Robotics and Autonomous Systems 83 (2016) 15–31

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Planning for robotic exploration based on forward simulation

Mikko Lauri *, Risto Ritala

Department of Automation Science and Engineering, Tampere University of Technology, P.O. Box 692, 33101 Tampere, Finland

HIGHLIGHTS

- Exploration formulated and solved as partially observable Markov decision process.
- New sampling-based approximation for mutual information in mobile robotics.
- Efficient algorithm for drawing samples for forward-simulation based planning.
- Experimental validation in simulated and real-world exploration domains.
- Software available at https://goo.gl/ENGkIf.

ARTICLE INFO

Article history:
Received 29 October 2015
Accepted 14 June 2016
Available online 23 June 2016

Keywords:
Partially observable Markov decision process
Active sensing
Robotic exploration
Mutual information
Sensor management

ABSTRACT

We address the problem of controlling a mobile robot to explore a partially known environment. The robot's objective is the maximization of the amount of information collected about the environment. We formulate the problem as a partially observable Markov decision process (POMDP) with an information-theoretic objective function, and solve it applying forward simulation algorithms with an open-loop approximation. We present a new sample-based approximation for mutual information useful in mobile robotics. The approximation can be seamlessly integrated with forward simulation planning algorithms. We investigate the usefulness of POMDP based planning for exploration, and to alleviate some of its weaknesses propose a combination with frontier based exploration. Experimental results in simulated and real environments show that, depending on the environment, applying POMDP based planning for exploration can improve performance over frontier exploration.

1. Introduction

Autonomous robotic agents performing tasks such as monitoring, surveillance or exploration must be able to plan their future information-gathering actions. Real-world environments are typically partially observable and stochastic, and planning in them requires reasoning over uncertain outcomes in the presence of sensor noise. The true state of the system is hidden, and knowledge about the state is represented by a belief state, a probability density function (pdf) over the true state. The utility of actions is measured by an appropriate reward function, and the agent's objective is to maximize the sum of expected rewards over a specified horizon of time. Such planning problems are instances of partially observable Markov decision processes [1], or POMDPs.

The solution of a POMDP is a control policy, i.e. a mapping from belief states to actions. To find policies for information-gathering and exploration tasks, several authors have proposed applying quantities such as entropy or mutual information as reward functions of the POMDP [2–6]. Although finding optimal policies for POMDPs is computationally hard (PSPACE-complete; [7]), they remain an attractive modelling choice due to the ability to simultaneously handle uncertainties in the robot's action and sensing outcomes.

In this article, we address the problem of finding control policies for robotic exploration problems formulated as POMDPs. We apply forward simulation algorithms for finding a solution to an open loop approximation of a POMDP. This approach allows general belief-dependent reward functions and with suitable choice of algorithm can avoid discretization of the continuous planning space. We derive a sampling-based approximation for mutual information that can be applied in conjunction with forward simulation based planning, and describe a method for efficiently drawing the required samples. We provide an empirical evaluation of our proposed approach in simulated and real-world exploration experiments.

This article is organized as follows. Section 2 provides a survey of related work and discusses the relation of our contribution to

* Corresponding author.
E-mail addresses: mikko.lauri@tut.fi (M. Lauri), risto.ritala@tut.fi (R. Ritala).

http://dx.doi.org/10.1016/j.robot.2016.06.008
0921-8890/

the state-of-the-art. In Section 3, we formulate exploration as a POMDP, and discuss possible solution methods. Section 4 reviews two forward simulation based methods for non-myopic planning in POMDPs. Section 5 introduces the sample-based approximation of mutual information suitable for robotic exploration problems, and describes a method for efficiently drawing the required samples. Section 6 describes the results of simulation experiments, and Section 7 presents the software architecture for implementation of our approach and reports the results of real-world exploration trials. Finally, Section 8 provides concluding remarks.

2. Related work

Mobile robots typically collect information on both their internal state and the state of the environment they are interacting with. In simultaneous localization and mapping (SLAM) (see Durrant-Whyte and Bailey [8] for a review) the robot must jointly estimate both its pose (internal state) and the map of the environment based on its actions and observations. Robots' information-gathering actions consist of actions that affect their pose, and hence the area covered by their sensors, and actions explicitly selecting between sensors or their operating modes. In the active SLAM problem [9], the robot's actions are selected to obtain best estimates on the pose and the map. Thus, the active SLAM problem is an exploration or sensor selection problem. The goal is to maximize information on both the robot pose and the map.

Techniques to control mobile robot exploration may be categorized e.g. by whether they apply heuristic rules or formal decision theory for selection of exploration targets. Applying heuristic rules for guiding exploration spans frontier-based approaches [10], or some next-best-view approaches such as [11]. Juliá et al. [12] classify exploration methods according to the levels of multi-robot coordination and integration with SLAM algorithms. They conclude that SLAM-integrated exploration performs best with regard to quality of map information. Their results also agree with Amigoni [13], Amigoni and Caglioti [14], who found decision theoretic criteria combining both utility of exploration and its cost (e.g. time or distance) to be preferable if both extent of the explored area and exploration time were optimized. In the following, we review in more detail some single-robot exploration techniques that employ decision theoretic criteria to guide the exploration process.

Information on the location of the robot and environment features, or landmarks, may be modelled by a multivariate Gaussian distribution. The SLAM problem can then be solved for example by applying the Extended Kalman Filter. Exploration with such feature-based maps was studied by Sim and Roy [15] who describe an A-optimal exploration method, i.e. they minimize the trace of the state covariance matrix. They discretize the location of the robot to a grid and plan an informative trajectory in open loop as a sequence of discrete positions via a breadth-first search. A similar objective function was used by [16], adopting a model predictive control (MPC) approach for optimization over multiple time steps. Discretization of the action space was also applied by [17], who applied reinforcement learning to learn parameterized robot trajectories for exploration. A somewhat complementary approach was adopted in [18], where a set of candidate exploration targets were evaluated based on an utility function designed to balance exploration of unknown areas and seeing known landmarks to help maintain good localization information. However, an explicit information-theoretic quantification of the information gain was avoided.

Martinez-Cantin et al. [19] relaxed assumptions made in earlier work, such as discretization of the action space and the myopic optimization horizon. They applied a Monte Carlo search algorithm in policy space with a Gaussian process approximation of the objective function. The policies to evaluate were selected by minimizing the average mean square error of the state estimate consisting of robot and landmark locations.

A belief space planning approach investigated by [20,21] addressed many of the limitations of earlier studies. A planner architecture consisting of an estimation layer and a decision layer combined with a model predictive control strategy for non-myopic planning was applied, assuming a Gaussian belief over robot and landmark poses. Discretization was avoided by applying a gradient descent method for computing optimal actions. Possible future measurements were treated as random variables, relaxing the assumption of maximum likelihood measurements. Exploration was considered in the sense that the objective function included an A-optimality criterion for state covariance.

