Sequential Bayesian Optimisation for Spatial-Temporal Monitoring

Roman Marchant
NICTA & University of Sydney
Sydney, Australia
roman.marchant@sydney.edu.au

Fabio Ramos
NICTA & University of Sydney
Sydney, Australia
fabio.ramos@sydney.edu.au

Scott Sanner
NICTA & ANU
Canberra, Australia
ssanner@nicta.com.au

Abstract

Bayesian Optimisation has received considerable attention in recent years as a general methodology to find the maximum of costly-to-evaluate objective functions. Most existing BO work focuses on *where* to gather a set of samples without giving special consideration to the sampling sequence, or the costs or constraints associated with that sequence. However, in real-world sequential decision problems such as robotics, the order in which samples are gathered is paramount, especially when the robot needs to optimise a temporally non-stationary objective function. Additionally, the state of the environment and sensing platform determine the type and cost of samples that can be gathered. To address these issues, we formulate *Sequential Bayesian Optimisation* (SBO) with side-state information within a *Partially Observed Markov Decision Process* (POMDP) framework that can accommodate discrete and continuous observation spaces. We build on previous work using *Monte-Carlo Tree Search* (MCTS) and *Upper Confidence bound for Trees* (UCT) for POMDPs and extend it to work with continuous state and observation spaces. Through a series of experiments on monitoring a spatial-temporal process with a mobile robot, we show that our UCT-based SBO POMDP optimisation outperforms myopic and non-myopic alternatives.

1 INTRODUCTION

Bayesian Optimisation (BO) [1, 6, 10] is a global optimisation technique that has recently gained popularity in the machine learning community. BO possesses major advantages when used to find the maximum of partially observed objective functions that are costly to evaluate, lack gradient information, and can only be inferred indirectly from noisy observations. BO is robust to this setting because it builds a statistical model over the objective. More specifically, it places a *prior* over the space of functions and combines it with noisy samples to produce an incremental prediction for the unknown function. The prior usually takes the form of a *Gaussian Process* (GP) [15], which has proved successful in modelling spatial-temporal data [4, 9, 17]. The key component for the effectiveness of BO is the use of an *Acquisition Function* (AF) that guides the search for the optimum by selecting the locations where samples are gathered based on the posterior in each iteration.

BO can be readily applied to scenarios where the objective function does not vary in time and sampling locations can be chosen freely within the input domain. In real-world robotics applications, functions are likely to change with time [11] indicating that *when* to sample is as important as *where* to sample. Another important aspect in realistic settings is that the state of the environment and sampling platform determines the reachable space for gathering the next sample. Combined, these issues create an imperative for finding optimal *sequences* of sampling locations.

Most of the existing work focuses on myopic decision-making by evaluating one-step lookahead for objective sampling. Non-myopic solutions have been proposed in [5, 12], but the authors acknowledge they are considerably expensive to evaluate and do not account for possible side-state presence due to external conditions. An optimal solution to non-myopic decision-making with side-state can be formalised in the *Partially Observed Markov Decision Process* (POMDP) framework. The key here is to consider the state as a tuple, consisting of the unknown function and the state of a sensing robot. However, this leaves open the question of how one can efficiently solve the resulting POMDP.

The *online* setting for POMDP planning has received increased attention in recent years for helping overcome perceived efficiency limitations of POMDP solutions [16]. Silver and Veness [18] show how to use UCT for large POMDPs, however, this does not extend to continuous observations (without sampling). Porta *et al* [14] present

Point-Based Value Iteration (PBVI) for continuous state, action and observation POMDPs, however, this approach aims for a closed-form value function that generalises over all states, which can only be computed in more restricted cases than the general sequential BO POMDP framework we would like to propose in this work. A first connection between BO and POMDPs has been noted by [20], that solved the two-step lookahead without any efficient strategies, or considering the side-state as we do. In this work we intend to build on both [18] and [20] to apply UCT to a general POMDP formulation of SBO with side-state and continuous observations.

We begin by briefly describing *Gaussian Processes* (GPs), BO and POMDPs. We show the connections between SBO and POMDPs, followed by possible online solutions for multi-step lookahead in POMDPs that aim to provide an optimal sequence of sampling locations. In section 4 we evaluate our model for spatial-temporal monitoring problems that clearly demonstrate the benefits of our UCT algorithm for non-myopic SBO optimisation.

2 BACKGROUND

We start with a brief description of Gaussian processes as the underlying regression technique for Bayesian optimisation. We then describe BO and define notation for our POMDP formulation.

2.1 GAUSSIAN PROCESSES

A GP is a collection of random variables with a joint Gaussian distribution. A GP places a Gaussian prior over the space of functions and is completely defined by a mean function, $m(\mathbf{x})$, and a positive semi-definite covariance function $k(\mathbf{x}, \mathbf{x}')$, where $\mathbf{x}$ is an input in a D dimensional space, $\mathbf{x} \in \mathcal{R}^D$. A latent noisy function f can be represented as $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$. Further, we assume an additive noise model $y = f(\mathbf{x}_i) + \epsilon$ for noisy observations y from f, where $\epsilon \overset{iid}{\sim} \mathcal{N}\left(0, \sigma_n^2\right)$ is an independent Gaussian noise.

