the need for computing MI over the entire map as done in other exploration techniques. Rather, it computes variations to the path based on functional gradient of MI which are efficiently derived

in closed-form from the map model.

Comparisons with other exploration methods show that the proposed method improves on both safety and MI. Point exploration methods, such as frontier, which do not optimise the path selection, exhibit slower exploration rates. On the other hand, sampling-based exploration methods, such as (Yang et al., 2013), do not include safety in their objective, hence the resulting paths tends to move closer to obstacles. Moreover, these methods are computationally expensive due to the need to repeatedly sample the MI objective over entire paths. In comparison, our proposed method achieves similar exploration rates to (Yang et al., 2013) while taking less time to compute and still maximising safety.

# Chapter 7

# Conclusions

This thesis addresses the problem of autonomous exploration over continuous paths. Exploration is an active learning process aimed at minimising uncertainty and producing high-fidelity maps. Ideally, each autonomous decision corresponds to trajectories a robot should follow in order to maximise learning while ensuring its safety. However, optimising the selection of an uncountable object such as a trajectory poses computational difficulties. Consequently, the existing methods apply various heuristics in order to relax the general exploration problem.

In this work, we apply a holistic approach to exploration which solves simultaneously the coupled problems of where and how to explore. Instead of breaking exploration into independent sub-problems of finding goal points and planning safe paths to these points, we find an optimal path that maximises the robot's objective over the entire path. With exploration defined as an optimisation problem over continuous paths, we examine two different optimisation paradigms; Bayesian and functional.

In this chapter, Section 7.1 reviews the contributions and conclusions developed over the course of this thesis. Section 7.2 identifies potential directions for future work.

## 7.1 Summary of contributions

### 7.1.1 Constrained Bayesian Exploration

*Constrained Bayesian exploration* (CBE) is a global optimisation framework for exploration, which is based on constrained Bayesian optimisation. CBE optimises path selection directly, without a need to define intermediate goal points for optimisation. Since the exploration objective has no closed-form and is expensive to evaluate, CBE replaces its direct optimisation by an optimise-while-learn approach.

To facilitate an efficient optimisation, the optimiser utilises Bayesian inference to learn the models of the objective and constraints. These models are then used to generate a coherent objective function that incorporates gains, costs and risks of any path, allowing efficient identification of potential optimal solutions

that satisfy the constraints with high confidence. Consequently, CBE provides a principled and robust approach for optimising exploration over continuous paths that guarantees convergence to a local solution and follows well known BO's regret bounds.

The main disadvantage of CBE is the restrictions on path parameterisation. To maintain an efficient BO, the number of parameters that define a path is limited. In effect, CBE uses a predefined family of trajectories, such as quadratic or cubic splines, which are a small subspace of the entire valid trajectories space. With limited expressiveness, there is a higher risk for planning failures, where a valid solution exists, however it requires a richer representation.

### 7.1.2 Stochastic Path Planning using Continuous Occupancy Maps

The stochastic path planner offers a functional optimisation framework for continuous occupancy maps that employs highly expressive non-parametric path representation. Effectively, The stochastic path planner extends the trajectory optimisation framework, which is suited for discrete cost maps, to a continuous environment representation. This enables a new planning paradigm for occupancy maps, which is not based on the prevalent approach of planning using sampling-based techniques.

The stochastic path planner takes advantage of the fact that in a continuous occupancy map, spatial gradients of the occupancy can be computed in closed form. It forms a flexible support for a GP path representation using stochastic samples of the occupancy gradient. Unlike other trajectory optimiser, the planner does not commit to an a-priori parametric resolution of the support, rather it employs support points from the entire domain. As a result, the GP path establishes a highly expressive non-parametric representation, which allows the stochastic planner to better handle obstacles while optimising its functional objective.

The main limitation of the stochastic planner is its computational complexity. The underlying GP path representation carries a cubic update cost and a quadratic query cost. These costs limit performance once the number of gradient samples reaches several hundreds.