Another body of work in exploration employs metric map representations such as occupancy grids [22]. Bourgault et al. [23] combined occupancy grid mapping with feature-based SLAM and used mutual information as the reward function. A discretized action space was applied with myopic optimization.

Rao–Blackwellized particle filtering (RBPF) is often applied in state-of-the-art SLAM filters for occupancy grid maps [24]. Each particle represents a map and a robot trajectory hypothesis. Stachniss et al. [2] studied myopic exploration in RBPF SLAM by discretizing the action space to a set of possible waypoints and then evaluating the approximate expected information gain when travelling to the waypoints by sampling. This work was later expanded upon [25,9] by considering alternative measures of uncertainty of the SLAM solution. These approaches consider both exploration of new areas and maintaining the consistency of the particle filter approximation.

Contribution. We present a new approximation for mutual information that is useful in mobile robotics exploration problems. The approximation can be easily integrated with forward simulation planning methods, and avoids computing full SLAM filter updates during the planning phase. In contrast to e.g. [19–21], we do not assume a Gaussian belief state. We propose and empirically evaluate in simulated and real-world domains a exploration method combining strengths of decision-theoretic POMDP based exploration and classical frontier based exploration. In all cases, we concentrate on non-myopic planning instead of the greedy one-step maximization of utility.

3. Exploration as a POMDP

Consider a robot exploring a partially observable environment. Let $s \in \mathcal{S}$ denote the hidden state of the system, comprising the state of the robot and the state of the environment. At each decision epoch in the set $\mathcal{T} = \{0, 1, \ldots, H-1\}, H \in \mathbb{N} \cup \{\infty\}$, where H is the horizon of the problem, the robot selects a control action $u \in \mathcal{U}$. Consequently, the state at the next decision epoch is determined by a transition according to a Markovian state transition model $\mathbb{T}(s', s, u)$, giving the conditional probability of transitioning to $s' \in \mathcal{S}$ from $s \in \mathcal{S}$ when action $u \in \mathcal{U}$ is executed. After the state transition, the robot obtains information regarding the state in the form of an observation. The observation is modelled by a probabilistic model $\mathbb{O}(z', s', u)$ giving the conditional probability of perceiving observation $z' \in \mathcal{Z}$ in state $s' \in \mathcal{S}$ after action $u \in \mathcal{U}$ was executed. As the robot's knowledge of the true system state is incomplete, it is represented by a belief state $b \in \mathcal{B}$, a probability density function (pdf) over the state space $\mathcal{S}$. The set $\mathcal{B}$, containing all pdfs over $\mathcal{S}$, is called the belief space.

As the robot executes control actions and perceives observations, its belief state is tracked by Bayesian filtering. Given a belief state $b \in \mathcal{B}$ and an action $u \in \mathcal{U}$, the predicted belief state b^u is computed by

$$b^u(s') = \int_{s \in \mathcal{S}} \mathbb{T}(s', s, u) b(s) \mathrm{d}s. \tag{1}$$

Given an observation z', the posterior belief b' is obtained in the update step by applying Bayes' rule:

$$b'(s') := \tau(b, u, z')(s') = \frac{\mathbb{O}(z', s', u)b^u(s')}{\eta(z' \mid b, u)}, \tag{2}$$

where the function $\tau : \mathcal{B} \times \mathcal{U} \times \mathcal{Z} \to \mathcal{B}$ is referred to as the belief update function. The normalization term in the denominator is the prior probability of observing z' : $\eta(z' \mid b, u) = \int_{s' \in \mathcal{S}} \mathbb{O}(z', s', u)b^u(s')\mathrm{d}s'$.

For each action, the robot receives an instantaneous reward given by the reward function $\rho : \mathcal{B} \times \mathcal{U} \to \mathbb{R}$. Throughout the remainder of the paper, we will assume that the reward function is bounded, i.e., there exist $\rho_{\min}, \rho_{\max} \in \mathbb{R}$ such that $\forall b, u : \rho_{\min} \leq \rho(b, u) \leq \rho_{\max}$. The reward at decision epoch $t \in \mathcal{T}$ is discounted by a factor γ^t, typically $\gamma \in [0, 1)$ [26]. The robot is said to act optimally when the expected sum of discounted instantaneous rewards over all decision epochs in $\mathcal{T}$ is maximized. The elements thus defined constitute a discounted reward partially observable Markov decision process (POMDP).

3.1. Reward functions for exploration

Suitable reward functions for exploration encourage reducing uncertainty in the belief states. Such reward functions should yield a high reward for beliefs that have the majority of their probability mass concentrated on a small subset of states. Vice versa, beliefs that correspond to uncertain state information should yield a low reward. Uncertainty functions and information gain provide useful characterizations that can be applied to define such reward functions.

Definition 1 (*Uncertainty Function and Information Gain [27]*)**.** An uncertainty function is a concave function $f : \mathcal{B} \to \mathbb{R}^+$. The information gain I_f related to an uncertainty function f is the expected reduction in the uncertainty function:

$$I_f(b, u) = f(b^u) - \mathbb{E}[f(\tau(b, u, z'))], \tag{3}$$

taking the expectation under $\eta(z' \mid b, u)$.

For example, $I_f(b, u)$ or $-f(b^u)$ may then be applied as a reward function. A common choice for the uncertainty function f is the Shannon entropy, for which the corresponding information gain is the mutual information [28]. Mutual information has been applied as a reward function in several works on robotic information gathering, see e.g. [2–6].

3.2. Optimal policies

An optimal solution to a finite-horizon POMDP is a sequence of policies $(\pi_1^*, \pi_2^*, \ldots, \pi_H^*)$, where $\pi_t^* : \mathcal{B} \to \mathcal{U}$ maps belief states to control actions, such that acting according to π_t^* at every decision epoch t maximizes the expected sum of discounted rewards over the remaining decision epochs.[1] An optimal solution is characterized by the value iteration procedure for $t = 1, 2, \ldots, H$ indicating the number of decision epochs remaining, starting with $Q_1(b, u) \equiv \rho(b, u)$:

$$Q_t(b, u) = \rho(b, u) + \gamma \int_{z' \in \mathcal{Z}} \eta(z' \mid b, u) V_{t-1}^*(\tau(b, u, z'))\mathrm{d}z', \tag{4}$$

$$V_t^*(b) = \sup_{u \in \mathcal{U}} Q_t(b, u), \tag{5}$$

$$\pi_t^*(b) = \operatorname*{argsup}_{u \in \mathcal{U}} Q_t(b, u). \tag{6}$$

[1] Note that by this formulation, at decision epoch 0, π_H^* is followed, at decision epoch 1, π_{H-1}^*, and so on.