Given a set of N training inputs $X = \{\mathbf{x}_i\}_{i=1}^N$ and corresponding outputs $\mathbf{y} = \{y_i\}_{i=1}^N$ we can calculate the predictive distribution of f at an unknown query location $\mathbf{x}^\star$ by computing the posterior $p(f(\mathbf{x}^\star) | \mathbf{y}, X, \mathbf{x}^\star)$. For a GP, this predictive distribution is Gaussian, $f(\mathbf{x}^\star) \sim \mathcal{N}\left(\bar{f}^\star, \mathrm{cov}(f^\star)\right)$, where

$$\begin{aligned} \bar{f}^\star &= K(\mathbf{x}^\star, X) K_X^{-1} (\mathbf{y} - m(X)) \, , \\ \mathrm{cov}(f^\star) &= K(\mathbf{x}^\star, \mathbf{x}^\star) - K(\mathbf{x}^\star, X) K_X^{-1} K(X, \mathbf{x}^\star) \, , \end{aligned} \tag{1}$$

and $K(A, B)$ is a covariance matrix whose element (i, j) is calculated as $k_{i,j} = k\left(\mathbf{x}_i, \mathbf{x}_j\right)$, with $\mathbf{x}_i \in A$ and $\mathbf{x}_j \in B$. $K_X = K(X, X) + \sigma_n^2 I$ is the covariance matrix between observations, with identity matrix I. The parameters of the mean and covariance functions can be estimated automatically by maximising the marginal likelihood of the data [15]. Since we will be dealing with space-time inputs, $\mathbf{x}$ can be represented by space and time components, covariance functions can be separable [19] and learn periodic patterns [15, 21].

2.2 BAYESIAN OPTIMISATION

BO is an optimisation technique for finding the optimum $\widehat{\mathbf{x}} \in \mathbb{R}^D$ of an unknown, costly to evaluate and noisy function $f : \mathbb{R}^D \to \mathbb{R}$,

$$\widehat{\mathbf{x}} = \arg\max_{\mathbf{x}} f(\mathbf{x}). \tag{2}$$

In this setting f is not directly observable but we have available noisy samples from f, i.e. the ith observation can be seen as $y_i = f(\mathbf{x}_i) + \epsilon$, where $\epsilon \overset{iid}{\sim} \mathcal{N}\left(0, \sigma_n^2\right)$ is the noise associated to each independent observation. BO uses a GP to model f which is incrementally updated at every iteration, as new observations become available. The benefit of this approach is that for every optimum candidate location we can evaluate an analytical expression for the expected value of f and its variance (Equation 1). This information is used by an *Acquisition Function* (AF), $h(\mathbf{x})$, whose purpose is to guide the search for the optimum. At each iteration, one sample is gathered from f at a location selected by maximising $h(\mathbf{x})$, which is a simpler and faster optimisation procedure (compared to the original problem of optimising f). Algorithm 1 presents the BO algorithm.

Algorithm 1 Bayesian Optimisation

```
Inputs: $f, h$
Outputs: $\widehat{\mathbf{x}}$, $f(\widehat{\mathbf{x}})$
  for $j = 1, 2, 3, \ldots$ {Max iterations} do
      Find $\mathbf{x}_j = \arg\max_{\mathbf{x}} h(\mathbf{x})$
      $y_j \leftarrow f(\mathbf{x}_j)$                    ▷ Gather sample from $f$
      Augment training set with $(\mathbf{x}_j, y_j)$.
      Update GP
      if $y_j > \mu(\widehat{\mathbf{x}})$ then
          $\widehat{\mathbf{x}} \leftarrow x_j$              ▷ Update location of optimum
      end if
  end for
```

Many AFs have been proposed on the literature [6–8, 12]. In this paper we use the *Upper Confidence Bound* (UCB) [3] acquisition function, however, none of the algorithms presented here are strongly linked to this specific AF.

2.3 PARTIALLY OBSERVABLE MDPs

POMDPs are a unified framework for sequential decision making under uncertainty when the state is not directly observable. A POMDP is fully represented by the following tuple $\langle S, A, T, R, Z, O\rangle$, where:

- S : Set of states $\{s_1, s_2, \ldots, s_n\}$.
- A : Set of actions $\{a_1, a_2, \ldots, a_n\}$.
- $T : S \times A \times S \rightarrow [0, 1]$ is a transition function that represents the probability of transition between states s and s' when executing action a, i.e. $T(s, a, s') = p(s'|s, a)$.
- $R : S \times A \rightarrow \mathbb{R}$ is a reward function that encodes the reward of executing action a on state s, i.e. $R(s, a)$.
- Z : Finite set of observations $\{z_1, z_2, \ldots, z_n\}$.
- $O : S \times A \times Z \rightarrow [0, 1]$ is a observation function that represents the probability of observing o if action a is executed with resulting state s, i.e. $O(o, a, s) = p(o|a, s)$.

In POMDPs, it is well-known that a belief state summarises *all* relevant information in the observation history of a POMDP. Given a belief state $b_{t-1}(s)$, the belief at time t can be updated as:

$$b_t(s') \propto P(o|s') \int b_{t-1}(s)P(s'|s, a)ds. \tag{3}$$

where $b_0(s) = p(s)$ represents the initial belief state.

Solving a POMDP is equivalent to determining a policy $\pi^\star$ mapping belief states to actions which maximizes some objective criterion. An optimal policy over an infinite horizon can be found by maximising the expected cumulative discounted reward r_t (for discount $\gamma \in (0, 1]$) at time step t when executing π starting from belief state $b_0 := b_0(s)$,

$$\pi^\star = \arg\max_\pi E\left[\sum_{t=0}^{\infty} \gamma^t \cdot r_t^\pi | b_0\right]. \tag{4}$$

3 SEQUENTIAL BAYESIAN OPTIMISATION

With the definitions above we can now extend BO to a sequential setting. In order to apply BO to more realistic problems we expand the existing theory to a more generic framework and include the notion of state in the definition of the problem. This means that at every step a generic reward, r, can be obtained by sampling at $\mathbf{x}$. This reward depends on the state $\mathbf{x}$ of a mobile robotic sensor and the expected value of the objective function $f(\mathbf{x})$. In the general case, because gathering each sample has an associated reward, the order in which they are gathered has a direct influence over the total accumulated reward for a specific lookahead. We call this kind of optimisation technique *Sequential Bayesian Optimisation* (SBO).