### 7.1.3 Scalable Stochastic Path Planner

The GP path model provides great flexibility but with a high, and potentially limiting, computational cost. The scalable stochastic planner fixes this cost by constructing an expressive and tractable path model based on kernel approximations. Using a finite set of approximating features, the scalable planner employs

SGD to optimise the path functional. SGD combines the approximate kernel path model with a stochastic sampling schedule to form a computationally efficient optimisation process under SGD convergence guarantees.

The scalable planner is a generalisation of the motion planning in RKHS paradigm of Marinho et al. (2016) as it can utilise any kernel matrix approximation technique. Moreover, the approximate kernel representation is naturally updated by stochastic samples, which replaces the predetermined sampling schedule of other kernel-based path planners (Marinho et al., 2016; Mukadam et al., 2016). With a finite path representation, the cost of updating and querying the path model is fixed, which resolves the main computational bottleneck of the non-parametric GP path representation.

The introduction of an adaptive sampling procedure provides a useful mechanism to accelerate optimisation and to indicate convergence. The adaptive sampler constructs a proposal distribution based on the usefulness of past samples. It then uses this distribution to draw more samples in areas of interest. As the proposal distribution adapts according to the changing functional, its entropy can also be used to identify convergence. This approach replaces the exhaustive evaluations of path safety typically required in a standard functional optimisation setting. It also provides a simple convergence indicator for an objective functional which consists of several sub-objectives.

### 7.1.4 Functional Exploration

Functional exploration is an information-driven variational framework for safe autonomous exploration in continuous occupancy maps. The functional objective of the proposed method explicitly optimises safety and information collection over the entire path, finding the *Next Best Path*. As a functional optimiser, the functional exploration planner is invariant to the choice of path representation. It uses stochastic samples of the functional gradient in order to optimise the objective along the entire path.

To optimise information collection, a mutual information variational objective was developed. While the MI objective can potentially be computed from any type of occupancy map, it is well suited for Hilbert maps, where the MI functional can be derived efficiently, in closed-form from a perturbed Hilbert map occupancy model. Using MI-driven functional path variations eliminate the need for an explicit computation of MI over the entire map, as done in other exploration techniques. Consequently, the functional exploration planner requires less computational resources, while still maximising safety and information.

## 7.2 Future work

In this section we provide future research ideas based on the methods and results presented throughout this thesis.

### 7.2.1 Stochastic Variational Inference GPC

The main drawback of the constrained Bayesian exploration method is its computational cost, arising from the cubic cost of updating and querying GPCs. To alleviate this limitation, GPCs can be replaced by a stochastic variational inference GPC (Hensman et al., 2015), which fixes the computational cost, making it invariant to the number of data points.

### 7.2.2 Latent Variable Functional Path Planning

*Latent variables models* (LVMs) (Bishop, 2006) are a dimensionality reduction technique that can offer an abstraction layer for path planning. The choice of approximating features in current stochastic path planners is quite arbitrary. LVMs, on the other hand, provide a principled mechanism for choosing these features. Conceptually, an LVM gives a "physical explanation" for the data. In an occupancy gradient observations, an LVM can capture high-level features such as corners or intersections. Another benefit of a lower dimensional representation is in the stability and efficiency of the optimisation process.

As LVMs are probabilistic, they can be used as generative models. Meaning, that a path can be sampled from the LVM distribution. Such a generative model can be employed under a a global optimisation framework, such as Hamiltonian Monte Carlo (Neal, 2010).

### 7.2.3 Generative Adversarial Functional Path Planning

The *generative adversarial networks* (GAN) is a neural network architecture that employs a pair generator-discriminator networks (Goodfellow et al., 2014). GANs are used to improve unsupervised learning or as generative models. In a path planning context, GANs can be used as generative model that suggest "intuitive" initial guesses, $\xi_o$ in (5.5), for a functional path planning. Currently, $\xi_o$ can be defined by a simple crude planner, such as RRT. However, GANs can offer a more principled approach for producing $\xi_o$.