The value function $V_t^*(b)$ gives the expected sum of discounted rewards attainable by acting optimally for t decision epochs starting at belief state b, and $Q_t(b, u)$ gives the expected sum of discounted rewards when action u is first executed and then for the remaining $(t - 1)$ decision epochs an optimal policy is followed. For an infinite horizon problem, value iteration converges to a unique fixed point V^*, and the corresponding optimal policy is stationary [29].

Traditionally, reward functions in POMDPs are state and action dependent, and thus linear in the belief state [1]. For $\rho(b, u)$ linear in the belief state, the optimal finite-horizon value function can be represented by a piece-wise linear and convex (PWLC) function [30]. Several of the most efficient currently known approximate algorithms for POMDPs leverage the PWLC representation to find solutions [31–34].

However, a reward function defined as an uncertainty function or information gain (Definition 1) is nonlinear in the belief. Araya et al. [35] approximated a reward function convex in the belief state by a piece-wise linear function and were able to apply standard POMDP algorithms to approximate the optimal policy. Online POMDP algorithms [36] that repeatedly find an optimal action only for the current belief state during task execution and do not rely on an explicit closed form representation of the value function are also applicable to POMDPs with nonlinear rewards. Several online algorithms are based on a finite depth tree search over reachable belief states. The number of reachable belief states after k decisions is $(|\mathcal{U}|\ |\mathcal{Z}|)^k$, which quickly makes the search intractable for problems with large action and observation spaces. If either $\mathcal{U}$ or $\mathcal{Z}$ are uncountable, the basic tree search method is not applicable.

In mixed-observability problems, a part of the state is fully observable and evolves deterministically. In such mixed-observability domains with a mutual information reward function, Atanasov et al. [5] showed that if the observation model is linear-Gaussian w.r.t. the target state, the optimal policy for maximizing mutual information is open-loop, and if possible actions are constrained by the fully observable part of the state, Lauri and Ritala [6] showed that under certain conditions the problem can be relaxed into a multi-armed bandit problem.

3.3. Open-loop solution to a POMDP

Instead of resorting to the aforementioned approaches for POMDPs with nonlinear rewards, we consider an open loop approximation that ignores the effect of information provided by future observations, instead choosing actions only on the basis of currently available information. By this approach, we are able to handle continuous action and observation spaces, although at a loss in optimality of the solution.

Denote the current belief state by b_0. Given an action sequence $u_{0:H-1}$, the predicted pdf of the measurements $z_{1:H}$ given the information in the current belief state is

$$p(z_{1:H} \mid b_0, u_{0:H-1}) = \int_{\mathcal{S}^{|H+1|}} b_0(s_0) \prod_{k=0}^{H-1} \mathbb{O}(z_{k+1}, s_{k+1}, u_k) \times \mathbb{T}(s_{k+1}, s_k, u_k)\mathrm{d}s_{0:H}. \tag{7}$$

The open loop objective function over the corresponding decision epochs is

$$J_H(b_0, u_{0:H-1}) = \rho(b_0, u_0) + \mathbb{E}\left[\sum_{k=1}^{H-1} \gamma^k \rho(\tau(b_{k-1}, u_{k-1}, z_k), u_k)\right], \tag{8}$$

where the expectation is taken w.r.t. $p(z_{1:H} \mid b_0, u_{0:H-1})$, and belief states follow the recursion $b_k = \tau(b_{k-1}, u_{k-1}, z_k)$. The crucial difference compared to the value iteration and policies in

Eqs. (4)–(6) is that the sequence of actions is *fixed* beforehand. Thus, an optimal open loop solution $u^*_{0:H-1}$ is a sequence of H actions that maximizes the expected sum of discounted rewards when information provided by observations during the subsequent decision epochs is not exploited to select any of the subsequent actions:

$$u^*_{0:H-1} = \operatorname*{argsup}_{u_{0:H-1} \in \mathcal{U}^{|H|}} J_H(b_0, u_{0:H-1}). \tag{9}$$

We denote the optimal open loop value as $J^*_H(b_0) \equiv J_H(b_0, u^*_{0:H-1})$.

A receding horizon control strategy is adopted as follows. First, Eq. (9) is solved, u^*_0 is executed and an observation z_1 is perceived. The belief state is revised to $\tau(b_0, u^*_0, z_1)$, and the process is repeated. In a sense, feedback is reintroduced by computing the optimal open loop action sequence again at every decision epoch, applying updated information. This control strategy is also referred to as open-loop feedback control (OLFC), and we denote the expected value of following this OLFC policy for H decision epochs as $\hat{J}_H(b_0)$.

It is interesting to consider how much worse is the value $\hat{J}_H(b_0)$ of the OLFC policy compared to the value $V^*_H(b_0)$ of an optimal closed loop policy. If the reward function is bounded, then the loss of OLFC compared to an optimal policy is bounded. To see this, consider that the worst and best possible policies attain a reward of $\rho_{\min}$ and $\rho_{\max}$, respectively, on every decision epoch. Taking into account the discounting of rewards, we have that the difference in value of *any* two policies over H decision epochs is at most $\sum_{k=0}^{H-1} \gamma^k(\rho_{\max} - \rho_{\min})$. It should be noted that in practice an optimal policy rarely obtains the maximum reward on every decision epoch, nor does an open loop policy obtain the minimum reward on every decision epoch, such that the suboptimality is often less in practice than as indicated by the bound.

The OLFC policy always performs at least as well as executing an optimal pure open-loop action sequence $u^*_{0:H-1}$, as shown by the following theorem.

Theorem 1 (*Performance of Open-Loop Feedback Control [37]*)**.** *For any $b \in \mathcal{B}$,*

$$\hat{J}_H(b) \geq J^*_H(b). \tag{10}$$

The suboptimality of following the OLFC policy may be further analysed by noting that in certain special cases an optimal policy is effectively an open loop policy, see [38] for a discussion of such cases. Furthermore, an approximation of an upper bound for the optimal closed loop value may be found e.g. applying hindsight optimization, see [39].

4. Forward simulation for open loop control

Solving the open loop control problem of Eq. (9) corresponds to finding an optimal policy to a POMDP with no observations, which itself is an NP-complete problem [7]. Since the optimal solution is a sequence of actions $u^*_{0:H-1}$, it is possible to apply search algorithms to find the solution. The search space is $\mathcal{U}^{|H|}$, the space of action sequences with H actions. As the objective function J_H may not be differentiable with respect to the action sequence, gradient methods are not generally applicable. We review an algorithm for both a finite (Section 4.1) and an uncountable (Section 4.2) action space, leading to finite or uncountable search spaces, respectively.