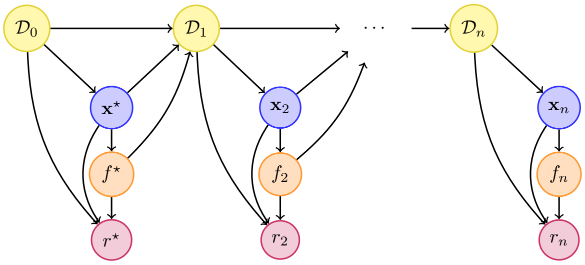


Figure 1: Bayesian network representation for SBO.

Sampling locations and their associated observations are grouped in $\mathcal{D}$, which is built incrementally as shown in Figure 1. Using a similar treatment to plain BO, the myopic expectation of the reward r (ER), can be obtained by marginalising out all unknown outcomes,

$$\mathrm{ER}(\mathbf{x}^\star|\mathcal{D}_0) = \mathbb{E}_{f^\star}\left[r(\mathbf{x}^\star, f^\star|\mathcal{D}_0)\right] \tag{5}$$

$$= \int r(\mathbf{x}^\star, f^\star|\mathcal{D}_0)p(f^\star|\mathbf{x}^\star, \mathcal{D}_0)\mathrm{d}f^\star \ . \tag{6}$$

The n-step lookahead expression is given by

$$\begin{aligned}&\mathrm{ER}_n(\mathbf{x}^\star|\mathcal{D}_0)\\ &= \int \cdots \int \left(r(\mathbf{x}^\star, f^\star|\mathcal{D}_0) + \sum_{i=2}^{n} \left(r(\mathbf{x}_i, f_i|\mathcal{D}_{i-1})\right)\right)\\ &\qquad p(f^\star|\mathbf{x}^\star, \mathcal{D}_0) \times \prod_{i=2}^{n} p(f_i|\mathbf{x}_i)p(\mathbf{x}_i|\mathcal{D}_{i-1})\\ &\qquad \mathrm{d}f^\star \mathrm{d}f_2 \cdots \mathrm{d}f_n \mathrm{d}\mathbf{x}_2 \cdots \mathrm{d}\mathbf{x}_n \ ,\end{aligned} \tag{7}$$

where we are marginalising out all future outcomes $(f^\star, f_2 \ldots f_n)$ and locations $(\mathbf{x}_2 \ldots \mathbf{x}_n)$. This expression has been derived in [12], however, it presents a slight modification because we are considering the whole sequence of locations for reward calculation, not just the expected improvement for the last sample. It is important to note that within the BO algorithm, ER can be seen as the acquisition function $h(\mathbf{x})$ for selecting sampling locations. ER needs to be maximised w.r.t. $\mathbf{x}^\star$ in each iteration of the algorithm.

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

$$\begin{aligned}&\mathrm{ER}_n(\Theta^\star,\mathcal{D}_0)\\&=\int_{f^\star}\int_{f_2}\cdots\int_{f_n}\int_{\Theta_2}\cdots\int_{\Theta_n}\\&\qquad\left(r(\mathcal{C}_{\Theta^\star}|\mathcal{D}_{N-1})+\sum_{i=2}^{n}r(\mathcal{C}_{\Theta_i}|\mathcal{D}_{i-1})\right)\\&\qquad p(f^\star|\Theta^\star,\mathcal{D}_{N-1})\prod_{i=2}^{n}p(f_i|\Theta_i,\mathcal{D}_{i-1})p(\Theta_i|\mathcal{D}_{i-1})\\&\qquad\qquad\mathrm{d}f^\star\mathrm{d}f_2\cdots\mathrm{d}f_n\mathrm{d}\Theta_2\cdots\mathrm{d}\Theta_n\end{aligned}\tag{8}$$

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

3.1 SBO AS ONLINE POMDPs

Our SBO formulation is *state-aware*, i.e. it considers the state of a mobile robot for decision making. This problem can be formulated as a POMDP problem in a similar manner as described in [20] for regular BO. The main idea is to include the objective function, which is partially observable, together with the state of the robot, into the state definition. We assume the robot's pose is fully observable and part of the state as side information $\mathbf{p}$. The decision of where to sample f is encoded by the action space, which is limited by the possible actions that can be performed by the robot. In the discrete case, an action is represented by moving to a specific cell. For the continuous case an action means travelling along a continuous path. More formally, the elements of the POMDP definition for side-state SBO are:

- S : The state which is a tuple $\{f,\mathbf{p}\}$, where f is a latent (not directly observable) function defined over space and time representing the unknown process. Additionally, we include the state of the sensing robot, $\mathbf{p}$, which is fully observable, as the side information.
- A : The parametrised action space $a\left(\Theta\right)$. The actions can be described as move according to parameters Θ and gather a samples from f in the process. For the discrete sampling case, Θ represents a location in the spatial domain of f. For the continuous case, Θ are the parameters of a continuous curve defined over the domain of f.
- T : The transition function which is defined over the entire state $\{f,\mathbf{p}\}$. $T(\{f,\mathbf{p}\},a\left(\Theta\right),\{f',\mathbf{p}'\})$ is the transition probability of resulting in state $\{f',\mathbf{p}'\}$ given that action $a\left(\Theta\right)$ was taken at state $\{f,\mathbf{p}\}$. Assuming that the robot does not affect or change the objective function, the joint transition probability can be decomposed into the product of two independent transition functions:

$$\begin{aligned}&T(\{f,\mathbf{p}\},a(\Theta),\{f',\mathbf{p}'\})\\&\quad=T_f(f,a(\Theta),f')T_{\mathbf{p}}(\mathbf{p},a(\Theta),\mathbf{p}')\end{aligned}\tag{9}$$

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

$$T_f(f,a(\Theta),f')=\delta(f'-f).\tag{10}$$

The transition function $T_{\mathbf{p}}$ depends on the definition of the action space, and can often be modeled deterministically since robots can navigate with accurate positioning and path following controllers in many large-scale outdoor monitoring applications. When the action space is defined as a location, the action parameters Θ represent a location, and $T_{\mathbf{p}}$ can be calculated using

$$T_{\mathbf{p}}(\mathbf{p},a(\Theta),\mathbf{p}')=\delta(\mathbf{p}'-\Theta)\tag{11}$$