# Bibliography

M.A. Alvarez, L. Rosasco, and N.D. Lawrence. Kernels for Vector-Valued Functions: a Review. *Foundations and Trends in Machine Learning*, 4(3):195–266, 2012.

N. Basilico and F. Amigoni. Exploration Strategies Based on Multi-Criteria Decision Making for an Autonomous Mobile Robot. In *Proc. of International Conference on Autonomous Agents and Multiagent Systems*, pages 1–6, 2011.

P. Bhattacharya and M.L. Gavrilova. Voronoi Diagram in Optimal Path Planning. In *4th International Symposium on Voronoi Diagrams in Science and Engineering*, pages 38–47, 2007.

C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

L. Bottou. Large-Scale Machine Learning with Stochastic Gradient Descent. In *Proc. of the 19th International Conference on Computational Statistics*, pages 177–186, Heidelberg, 2010.

L. Bottou, F.E. Curtis, and J. Nocedal. Optimization Methods for Large-Scale Machine Learning. *arXiv preprint arXiv:1606.04838*, 2016.

F. Bourgault, A.A. Makarenko, S.B. Williams, B. Grocholsky, and H.F. Durrant-Whyte. Information Based Adaptive Robotic Exploration. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 540–545, 2002.

E. Brochu, V.M. Cora, and N. De Freitas. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. *preprint arXiv:1012.2599*, 2010.

B. Charrow, G. Kahn, S. Patil, S. Liu, K. Goldberg, P. Abbeel, N. Michael, and V. Kumar. Information-Theoretic Planning with Trajectory Optimization for Dense 3D Mapping. In *Proc. of Robotics: Science and Systems*, 2015.

B. Curless and M. Levoy. A Volumetric Method for Building Complex Models from Range Images. In *Proc. of the Annual Conference on Computer Graphics and Interactive Techniques*, pages 303–312, 1996.

J. Dong, M. Mukadam, F. Dellaert, and B. Boots. Motion Planning as Probabilistic Inference using Gaussian Processes and Factor Graphs. In *Proc. of Robotics: Science and Systems*, 2016.

C. Dornhege and A. Kleiner. A Frontier-Void-Based Approach for Autonomous Exploration in 3D. *Advanced Robotics*, 27(6):459–468, apr 2013.

A. Elfes. Using Occupancy Grids for Mobile Robot Perception and Navigation. *Computer*, 22(6):46–57, 1989.

A. Elfes. Robot Navigation: Integrating Perception, Environmental Constraints and Task Execution within a Probabilistic Framework. In *Reasoning with Uncertainty in Robotics*, volume 1093, chapter 4, pages 91–130. Springer Berlin Heidelberg, 1996.

G. Francis, L. Ott, and F. Ramos. Stochastic Functional Gradient for Motion Planning in Continuous Occupancy Maps. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 3778–3785, 2017.

S. Garrido, L. Moreno, M. Abderrahim, and F. Martin. Path Planning for Mobile Robot Navigation Using Voronoi Diagram and Fast Marching. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2376–2381, 2006.

M.A. Gelbart, J. Snoek, and R.P. Adams. Bayesian Optimization with Unknown Constraints. In *Proc. of Uncertainty in Artificial Intelligence*, 2014.

H.H. González-Baños and J.C. Latombe. Navigation Strategies for Exploring Indoor Environments. *The International Journal of Robotics Research*, 21(10-11):829–848, 2002.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In *Proc. of Advances in Neural Information Processing Systems*, pages 2672—-2680, 2014.

R.B. Gramacy and H.K.H. Lee. Optimization Under Unknown Constraints. In *Bayesian Statistics 9*, pages 229–256. Oxford University Press, 2011.