4.1. Partially observable Monte Carlo planning

The partially observable Monte Carlo planning algorithm (POMCP) of [40] is a variant of Monte Carlo Tree Search (MCTS) [41] to solve POMDPs with finite action and observation spaces. POMCP runs a sequence of forward simulations called episodes on the problem, and collects the rewards obtained in the simulation and uses them to evaluate alternative strategies by constructing a search tree over the possible policies. An exploration parameter e determines the balance between exploitation and exploration behaviour during the search.

We have adapted POMCP for solving the open loop control problem of Eq. (9). A search tree over action sequences is iteratively constructed, starting from a tree containing just the root node corresponding to the current belief state b. The overall objective is to estimate the values of possible open loop action sequences, in order to be able to select an action to execute in the current belief state that leads to the most favourable outcome over the next H decisions.

Each node in the search tree corresponds to a particular action sequence, and we will refer to a node by an action sequence $u_{0:k}$. The root node corresponds to the current belief state b when no actions have yet been taken, and is referred to by an empty set. For each node $u_{0:k}$, a value estimate and a count are maintained. The value estimate $V(u_{0:k})$ is the mean sum of discounted rewards recorded over all simulations that continue from $u_{0:k}$ until the final decision epoch H. The count $N(u_{0:k})$ is the number of times a node has been visited during the search.

Fig. 1 gives an overview of a simulation episode, and shows how the search tree is iteratively constructed. Every episode consists of a tree policy stage and a rollout stage. In the next two paragraphs, we give an informal overview of both phases. After that, we describe the algorithm in more detail, filling in the missing details.

During the tree policy stage, the current belief state b is propagated applying the belief update equation τ with each action u determined by the search tree nodes visited and each observation z' determined by sampling from the prior distribution $\eta(z' \mid b, u)$. For example, at the root node corresponding to b, the value estimates and counts of the child nodes are applied to choose one child node corresponding to an action u. An observation is sampled from $\eta(z' \mid b, u)$, the belief state is updated to $b = \tau(b, u, z')$, and the tree policy continues to a new node u. Similarly, at any node $u_{0:k}$ a child node $u_{0:k+1} = \{u_{0:k}, u\}$ is chosen according to the values and counts of the child nodes, the belief state is updated, and the tree policy stage continues at node $u_{0:k+1}$. An example of a trajectory of nodes traversed during the tree policy stage is indicated by the arrows in the leftmost part of Fig. 1.

Once a leaf node that does not have any children is encountered, the tree policy cannot be continued. The tree is expanded by adding new child nodes as appropriate, an example of expanding the tree with a single node is shown in the second from left part of Fig. 1, under the "Expansion" indicator. The search then moves to the rollout stage. In the rollout stage, a simulation until decision epoch H is performed, updating belief and recording rewards along the way, see the third from left part of Fig. 1. In the final backpropagation phase of the rollout stage, shown on the rightmost part of Fig. 1, the total reward recorded is applied to update the values of each node visited during the simulation episode. Once a simulation episode is completed, a new one is again started at the root of the search tree, now expanded with new nodes.

Algorithm 1 presents the version of POMCP that we have adapted to solve the open loop control problem of Eq. (9). The algorithm takes as parameters the POMDP model, an exploration parameter e, an optimization horizon H, and the current belief state b. The while loop (Line 2) is executed until a specified stopping criterion, e.g. the maximum number of simulation episodes N_s, has been reached. Each iteration of the while loop corresponds to one simulation episode.

If the condition on Line 10 is false, the current search tree is applied to guide the action selection in the tree policy stage. The

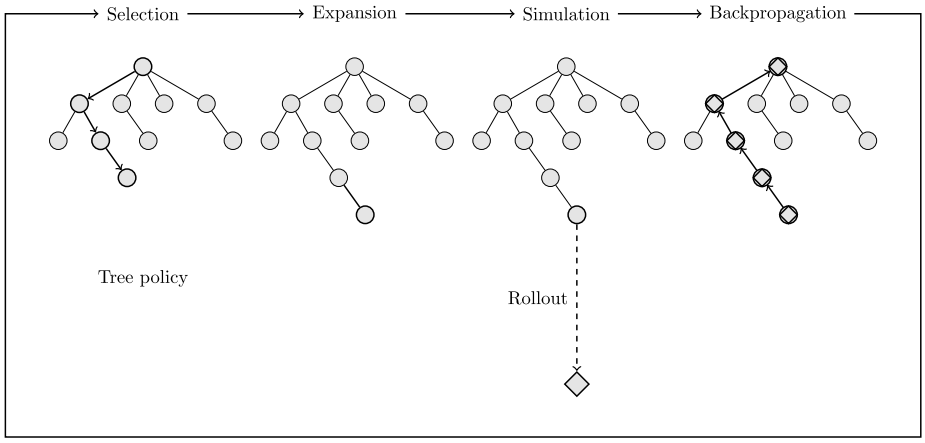


Fig. 1. Overview of POMCP simulation episodes. The tree policy stage consists of a selection and expansion phase, and the rollout stage consists of a simulation and backpropagation phase.
Source: Figure adapted from [41].

Algorithm 1 POMCP [40] for open-loop control

Input: The POMDP model, exploration parameter e, optimization horizon H, current belief state b_0

1: **function** POMCP(b_0)
2: **while** stopping conditions not fulfilled **do**
3: SIMULATE($b_0, \emptyset, 0$)
4: $N(\emptyset) \leftarrow N(\emptyset) + 1$
5: **end while**
6: **return** $\operatorname{argmax}_u V(u)$
7: **end function**

8: **function** SIMULATE($b, u_{0:d}, d$)
9: **if** $d = H - 1$ **then return** 0
10: **else if** ISLEAF($u_{0:d}$) **then**
11: Add $\{u_{0:d}, u\}$ to tree $\forall u \in \mathcal{U}$
12: **return** ROLLOUT(b, d)
13: **end if**
14: $u \leftarrow \operatorname{argmax}_{u'} V(\{u_{1:d}, u'\}) + e\sqrt{\log N(u_{0:d})/N(\{u_{0:d}, u'\})}$
15: Sample $z' \sim \eta(z' \mid b, u)$
16: $r \leftarrow \rho(b, u) + \gamma \cdot$ SIMULATE($\tau(b, u, z'), \{u_{0:d}, u\}, d + 1$)
17: $N(\{u_{0:d}, u\}) \leftarrow N(\{u_{0:d}, u\}) + 1$
18: $V(\{u_{0:d}, u\}) \leftarrow V(\{u_{0:d}, u\}) + \frac{r - V(\{u_{0:d}, u\})}{N(\{u_{0:d}, u\})}$
19: **return** r
20: **end function**

21: **function** ROLLOUT(b, d)
22: **if** $d = H - 1$ **then return** 0
23: **else**
24: Sample $u \sim \pi_{\text{rollout}}(b)$
25: Sample $z' \sim \eta(z' \mid b, u)$
26: **return** $\rho(b, u) + \gamma \cdot$ ROLLOUT($\tau(b, u, z'), d + 1$)
27: **end if**
28: **end function**

action to execute is determined by a selection rule maximizing an upper confidence bound [42,43] as shown on Line 14.[2] The first term in the bound is the current estimate of the expected value of the action, and the second term is an exploration bonus dependent on the exploration parameter e. The parameter e is typically selected such that it is on a similar scale as typical rewards in the problem. The exploration bonus encourages trying actions that have rarely been tried, prompting exploratory behaviour. After sampling an observation z', the tree policy stage is continued by a recursive call to the function SIMULATE (Line 16). Finally, the counts and values of all action sequences visited during the episode are updated with the reward information gained during the episode in the backpropagation phase (Lines 17–18).