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

$$\mathrm{ER}(\{f,\mathbf{p}\},a(\Theta))=\mathbb{E}_{b(f)}[f(\mathbf{x})]+cost(\mathbf{p},a(\Theta))\tag{12}$$

When the action space is parametrised as locations the reward can be evaluated directly. However, when the action space is parametrised by curves, the reward associated to an action is given by the sum of the rewards along the curve $\mathcal{C}$:

$$R(\{f,\mathbf{p}\},\mathcal{C}(\Theta))=\sum_{\mathbf{x}\in\mathcal{C}(\Theta)}R(\{f,\mathbf{p}\},\mathcal{C}(\Theta))\ ,\tag{13}$$

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

- Z : In SBO, objective observations $z\in\mathbb{R}$ are simply noisy observations of $f(\Theta)$ as defined next.
- O : The observation function is defined according to the action space parametrisation. When the action space is defined as a sampling location Θ, f can be evaluated directly on Θ.

$$O(z,a(\Theta),\{f,\mathbf{p}\})=p(z|f(\mathbf{x}=\Theta))\tag{14}$$

We observe that for GPs, we can generate z by sampling from a GP marginal for f at location Θ. When the action space is a curve $\mathcal{C}$, f is evaluated at a number of sample locations within $\mathcal{C}$. The observation function for this set of observations $\{z_i\}$ is

$$O(\{z_i\},\mathcal{C}(\Theta),\{f,\mathbf{p}\})=\prod_{\mathbf{x}_i\in\mathcal{C}(\Theta)}p(z_i|f(\mathbf{x}_i))\tag{15}$$

The belief is then the probability distribution over the space of functions f and updated as described in equation (3). If the model for f is a GP, the belief update for an action-observation pair can be computed directly. The action component can be ignored for purposes of updating a belief in f, since as stated earlier, the robot's physical state does not affect or change the objective function, it only restricts the observations that can be made regarding f. Therefore, the belief update over f is simply computed by adding new location-observation pairs to the GP training data set.

Next, we present a methodology to solve this POMDP by sampling a subset of action primitives that the robot can execute in the environment. Action primitives and maximum likelihood observation selection are the key points to approximate Equation 8.

3.2 MCTS AND UCT FOR SBO

MCTS is a popular technique for solving large POMDPs [2, 18]. This method can turn a tedious search in decision trees into an efficient approximation using Monte-Carlo samples from the tree. MCTS efficiently searches reachable beliefs from a given initial belief state and is useful for real-time online planning. Its main advantage over other techniques, such as Point-Based Value Iteration is that it does not require the overhead of maintaining alpha-functions over all states nor choosing the states for which alpha-functions should be maintained.

[18] have shown how MCTS can reach impressive scalability through the use of UCT, which they call POMCP. In this work we conserve their idea of efficient tree search. However, we consider the case where the belief update is a GP update for f and use the maximum likelihood observation, as it is done by [13]. The maximum-likelihood observation assumption helps reducing the branching factor of the tree, which would grow uncontrollably when sampling observations.

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

The tree is built incrementally starting with an initial node v_0. Figure 2 shows an example of a small tree that has been expanded partially with two action primitives. Each ellipse represents a node, that consists of a belief over f, $b(f)$, and side-state $\mathbf{p}$. A node is expanded by simulating the outcomes of executing an action. The outcomes (noisy observations of f) are the maximum-likelihood observations. The branching factor of the tree will be the number of action primitives. When a node is expanded, a new node is created using the updated belief and new side-state.

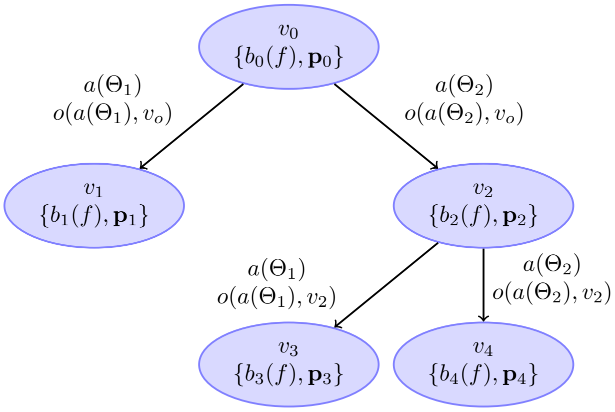


Figure 2: Example of a tree with depth 2, partially expanded from a set of two action primitives.

The first step in each iteration is to find a leaf node candidate for expansion/evaluation, which is done inside of the function TREEPOLICY. This search is guided by the function BESTCHILD, which uses the statistics stored for each node (accumulated reward and number of visitations) to select the most *promising* child. Starting from the chosen leaf node, a random action selection is conducted until the maximum depth is reached, executed within DEFAULTPOLICY. The total accumulated reward is then backed up in function BACKUP, that updates the statistics on all the nodes visited during the current iteration. Each iteration of the search algorithm simulates a sequence of up to n actions, where n is the maximum depth. When the iteration loop is completed, the best action is determined by picking the best child from the parent node v_0. Algorithm 2 shows the full procedure for building a tree and returning the best immediate action.

4 EXPERIMENTS

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

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

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

In these experiments, the pose $\mathbf{p} = (x_{1r}, x_{2r}, \theta_r)$ of a robot is the side-state for the SBO formulation and f is the unknown function to be estimated. The belief $b(f)$ is represented by a GP using a separable space-time covariance function [19]. The structure of the GP's covariance function can capture periodicity in f from the training data.