G. Grisetti, C. Stachniss, and W. Burgard. Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

L. Györfi, M. Kohler, A. Krzyżak, and H. Walk. *A Distribution-Free Theory of Nonparametric Regression*. Springer Series in Statistics. Springer New York, 2002.

J. Hensman, N. Fusi, and N.D. Lawrence. Gaussian Processes for Big Data. In *Proc. of Uncertainty in Artificial Intelligence*, 2013.

J. Hensman, A. Matthews, and Z. Ghahramani. Scalable Variational Gaussian Process Classification. In *Proc. of International Conference on Artificial Intelligence and Statistics*, pages 351–360, 2015.

G.A. Hollinger and G.S. Sukhatme. Sampling-Based Robotic Information Gathering Algorithms. *The International Journal of Robotics Research*, 33(9), 2014.

D. Holz, N. Basilico, F. Amigoni, and S. Behnke. A Comparative Evaluation of Exploration Strategies and Heuristics to Improve Them. In *Proc. of European Conference on Mobile Robots*, pages 25–30, 2011.

V. Indelman, L. Carlone, and F. Dellaert. Planning in the Continuous Domain : a Generalized Belief Space Approach for Autonomous Navigation in Unknown Environments. *The International Journal of Robotics Research*, 34(7):1–56, 2015.

M. G. Jadidi, J. V. Miro, and G. Dissanayake. Mutual Information-based Exploration on Continuous Occupancy Maps. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6086–6092, 2015.

M.G. Jadidi, J.V. Miro, R. Valencia, and J. Andrade-Cetto. Exploration on Continuous Gaussian Process Frontier Maps. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 6077–6082, 2014.

M. Juliá, A. Gil, and O. Reinoso. A Comparison of Path Planning Strategies for Autonomous Exploration and Mapping of Unknown Environments. *Autonomous Robots*, 33(4):427–444, 2012.

B.J. Julian, S. Karaman, and D. Rus. On Mutual Information-Based Control of Range Sensing Robots for Mapping Applications. *The International Journal of Robotics Research*, 33(10):1375–1392, 2014.

S.J. Julier and J.K. Uhlmann. New Extension of the Kalman Filter to Nonlinear Systems. In *Proc. of 11th International Symposium on Aerospace/Defense, Simulation and Controls*, pages 182–193, 1997.

M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. STOMP: Stochastic Trajectory Optimization for Motion Planning. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 4569–4574, 2011.

S. Karaman. Incremental Sampling-Based Algorithms for Optimal Motion Planning. *Proc. of Robotics: Science and Systems*, 104, 2010.

S. Karaman and E. Frazzoli. Sampling-Based Algorithms for Optimal Motion Planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

L.E. Kavraki, P. Svestka, J.C. Latombe, and M.H. Overmars. Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

O. Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

T. Kollar and N. Roy. Trajectory Optimization using Reinforcement Learning for Map Exploration. *The International Journal of Robotics Research*, 27(2): 175–196, 2008.

M. Lauri and R. Ritala. Planning for Robotic Exploration Based on Forward Simulation. *Robotics and Autonomous Systems*, 83:15–31, 2015.

S. M. Lavalle. Rapidly-Exploring Random Trees: A New Tool for Path Planning. Technical report, Department of Computer Science. Iowa State University., 1998.

S. M. LaValle. *Planning Algorithms*. Cambridge university press, 2006.

T. Lozano-Pérez and M.A. Wesley. An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles. *Communications of the ACM*, 22(10): 560–570, 1979.

A.A. Makarenko, S.B. Williams, F. Bourgault, and H.F. Durrant-Whyte. An Experiment in Integrated Exploration. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and System*, volume 1, pages 534–539, 2002.

R. Marchant and F. Ramos. Bayesian Optimisation for Informative Continuous Path Planning. In *Proc. of International Conference on Robotics and Automation*, pages 6136–6143, 2014.

R. Marchant, F. Ramos, and S. Sanner. Sequential Bayesian Optimisation for Spatial-Temporal Monitoring. In *International Conference on Uncertainty in Artificial Intelligence*, pages 553–562, 2014.