If an action sequence is encountered that is a leaf node of the current search tree, a rollout stage is entered (Line 10). The search tree is expanded to include possible successor actions (Line 11). By calling the function ROLLOUT, a sample of the reward until end of the optimization horizon is obtained. This is achieved by simulating a given rollout policy (Line 24) until the end of the optimization horizon. A typical choice of rollout policy is to sample an action uniformly at random from $\mathcal{U}$ [40].

Once the while loop of the main function terminates, an action recommendation $\hat{u} \leftarrow \operatorname{argmax}_u V(u)$ is returned (Line 6). The value estimates of POMCP converge asymptotically to the optimal open loop value [40].

4.2. Sequential Monte Carlo planning

Kantas et al. [44] present a sequential Monte Carlo (SMC) method to maximizing functions of the form

$$J_H(\phi) = \int_{\mathcal{Y}} G(y, \phi) p(y \mid \phi) \mathrm{d}y, \tag{11}$$

which is seen to be equivalent to Eq. (8) by setting $y = z_{1:H}$, $\phi = \{b_0, u_{0:H-1}\}$, and choosing G as a function that encompasses the appropriate summations.[3]

The key idea of the SMC optimization method of [44] is to construct a sequence of distributions $\lambda_\nu \propto p(\phi) J_H(\phi)^\nu$ where $p(\phi)$ is an arbitrary prior that is nonzero at the maximizers of Eq. (11). This approach is related to the Bayesian interpretation of simulated annealing and maximum likelihood estimation, see [45]. As $\nu \to \infty$, $\lambda_\nu(\phi)$ becomes concentrated on the set of maximizers of J_H. In practice, a set of particles is evaluated via the objective function and, based on the evaluation result, particles are either discarded or carried forward to the next evaluation round. Eventually the particle set converges to represent an action sequence maximizing Eq. (11).

[2] If $N(\{u_{0:d}, u'\}) = 0$ for any $u' \in \mathcal{U}$, the action is instead sampled uniformly at random from $\{u' \in U \mid N(\{u_{0:d}, u'\}) = 0\}$.

[3] For this method, G must be finite and strictly positive, which can be achieved by adding a finite positive constant without affecting the argument maximizing the objective function.

The SMC method is presented in Algorithm 2. The algorithm iterates over $l = 1, \ldots, l_{\max}$, where the number of iterations $l_{\max}$ is chosen according to the accuracy and run-time requirements. The algorithm maintains a set of M particles. At iteration l, each particle indexed by i with an associated weight $w_l^{(i)}$ represents a sequence of actions $u_{0:H-1,l}^{(i)}$ and a set of ν_l observation sequences conditional on the action sequence, denoted $z_{1:H,j}^{(i)}$, $j = 1, \ldots, \nu_l$. The integers $\{\nu_l\}_{l\geq 1}$ must form a strictly increasing sequence. The algorithm has been shown to converge to the optimal solution for logarithmic sequences, although in practice faster increasing linear or geometric sequences are used [45,44]. The sequence $\{\nu_l\}_{l\geq 1}$ is analogous to the temperature cooling schedule in simulated annealing.

Algorithm 2 Sequential Monte Carlo optimization, adapted from [44].

Input: The POMDP model, number of iterations l_{max}, increasing integer sequence $\{\nu_l\}_{l=1}^{l_{max}}$, sampling kernels q_l, optimization horizon H, resampling threshold M_t.

1: **function** SMC(b_0)
2: **for** $l = 1, \ldots, l_{max}$ **do**
3: **for** $i = 1, \ldots, M$ **do**
4: $u_{0:H-1,l}^{(i)} \sim q_l(\cdot \mid u_{0:H-1,l-1})$
5: **for** $j = 1, \ldots, \nu_l$ **do**
6: $z_{1:H,j}^{(i)} \sim p(z_{1:H} \mid b_0, u_{0:H-1,l}^{(i)})$
7: **end for**
8: $w_l^{(i)} \leftarrow w_{l-1}^{(i)} \prod_{j=1}^{\nu_l} \Big(\rho(b_0, u_{0,l}^{(i)}) + \sum_{k=1}^{H-1} \gamma^k \rho(\tau(b_{k-1}^{(i)}, u_{k-1,l}^{(i)}, z_{k,j}^{(i)}), u_{k,l}^{(i)}) \Big)$
9: **end for**
10: **for** $i = 1, \ldots, M$ **do**
11: $w_l^{(i)} \leftarrow w_l^{(i)} / \sum_{j=1}^{M} w_l^{(j)}$
12: **end for**
13: **if** $1/\sum_{i=1}^{M} (w_l^{(i)})^2 < M_t$ **then**
14: Resample particles, set weights $w_l^{(i)} \leftarrow 1/M \quad \forall i$
15: **end if**
16: **end for**
17: **end function**

A control sequence is sampled for each particle from a kernel q_l (Line 4). For $l = 1$, the initial control sequences are sampled from a prior distribution that is nonzero at the maximizers of (11). In general, the choice of q_l depends on the problem specifics. The selection of the kernel determines how new control sequence samples depend on the earlier ones, and it has a central role in determining how efficient the algorithm is. For further discussion on kernel selection, we direct the reader to [46].

After control sequences have been sampled, they are applied to generate ν_l replicas of possible state-observation sequences when executing the control sequence (Lines 5–7). The weights of the particles are updated by approximating the objective function applying the state and observation samples contained in each particle (Line 8). In this step, the belief states for given j, l, i follow the recursion $b_k^{(i)} = \tau(b_{k-1}, u_{k-1,l}^{(i)}, z_{k,j}^{(i)})$.

At the end of each iteration, resampling is performed if the effective sample size [47] falls below the threshold value M_t (Line 14). After step $l = l_{\max}$, the maximizer estimate is extracted as $u_{1:H-1,l_{\max}}^{(i_m)}$, where $i_m = \operatorname{argmax}_i w_l^{(i)}$. Algorithm 2 is easily parallelized, as each particle can be processed independently.