Algorithm 2 Monte Carlo Tree Search for SBO

```
function $a^\star$ = MCTS($b(f), \mathbf{p}$, depth$_{max}$)
    $v_0 = $ NEWNODE($b(f), \mathbf{p}$, reward$_{min}$)
    $i \leftarrow 0$
    while $i <$ {Max MCTS iterations} do
        $v_l \leftarrow$ TREEPOLICY($v_0$)
        $r \leftarrow$ DEFAULTPOLICY($v_l$)
        BACKUP($v_l, r$)
    end while
    return $a^\star$ = BESTCHILD($v_0$)
end function
function $v_l =$ TREEPOLICY($a$)
    $v \leftarrow v_0$
    while DEPTH($v$) $\leq$ depth$_{max}$ do
        if $v$ has untried actions then
            Choose $a$ from untried actions
            $r \leftarrow$ Simulate $a$                ▷ Simulate Reward
            Update $b(f)$ and $\mathbf{p}$.
            return $v_l$ =NEWNODE($b(f), \mathbf{p}, r$)
        else
            $v =$ BESTCHILD($v$)
        end if
    end while
    return $v$
end function
function $r =$ DEFAULTPOLICY($v_l$)
    $r \leftarrow$ Get reward accumulated until $v_l$
    $d \leftarrow$ DEPTH($v_l$)
    while $d \leq$ depth$_{max}$ do
        Select $a$ randomly
        Update $b(f)$ and $\mathbf{p}$.
        $r_a \leftarrow$ Simulate $a$
        $r \leftarrow r + r_a$
        $d \leftarrow d + 1$
    end while
end function
function BACKUP($v_l, r$)
    $v \leftarrow v_l$
    while $v \neq v_0$ do
        Increase visited counter for $v$
        Increase accumulated reward for $v$
        $v \leftarrow$ PARENT($v$)
    end while
end function
function $v_c =$ BESTCHILD($v_p$)
    $V \leftarrow$ Children of $v_p$
    for $v_i \in V$ do
        $N_p \leftarrow$ Visited counter of $v_p$
        $N_i \leftarrow$ Visited counter of $v_i$
        $R_i \leftarrow$ Accumulated reward
        $g(i) = \frac{R_i}{N_i} + \kappa_{MC}\sqrt{\frac{2\ln(N_p)}{N_i}}$
    end for
    $v_c \leftarrow \arg\max_{v_i \in V} g(i)$
end function
```

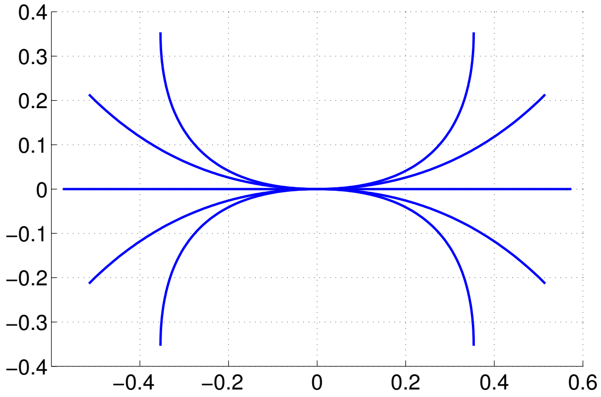


Figure 3: Motion primitives for a mobile robot. Axis in km.

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

The action space A is determined by a set of motion primitives parametrised as 2D cubic splines. A cubic spline $\mathcal{C}$ is a continuous function mapping from $\mathbb{R}$ to $\mathbb{R}^2$, $\mathcal{C}(u|\mathbf{\Theta}) = [\mathcal{C}_{x_1}\ \mathcal{C}_{x_2}]^T$, with $u \in [0,1]$, defined as

$$\mathcal{C} = \mathbf{\Theta} \left[\begin{array}{cccc} u^3 & u^2 & u^1 & 1 \end{array}\right]^T , \qquad (16)$$

where $\mathbf{\Theta}$ are the parameters expressed as a 2×4 matrix for the 2D case. With appropriate parametrisation, the curves generate the ten primitives $A = \{\mathcal{C}_i\}_{i=1\ldots10}$ shown in Figure 3 for $\mathbf{p} = \mathbf{p}_0 = (0,0,0)$. For values of $\mathbf{p} = (x_{1r}, x_{2r}, \theta_r)$ the curves are rotated and translated using translation and rotation matrices. We define a transition function $T_{\mathbf{p}}(\mathbf{p}, \mathcal{C}_i, \mathbf{p}') = 1$ for a cubic spline transformed from $\mathbf{p}$, with

$$\mathbf{p}' = \left(\mathcal{C}_i(u=1)_{x_1}, \mathcal{C}_i(u=1)_{x_2}, \left.\frac{\partial \mathcal{C}_{x_1}/\partial u}{\partial \mathcal{C}_{x_2}/\partial u}\right|_{u=1} \right) . \qquad (17)$$

Before an action (curve) is selected for execution, the robot computes the optimal action using the MCTS algorithm (Algorithm 2). The robot gathers noisy samples from f along $\mathcal{C}$ while the action is being executed.

4.1 STATIC FUNCTION

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

$$\begin{aligned} y = f(x_1, x_2, t) = \quad & e^{-(x_1-4)^2} e^{-(x_2-1)^2} \\ & + 0.8 e^{-(x_1-1)^2} e^{-\left(\frac{x_2-4}{2.5}\right)^2} \\ & + 4 e^{-\left(\frac{x_1-10}{5}\right)^2} e^{-\left(\frac{x_2-10}{5}\right)^2} , \end{aligned} \qquad (18)$$

where $x_1 \in [0,5]$, $x_2 \in [0,5]$, and $t \in [0,\infty]$. Figure 4 shows a plot for this function, where it is easy to distinguish two main peaks with different amplitudes. The robot is initially located at pose $\mathbf{p} = (0.5, 0.5, 0)$ and travels at a fixed speed of $0.2m/s$, gathering a sample every 5 minutes.