Z. Marinho, B. Boots, A. Dragan, A. Byravan, G. J. Gordon, and S. Srinivasa. Functional Gradient Motion Planning in Reproducing Kernel Hilbert Spaces. In *Proc. of Robotics: Science and Systems*, 2016.

R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. Castellanos. Active Policy Learning for Robot Planning and Exploration under Uncertainty. In *Proc. of Robotics: Science and Systems*, 2007.

R. Martinez-Cantin, N. De Freitas, E. Brochu, J. Castellanos, and A. Doucet. A Bayesian Exploration-Exploitation Approach for Optimal Online Sensing and Planning with a Visually Guided Mobile Robot. *Autonomous Robots*, 27(2): 93–103, aug 2009.

A. Melkumyan and F. Ramos. A Sparse Covariance Function for Exact Gaussian Process Inference in Large Datasets. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 1936–1942, 2009.

T. P. Minka. *A Family of Algorithms for Approximate Bayesian Inference*. Doctoral dissertation, Massachusetts Institute of Technology, 2001.

M. Mukadam, X. Yan, and B. Boots. Gaussian Process Motion Planning. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 9–15, 2016.

R.M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, New York, 1996.

R.M. Neal. Regression and Classification using Gaussian Process Priors. In *Bayesian Statistics 6*. Oxford University Press., 1998.

R.M. Neal. MCMC Using Hamiltonian Dynamics. In *Handbook of Markov Chain Monte Carlo*. Chapman & Hall /CRC Press, 2010.

S.T. O'Callaghan and F.T. Ramos. Gaussian Process Occupancy Maps. *The International Journal of Robotics Research*, 31(1):42–62, 2012.

H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto. Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1366–1373, 2017.

C. Park, J. Pan, and D. Manocha. ITOMP: Incremental Trajectory Optimization for Real-Time Replanning in Dynamic Environments. In *Proc. of the International Conference on Automated Planning and Scheduling*, 2012.

J.C. Platt. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

T. Poggio and F. Girosi. Networks for Approximation and Learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A.Y. Ng. ROS an open-source Robot Operating System. In *Proc. of the ICRA Workshop on Open Source Software*, 2009.

M. Rafieisakhaei, A. Tamjidi, S. Chakravorty, and P.R. Kumar. Feedback Motion Planning Under Non-Gaussian Uncertainty and Non-Convex State Constraints. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 4238–4244, 2016.

Ali Rahimi and Benjamin Recht. Weighted Sums of Random Kitchen Sinks: Replacing minimization with randomization in learning. In *Proc. of Advances in neural information processing systems*, pages 1313–1320, 2009.

F. Ramos and L. Ott. Hilbert Maps: Scalable Continuous Occupancy Mapping with Stochastic Gradient Descent. In *Proc. of Robotics: Science and Systems*, 2015.

C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. University Press Group Limited, 2006.

N. Ratliff, M. Zucker, J.A. Bagnell, and S. Srinivasa. CHOMP: Gradient Optimization Techniques for Efficient Motion Planning. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 489–494, 2009.

R. Rifkin and A. Klautau. In Defense of One-Vs-All Classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

B. Schölkopf and A.J. Smola. *Learning with Kernels : Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

B. Schölkopf, A. Smola, and K.R. Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10(5):1299–1319, 1998.

J. Schreiter, D. Nguyen-Tuong, M. Eberts, B. Bischoff, H. Markert, and M. Toussaint. Safe Exploration for Active Learning with Gaussian Processes. In *Proc. of the European Conference on Machine Learning*, pages 133–149, 2015.

R. Senanayake and F. Ramos. Bayesian Hilbert Maps for Dynamic Continuous Occupancy Mapping. In *Conference on Robotics Learning*, 2017.

R. Shade and P. Newman. Choosing Where to Go: Complete 3D Exploration with Stereo. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 2806–2811, 2011.