Fig. 2 shows an example of applying Algorithm 2 to a robotic exploration problem. The figure shows for three iterations of the algorithms how possible trajectories for the robot corresponding to the particles are evaluated, weighted and resampled, converging towards the most informative local trajectory.

5. Mutual information in mobile robotics

In this section, we present a new approximation for mutual information (MI) that is especially useful in mobile robotics domains. As the approximation is sample-based, it is straightforward to apply it together with forward simulation based planning. The approximation is derived in Section 5.1, and in Section 5.2 we present a method for efficiently drawing observation samples in the case occupancy grids are used as a map representation.

5.1. Approximation of mutual information

The state $s \in \mathcal{S}$ in a robotic exploration problem is decomposed into the robot's internal state $x \in \mathcal{X}$ and the environment state or map $m \in \mathcal{M}$, i.e. $\mathcal{S} = \mathcal{X} \times \mathcal{M}$. The internal state represents the pose of the robot, and the environment state represents all relevant features of the world. We make the following assumption asserting that the robot is unable to affect the environment state through its actions.

Assumption 1. The robot's internal state and environment state evolve independently, i.e.

$$\mathbb{T}(s', s, u) = \mathbb{T}_x(x', x, u)\mathbb{T}_m(m', m), \tag{12}$$

where $\mathbb{T}_x : \mathcal{X} \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}^+$ and $\mathbb{T}_m : \mathcal{M} \times \mathcal{M} \to \mathbb{R}^+$ are the robot and environment state transition models, respectively.

We remark that this assumption does not imply that the internal and environment state are independent.

Suppose information gain (Definition 1) with Shannon entropy as the uncertainty function is applied as the reward in an exploration POMDP. This leads to $I_f(b, u)$ equal to the mutual information, which can be approximated as follows.

Theorem 2. *Denote by $b \in \mathcal{B}$ the current belief state, by $u \in \mathcal{U}$ the current action, and let X, M, and Z denote the random variables depicting the robot state, environment state, and observation at the next decision epoch. Consider the mutual information of the state (X, M) and observation Z, defined*

$$I(X, M; Z \mid b, u) = \int_{\mathcal{M}} \int_{\mathcal{X}} p(x', m' \mid b, u) \times \log \frac{p(x', m' \mid b, u)}{p(x' \mid b, u)p(m' \mid b, u)} \mathrm{d}x' \mathrm{d}m'. \tag{13}$$

If Assumption 1 holds and if $\{x'^{(i)}, z'^{(i)}\}_{i=1}^N \sim p(x', z' \mid b, u)$, where N is the number of samples, then the approximation

$$I_N = \frac{1}{N} \sum_{i=1}^{N} \left[\log \frac{p(x'^{(i)} \mid b, u, z')}{p(x'^{(i)} \mid b, u)} + \int_{\mathcal{M}} p(m' \mid b, u, z'^{(i)}, x'^{(i)}) \times \log \frac{p(m' \mid b, u, z'^{(i)}, x'^{(i)})}{p(m' \mid b, u)} \mathrm{d}m' \right] \tag{14}$$

converges almost surely to $I(X, M; Z \mid b, u)$ as $N \to \infty$.

Proof. By the chain rule for MI [28],

$$I(X, M; Z \mid b, u) = I(X; Z \mid b, u) + I(M; Z \mid X, b, u). \tag{15}$$

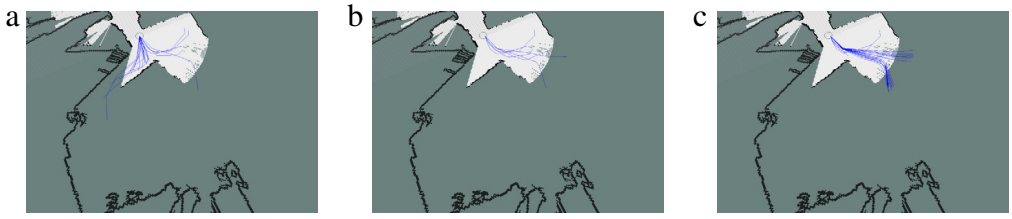


Fig. 2. Convergence of the SMC algorithm in a robotic exploration task. Subfigures 2(a) through 2(c) indicate increasing iterations l of the algorithm. The robot location and outline are indicated by a thin black circle. Local trajectories shown by the blue lines are superimposed on a greyscale image indicating current map information. The map shows occupied cells in black, free cells in white, and unknown cells in grey. The local trajectories are sequentially evaluated based on their informativeness, and eventually converge towards the most informative local trajectory. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The first term of Eq. (15) is the MI of X and Z:

$$I(X; Z \mid b, u) = \int_{\mathcal{Z}} \int_{\mathcal{X}} p(x', z' \mid b, u) \log \frac{p(x' \mid b, u, z')}{p(x' \mid b, u)} \mathrm{d}x' \mathrm{d}z'. \tag{16}$$

The second term of Eq. (15) is the conditional MI of M and Z, given X. By the definition of conditional MI [28],

$$I(M; Z \mid X, b, u) = \mathbb{E}_{Z,X,M} \left[\log \frac{p(m', z' \mid x', b, u)}{p(m' \mid x', b, u) p(z' \mid x', b, u)} \right], \tag{17}$$

where the expectation is taken w.r.t. $p(z', x', m' \mid b, u)$. By Assumption 1, $p(m' \mid x', b, u) = p(m' \mid b, u)$, and by the law of conditional probability we have

$$I(M; Z \mid X, b, u) = \int_{\mathcal{Z}} \int_{\mathcal{X}} p(x', z' \mid b, u) \left[\int_{\mathcal{M}} p(m' \mid x', z', b, u) \times \log \frac{p(m' \mid x', z', b, u)}{p(m' \mid b, u)} \mathrm{d}m' \right] \mathrm{d}x' \mathrm{d}z'. \tag{18}$$

We have thus shown that both terms on the right hand side of Eq. (15), namely, Eqs. (16) and (18), are expectations under the same joint pdf $p(x', z' \mid b, u)$. Drawing N samples from this pdf and defining I_N as above, by the lemma of Monte Carlo integration [48, Ch. 3.2] I_N converges to $I(X, M; Z \mid b, u)$ almost surely. □

To apply the approximation presented in Theorem 2, we draw samples from $p(x', z' \mid b, u)$ by the following three-step method for $i = 1, 2, \ldots, N$:

1. Sample from the current belief state $s^{(i)} = (x^{(i)}, m^{(i)}) \sim b$,
2. Propagate the sample through the factored state transition model of Assumption 1, yielding new samples $x'^{(i)} \sim \mathbb{T}_x(x', x^{(i)}, u)$ and $m'^{(i)} \sim \mathbb{T}_m(m', m^{(i)})$.
3. The sample from the previous step is distributed according to the predictive pdf $b^u = p(x', m' \mid b, u)$. Recalling $s' = (x', m')$, we have by marginalizing over $\mathcal{M}$

$$p(x', z' \mid b, u) = \int_{\mathcal{M}} \mathbb{O}(z', s', u) b^u(s') \mathrm{d}m', \tag{19}$$

and we may thus sample $z'^{(i)} \sim \mathbb{O}(z', s'^{(i)}, u)$ to obtain the desired samples.