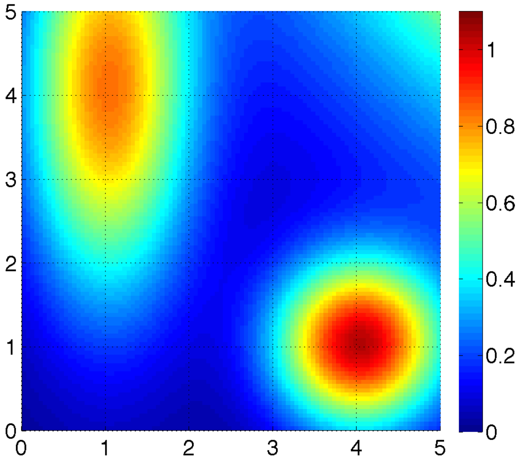
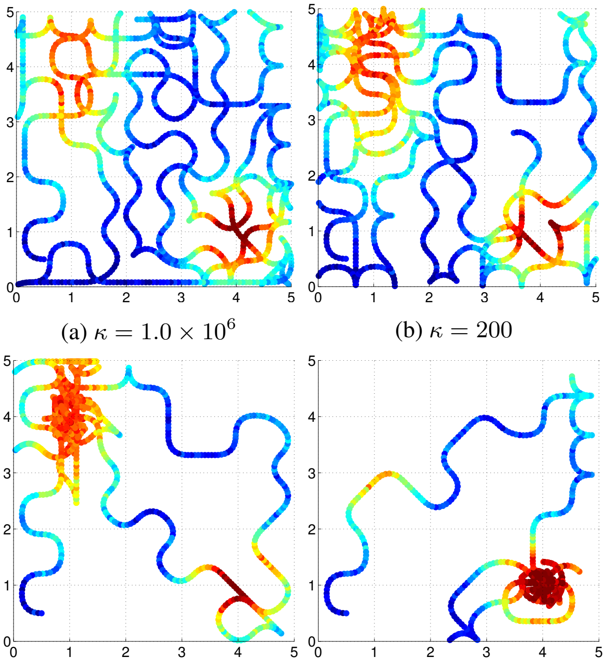


Figure 4: Static goal function. Axis in km.

We first want to evaluate how the definition of the reward function R within the POMDP context impacts the action selection properties of the algorithm. We compare four different reward functions based on the UCB acquisition function, $r(\mathbf{x}) = \mu(\mathbf{x}) + \kappa\sigma(\mathbf{x})$, where $\kappa_i \in \{1.0 \times 10^6, 200, 20, 10\}$. It is a well known fact that the value of κ affects the exploration-exploitation trade off and this is clearly reflected in the resulting paths followed by each robot, as shown in Figure 5. The most explorative path sequence corresponds to $\kappa = 1.0 \times 10^6$ (Figure 5a) and the least explorative is $\kappa = 10$ (Figure 5d). Between these two extremes there are intermediate solutions where exploitation is favoured more strongly for lower values of κ.

In the next experiment we focus on the depth of the action selection search, i.e. the number n of lookahead steps for decision making. This corresponds to the maximum depth allowed in the search tree. We first evaluated the entire decision tree, which means simulating all the possible combinations of actions. This approach, which we call *Full Tree* (FT) strategy, will need $|A|^n$ simulations which becomes impractical quickly. In fact, for this paper we only consider FT strategies with $n \leq 3$. We compare the performance of FT against MCTS (Algorithm 2) where the number of simulations is a parameter. Clearly, for a same depth, MCTS is bounded by FT, however MCTS can find near-optimal solutions much faster. For this reason we were able to experiment with depths up to $n \leq 5$. We compare six different combinations of depth and algorithm type as indicated in Table 1.

The reward function used for these simulations was $r(\mathbf{x}) = \mu(\mathbf{x}) + 1.0 \times 10^6\sigma(\mathbf{x})$ for all cases. Therefore, the only difference in action selection is due to the number of lookahead steps. Figure 6 shows the paths followed by the robot at $t = 2.3$ days, when it had already gathered 616 samples from f. This figure does not show all cases, only the four most relevant ones. It is interesting to observe that



(a) $\kappa = 1.0 \times 10^6$
(b) $\kappa = 200$
(c) $\kappa = 20$
(d) $\kappa = 10$

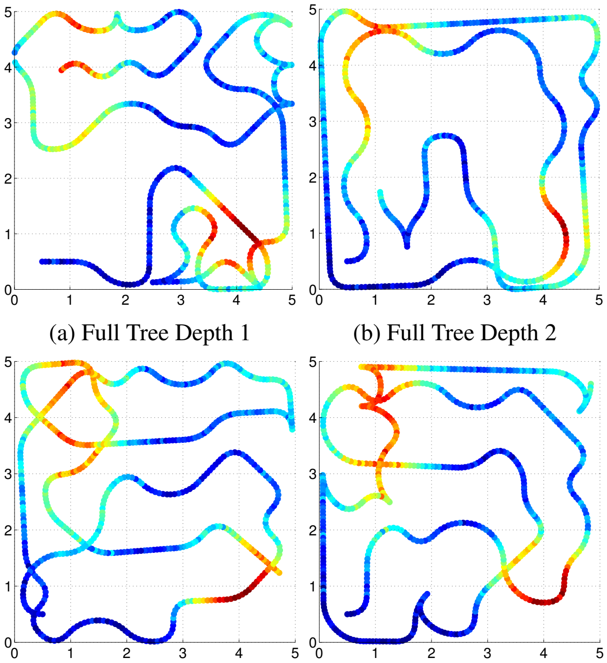
Figure 5: Comparison of followed paths for purely exploration behaviour using Full Tree and MCTS-UCT. Axis in km. Colour scale represents the value of sampled values.

Table 1: Experiment for Depth and Algorithm Type Comparison

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

the search with FT Depth 1 (Figure 6a) has not achieved a full coverage of the area and is highly susceptible to get trapped and collide into the edges of the domain, which is clearly sub-optimal from an exploration point of view. On the other hand, the FT Depth 2 shows increased coverage capability, which is improved further for deeper search strategies. FT Depth 3 and MCTS Depth 3 show similar result, with the clear advantage that MCTS requires only 10% of the number of simulations of FT.