S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal Estimated Sub-Gradient Solver for SVM. *Mathematical Programming*, 127(1):3–30, 2011.

S. Shen, N. Michael, and V. Kumar. Stochastic Differential Equation-Based Exploration Algorithm for Autonomous Indoor 3D Exploration with a Micro-Aerial Vehicle. *The International Journal of Robotics Research*, 31(12):1431–1444, 2012.

J. Snoek, H. Larochelle, and R.P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Proc. of Advances in Neural Information Processing Systems*, pages 2951–2959, 2012.

E. Solak, R. Murray-Smith, W.E. Leithead, C. Rasmusson, and D.J. Leith. Derivative Observations in Gaussian Process Models of Dynamic Systems. In *Proc. of Advances in neural information processing systems*, pages 1057–1064, 2002.

N. Srinivas, A. Krause, S.M. Kakade, and M. Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *Proc. of the International Conference on Machine Learning*, pages 1015–1022, 2010.

C. Stachniss. *Robotic Mapping and Exploration*. Springer, 2009.

C. Stachniss, G. Grisetti, and W. Burgard. Information Gain-Based Exploration Using Rao-Blackwellized Particle Filters. In *Proc. of Robotics: Science and Systems*, 2005.

I.A. Sucan, M. Moll, and L.E. Kavraki. The Open Motion Planning Library. *Robotics & Automation Magazine*, 19(4):72–82, 2012.

Y. Sui, A. Gotovos, J. Burdick, and A. Krause. Safe Exploration for Optimization with Gaussian Processes. In *Proc. of the International Conference on Machine Learning*, pages 997–1005, 2015.

J.A.K. Suykens and J. Vandewalle. Least Squares Support Vector Machine Classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

M.K. Titsias. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *Proc. of the International Conference on Artificial Intelligence and Statistics*, pages 567–574, 2009.

B. Tovar, L. Muñoz-Gómez, R. Murrieta-Cid, M. Alencastre-Miranda, R. Monroy, and S. Hutchinson. Planning Exploration Strategies for Simultaneous Localization and Mapping. *Robotics and Autonomous Systems*, 54(4):314–331, 2006.

E. G. Tsardoulias, A. Iliakopoulou, A. Kargakos, and L. Petrou. A Review of Global Path Planning Methods for Occupancy Grid Maps Regardless of Obstacle Density. *Journal of Intelligent & Robotic Systems*, 84(1-4):829–858, 2016.

M. Turchetta, F. Berkenkamp, and A. Krause. Safe Exploration in Finite Markov Decision Processes with Gaussian Processes. In *Proc. of Advances in Neural Information Processing Systems*, pages 4312–4320, 2016.

J. Vallvé and J. Andrade-Cetto. Potential Information Fields for Mobile Robot Exploration. *Robotics and Autonomous Systems*, 69:68–79, 2015.

E.A. Wan and R. Van Der Merwe. The Unscented Kalman Filter for Nonlinear Estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pages 153–158, 2000.

P. Whaite and F.P. Ferrie. Autonomous Exploration: Driven by Uncertainty. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):193–205, 1997.

C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Proc. of Advances in neural information processing systems*, pages 682–688, 2001.

B. Yamauchi. A Frontier-based Approach for Autonomous Exploration. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 146–151, 1997.

K. Yang, S. Keat Gan, and S. Sukkarieh. A Gaussian Process-Based RRT Planner for the Exploration of an Unknown and Cluttered Environment with a UAV. *Advanced Robotics*, 27(6):431–443, 2013.

M. Zucker, N. Ratliff, A.D. Dragan, M. Pivtoraiko, M. Klingensmith, C.M. Dellin, J.A. Bagnell, and S.S. Srinivasa. CHOMP: Covariant Hamiltonian Optimization for Motion Planning. *The International Journal of Robotics Research*, 32 (9-10):1164–1193, 2013.