Theorem 2 is applicable with three restrictions. First, we must be able to draw samples from the state transition and observation models either directly or e.g. applying importance sampling. Secondly, evaluating the first sum term in Eq. (14) requires us to be able to evaluate the predictive probability and the posterior probability of a robot state $x'^{(i)}$ given an action u and a measurement $z'^{(i)}$. The predictive pdf can be evaluated applying the state transition model. It is not generally easy to find the posterior $p(x'^{(i)} \mid b, u, z'^{(i)})$, but in some cases it can be approximated by a unimodal distribution. As argued by [24], such a case arises e.g. for robots equipped with accurate range sensors such as laser range finders (LRFs). LRF data can be leveraged via scan matching to obtain localization estimates that are significantly more precise than estimates based only on robot's state transition models. Consequently, the measurement likelihood is strongly peaked and the posterior pdf of the robot state given the observation may be approximated by a unimodal distribution, e.g. a Gaussian with a small covariance. Third and finally, evaluating the integral term in the sum of Eq. (14) requires us to be able to compute posterior map pdfs given a pose and an observation. This corresponds to the problem of mapping with known poses [22]. We must also be able to evaluate the integral expression over the map state. The difficulty of this depends on the map representation, but is simple e.g. for occupancy grid maps due to the independence assumption of the grid cells: the integral term reduces to a sum over cells.

The overall effect of the approximation is that we avoid computing the full state posterior pdf and solving the implied SLAM problem when evaluating the expected information gain of a control action. Note that this does not preclude executing a SLAM algorithm for the real process independently of the task of finding an optimal open loop action sequence. The current belief state $b = p(x, m)$ provided by the SLAM algorithm is applied as initial information for finding the optimal open loop solution. Sometimes the real process SLAM particle filter may be susceptible to losing consistency [25,9], and applying the proposed approximation may lead to a failure state as this possibility is not considered while planning trajectories. In such cases, either the SLAM algorithm must be improved or an alternative approximation for MI applied.

5.2. Observation sampling in occupancy grid maps

To apply the forward simulation based planning algorithms presented in Section 4 together with the approximation of MI presented above, efficient methods for drawing observation samples given a sequence of control actions are required. In this section, we introduce such a sampling method for occupancy grid maps [22], a map representation widely applied in mobile robotics.

An occupancy grid map defines a partition of a space into equally-sized cells c. We assume that the map $\mathcal{M}$ is composed of a finite number of such cells. Each cell is in one of two hidden states, either free (0) or occupied (1). As cell occupancy is often not known exactly, an occupancy probability $P(c = 1) := p_c \in [0, 1]$ models the information regarding the state of the cell.

The occupancy information is revised through sensor observations. Depending on the robot's sensors and its pose x, it is possible that only a subset $\tilde{\mathcal{M}}(x) \subset \mathcal{M}$ of the map is sensed by a robot at x. Fig. 3 shows an example of such a scenario. We note that even several measurements over a trajectory traversed by the robot may only provide information regarding a subset of the whole map area. As the cells of an occupancy grid map are independent, this suggests that an efficient method for sampling observations may be

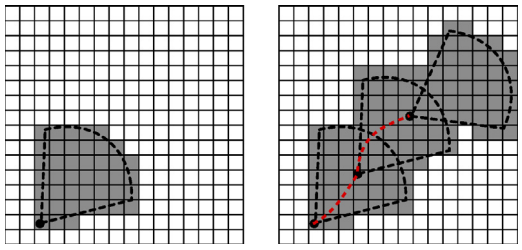


Fig. 3. Occupancy grids and observed areas. The robot's pose is marked by a black circle, and the dashed sector depicts sensor range. The cells with a grey background denote the subset of the occupancy grid that is potentially sensed either at a single pose (left side) or a trajectory depicted by dashed lines between robot poses (right side).

constructed by considering only the subset of potentially sensed map cells.

We represent observations z as collections of pairs of the form (c, j), where c denotes the cell and $j \in \{0, 1\}$ is the *observed occupancy* for the cell: free (0) or occupied (1). Given an optimization horizon H and the control actions $u_{0:H-1}$, a sequence $z_{1:H}^{(i)}$ of observation samples is constructed. In the case of a laser range finder, raytracing may be applied to draw observation samples. Raytracing simulates a laser beam originating from the robot's pose x, tracking which cells the beam travels through. When the ray intersects a cell c, the occupancy information p_c is applied to determine whether the cell is free and the beam passes through, or the cell is occupied and the beam hits an obstacle in the cell. In the former case, $(c, 0)$ is inserted to $z_t^{(i)}$, and in the latter case $(c, 1)$ is inserted to $z_t^{(i)}$ and the raytracing is terminated.

As the same cell may be observed at multiple decision epochs and the observations should be consistent, i.e. they are generated from the same true underlying map, a persistent map sample m^p keeping track of the occupancy states of cells intersected by the simulated laser beams is maintained. The persistent map sample is also a collection of pairs (c, o) of a cell c and its *sampled occupancy* $o \in \{0, 1\}$ in the map. Whenever a cell c is encountered during ray tracing, it is first checked whether m^p contains it already.

An observation sampling algorithm implementing these features is presented in Algorithm 3. A sample of the current pose $x_0^{(i)}$ is drawn from $p(x_0)$ obtained by marginalizing from the current belief state $b_0 = p(x_0, m_0)$ (Line 3). For each decision epoch $0 \leq t \leq H-1$, a pose sample $x_{t+1}^{(i)}$ is drawn (Line 5) from the robot's dynamic model. Raytracing with the persistent map sample m^p is applied to sample $z_{t+1}^{(i)}$ (Line 6) from the observation model. In this case, the sampling is implemented in the function RAYTRACE.

For each laser beam incidence angle and each cell along the beam, the occupancy state of the cell is first determined. If the cell c has already been encountered during the sampling of any $z_k^{(i)}$, $k < t$, its occupancy state is retained from the persistent map sample, and otherwise its occupancy state is sampled according to p_c (Lines 13–18). Finally, the actual observed occupancy j is sampled according to the sensor observation model (Line 19), and the resulting observation is added to the observation sample. The value $j = 1$ indicates that the beam hit an obstacle, and the raytrace for this incidence angle is terminated (Line 22).