We also compare the accumulated reward over time for each case in Figure 7. This illustrates the advantage of using a multi-step lookahead strategy in increasing the total accumulated reward. However, it is not clear the advantage of using higher depths than two, as they do not show a clear improvement in accumulated reward. The main rea-



(a) Full Tree Depth 1 (b) Full Tree Depth 2

(c) Full Tree Depth 3 (d) MCTS Depth 3 N° Iterations 100

Figure 6: Comparison of paths for purely exploration behaviour using FT and MCTS-UCT. Axis in km. Colour scale represents the value of samples.

son behind this is that f does not change over time, thus making the problem simple enough such that any strategy with depth greater than 1 would be very close to the optimal solution.

4.2 DYNAMIC FUNCTION

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

$$y = f(x_1, x_2, t) = e^{-\left(\frac{x_1-2-f_1(t)}{0.7}\right)^2} e^{-\left(\frac{x_2-2-f_2(t)}{0.7}\right)^2}, \tag{19}$$

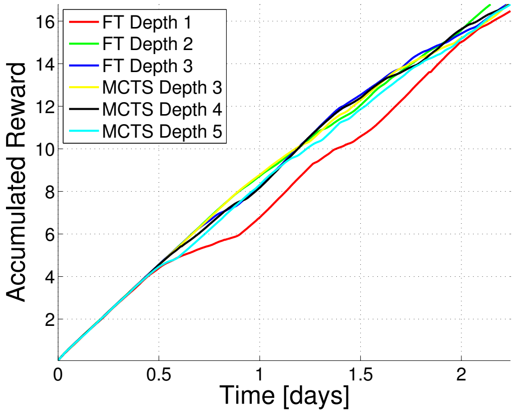
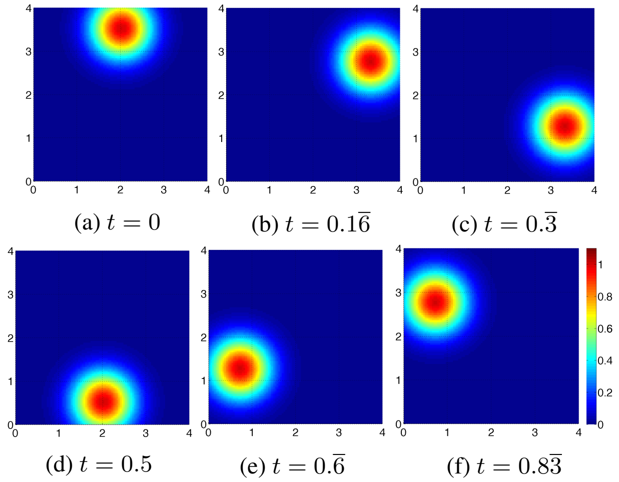


Figure 7: Accumulated reward for static goal function.



(a) $t = 0$ (b) $t = 0.1\overline{6}$ (c) $t = 0.\overline{3}$

(d) $t = 0.5$ (e) $t = 0.\overline{6}$ (f) $t = 0.8\overline{3}$

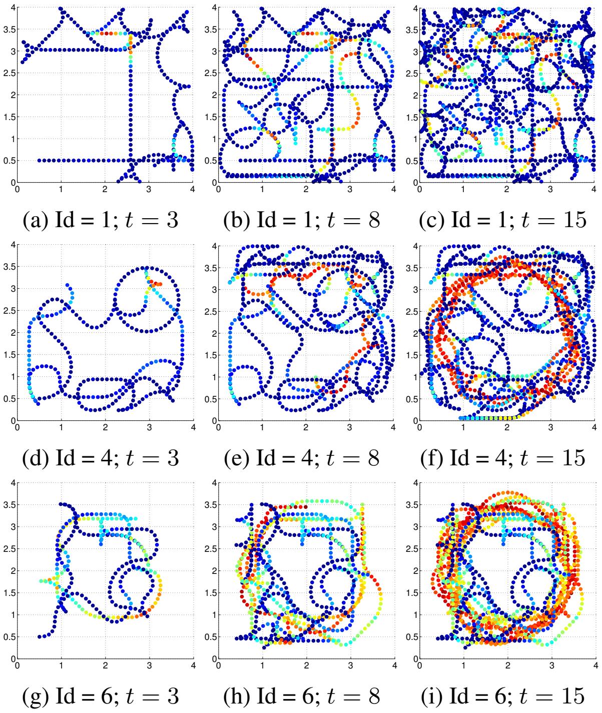
Figure 8: Dynamic goal function within one period. Axis in km.

with $f_1(t) = 1.5\sin(2\pi t)$, $f_2(t) = 1.5\cos(2\pi t)$, $x_1 \in [0,5]$, $x_2 \in [0,5]$, and $t \in [0,\infty]$. This expression generates a function where the peak moves over time. The peak circles clockwise around $(x_1, x_2) = (2,2)$ periodically, with a period of 1 day. The motivation for this example comes from air pollution monitoring tasks where we are interested in following peaks of pollution through time while learning how the entire process evolves in space-time. Figure 8 shows the goal function for 6 time steps within one period.

Similarly to the previous experiment, the robot is initially located at pose $\mathbf{p} = (0.5, 0.5, 0)$, travels at a fixed speed of $0.12m/s$ and gathers a sample every 15 minutes. The goal in this experiment is to find and follow the maximum of f over time. Therefore, we select the reward function $r(\mathbf{x}) = \mu(\mathbf{x}) + 10\sigma(\mathbf{x})$, which according to Figure 5, should generate paths concentrated over the maximum values of f. Ideally, the robot should learn to follow the peak through time which would be possible for speeds greater than 0.109m/s. We try the same set of depth-algorithm pairs as in Section 4.1 and detailed in Table 1. We only show results for the extreme cases with the purpose of avoiding clutter in the figures.

Figure 9 illustrates the advantage of using multi-step lookahead strategies. The first row shows paths for FT Depth 1, where it can be seen that the robot does not learn how to follow the peak around a circle within 15 days. The second row, MCTS Depth 2, which only does 15 more simulations per iteration than FT Depth 1, the robot is already able to learn the circular pattern at $t = 12$ days. With deeper search strategies, the time required to learn the pattern decreases significantly indicating a better exploration and exploitation solution. In fact, for MCTS Depth 5 the pattern is learnt in $t = 8$ days.

Figure 10 shows the benefits of using non-myopic strategies for action selection. The cumulative reward is clearly



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

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

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

Figure 9: Comparison of followed paths for FT and MCTS-UCT in a dynamic function. First row shows the paths for FT, Depth 1; Second row shows the paths for MCTS, Depth 2; Third row shows the paths for MCTS, Depth 5. Colour scale represents the value of samples.

larger for multi-step lookahead decision making algorithms. The best solution is MCTS Depth 5, that is clearly superior for the entire duration of the simulation. A steeper slope for accumulated reward indicates that a method has learnt how to follow the peak. Then from this plot it is also clear that FT Depth 1 is not able to capture space-time dependencies properly.

It is important also to compare FT Depth 2 with MCTS Depth 2. The fact that FT is an upper bound for MCTS Depth 2 can be confirmed from Figure 10. In addition, it can be seen that both strategies accumulate similar rewards, which is a good indication that MCTS will approximate the FT solution, even with only 25% of the total tree.

Finally, Figure 11 shows how MCTS prioritises the search over promising paths. The pose of the robot at this instant is $\mathbf{p} = (1.5, 3, 0)$. Red paths are result of the function DefaultPolicy that did not get further explored and blue paths are the paths present in the tree. In can be seen how the tree automatically grows towards potentially informative areas, i.e. where the reward is higher. The green curve is the best branch of the tree.

5 CONCLUSION

In this paper we proposed formulating the sequential BO problem as a POMDP. Our main contribution was to determine a non-myopic decision making solution that maximises reward and takes into account the belief of an unknown space time process and the state of a mobile robot acting as a sensor. We formulated the solution for the POMDP analogue of SBO using a modified version of the UCT algorithm for MCTS, which is a scalable and efficient way of finding asymptotically optimal decisions.

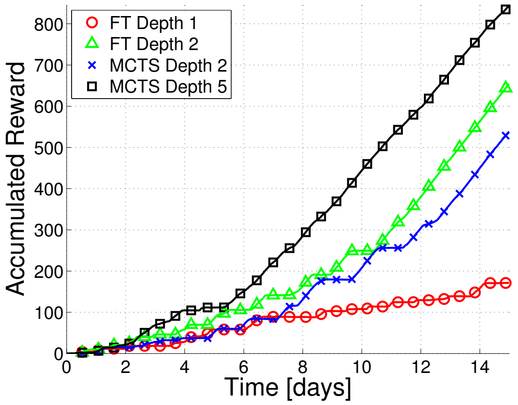


Figure 10: Accumulated reward for dynamic goal function.

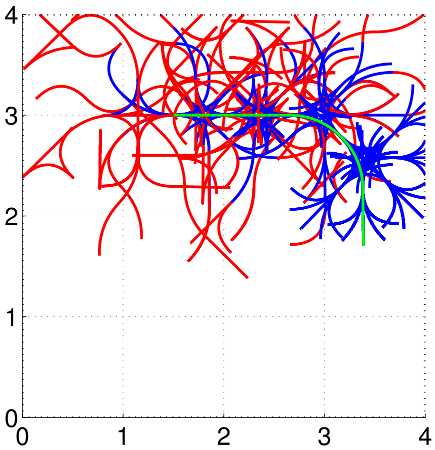


Figure 11: Example of tree built for MCTS Depth 5.

We demonstrate empirically the advantage of using non-myopic planning solutions, which becomes especially important when the objective function dynamically changes over time.

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

References

[1] 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. Technical Report arXiv:1012.2599, University of British Columbia, 2010.

[2] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

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

[4] N. Cressie and C. K. Wikle. *Statistics for Spatio-Temporal Data*. Wiley, 2011.

[5] D. Ginsbourger, R. L. Riche, and L. Carraro. A multi-points criterion for deterministic parallel global optimization based on gaussian processes. HAL: hal-00260579, 2008.

[6] M. Hoffman, E. Brochu, and N. de Freitas. Portfolio allocation for bayesian optimization. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2011.

[7] D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, pages 345–383, 2001.

[8] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimisation of expensive black-box functions. *Journal of Global Optimization*, pages 455–492, 1998.

[9] F. Lindgren, R. Håvard, and J. Lindström. An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2011.

[10] D. J. Lizotte. *Practical Bayesian Optimization*. PhD thesis, University of Alberta, 2008.

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

[12] M. A. Osborne, R. Garnett, and S. Roberts. Gaussian processes for global optimization. In *International Conference on Learning and Intelligent Optimization (LION)*, 2009.

[13] R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Perez. Belief Space Planning Assuming Maximum Likelihood Observations. In *Robotics Science and Systems (RSS)*, 2010.

[14] J. M. Porta, N. Vlassis, T. S. Matthijs, and P. Poupart. Point-Based Value Iteration for Continous POMDPs. *Journal of Machine Learning Research*, 2006.

[15] C. E. Rasmussen and C. Williams. *Gaussian processes for machine learning*. The MIT Press, Cambridge, Massachuset, 2006.

[16] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa. Online Planning Algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 2008.

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

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

[19] A. Singh, F. Ramos, H. D. Whyte, and W. J. Kaiser. Modeling and decision making in spatio-temporal processes for environmental surveillance. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2010. ISBN 978-1-4244-5038-1.

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

[21] A. G. Wilson and R. P. Adams. Gaussian Process Kernels for Pattern Discovery and Extrapolation. In *International Conference on Machine Learning (ICML)*, 2013.