As map occupancies are sampled only as required (when a simulated LRF ray enters a cell), computational savings are accrued compared to the naive approach of always sampling occupancy values for every cell before raytracing. After the observation samples $z_{1:H}^{(i)}$ have been obtained, the corresponding values of MI can be computed applying the approximation of Theorem 2. For integration with the SMC method (Algorithm 2), the sampling algorithm (Algorithm 3) is executed independently for the control action sequence of each particle. For integration with the POMCP method (Algorithm 1), the action u_t on Line 5 of Algorithm 3 is obtained either from the tree policy or from the rollout policy.

Algorithm 3 Observation sampling for a laser range finder (LRF).

Input: The POMDP model, optimization horizon H.

1: **function** SAMPLE(b_0)
2: $m^p \leftarrow \emptyset$
3: Sample $x_0^{(i)} \sim p(x_0)$
4: **for** $t = 0, \ldots H-1$ **do**
5: Sample $x_{t+1}^{(i)} \sim \mathbb{T}_x(x_{t+1} \mid x_t^{(i)}, u_t)$
6: $z_{t+1}^{(i)} \leftarrow$ RAYTRACE($x_{t+1}^{(i)}$)
7: **end for**
8: **return** $z_{1:t}^{(i)}$
9: **end function**

10: **function** RAYTRACE(x)
11: **for all** incidence angles α of the LRF **do**
12: **for all** cells c along a beam starting at x in direction α **do**
13: **if** $(c, \cdot) \notin m^p$ **then**
14: Sample $o \sim p_c$
15: $m^p \leftarrow m^p \cup \{(c, o)\}$
16: **else**
17: $o \leftarrow \{o \mid (c, o) \in m^p\}$
18: **end if**
19: Sample $j \sim p(z' \mid o, x)$
20: $z \leftarrow z \cup \{(c, j)\}$
21: **if** $j = 1$ **then**
22: **break**
23: **end if**
24: **end for**
25: **end for**
26: **return** z
27: **end function**

6. Simulated exploration experiments

We studied the POMDP based planning in a set of exploration experiments in three simulated domains. In particular, the effect of prior information and the optimization horizon H were examined. The proposed planning approaches were compared to myopic (one-step greedy) planning, which is the special case of the proposed approach with $H = 1$ (see e.g. [2,15,17]), and frontier-based exploration [10].

We begin by considering an illustrative toy example in Section 6.1. In Section 6.2, we study the performance of the proposed POMDP based planning approach in three domains: a maze, an office, and an outdoor environment, each having different scale and other properties. These simulations were implemented in MATLAB. Based on the results, we propose a further improvement of the POMDP approach, and compare the improved method with frontier-based exploration in Section 6.3.

6.1. An illustrative example

We first consider the effect of prior information and optimization horizon for making a single decision in a small toy domain as shown in Fig. 4. The white and black cells in the map are unoccupied and occupied cells, respectively. The robot was equipped with a simulated laser range finder with a maximum range of 2 m for sensing the environment state, and had three possible action choices to select between as indicated by the trajectories overlaid in the figure. Each action was designed specifically to have a varying effect on the achievable mutual information over a horizon of several decisions. Action a_2, corresponding to moving straight ahead, initially does not provide much information about the environment as the view of the range finder is partially blocked by the corridor leading towards the top of the figure. Despite this, executing a_2 eventually leads the robot to the large open area in the

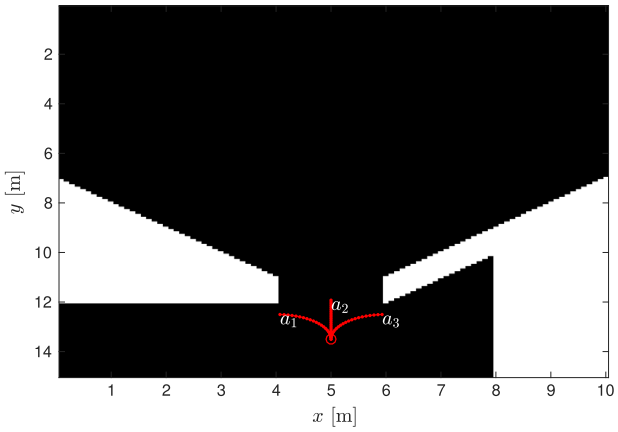


Fig. 4. A test map for a single decision. The robot's starting location is indicated by the circle marker, and three trajectories are labelled by the corresponding actions a_1, a_2 and a_3.

Table 1
Values of actions in the small map as function of the optimization horizon H and for a non-informative and informative prior, shown with the true optimal action (bottom row).

	Action	$H = 1$	$H = 2$	$H = 3$	$H = 4$
No additional prior	a_1	102.4	173.6	235.7	212.4
	a_2	101.8	167.1	**248.8**	**325.4**
	a_3	**106.5**	**177.3**	235.0	223.0
Informative prior	a_1	81.8	145.9	189.7	200.5
	a_2	78.9	**157.0**	**231.0**	**298.5**
	a_3	**97.2**	129.4	172.5	140.8
Optimal action		a_3	a_2	a_2	a_2

top part of the environment. In contrast, actions a_1 and a_3 yield greater immediate information gain, but ultimately lead to dead-ends at the left and right bottom parts of the map. The boundary of the area was assumed to be known to the robot, i.e., the robot can detect when an action would take it outside the boundary of the environment.

We note that in a real application also intermediate choices with a curvature between that of a_1 and a_2, or a_2 and a_3, would be available for the robot. However, for the purpose of this example, we did not consider them. In the subsequent experiments on larger maps, the SMC method considers the full uncountable action space.

As prior information, the robot recorded a single observation with the laser range finder from the initial location, and updated the map occupancy probabilities accordingly. The robot's initial location was selected so that this observation did not reveal the presence of the occupied cells in the map. Additionally, two cases were considered: one with no additional prior information, and another one with an informative prior that indicated the occupied areas in the environment (the white cells in Fig. 4), but leaving occupancy probabilities elsewhere as they were.

We applied the POMCP algorithm (Algorithm 1) to plan a single decision. Table 1 shows the value estimate of each action for either case of prior information as a function of the optimization horizon H. The action with the greatest value is indicated by a bold font. Also the optimal action, as found by an exhaustive search over all possible action sequences while assuming perfect sensing, is shown for reference. As H increases, action a_2 is chosen in the case of no additional prior for $H \geq 3$ and in case of the informative prior for $H \geq 2$. In case of no additional prior information, a_2 is eventually preferred as the other actions eventually lead (after two decision epochs in the planning phase) to a dead-end where the robot cannot progress any further and thus cannot collect more information. In the case of an informative prior, however, a_2 is preferred already for $H = 2$: as the occupied cells are indicated, the robot is able to avoid the dead-ends blocked by the occupied cells near the bottom corners of the map. We also note that the difference in the values between the recommended and second-best action tend to be greater for the informative prior, indicating that the algorithm is able to more confidently distinguish between the actions.

6.2. Performance of POMDP based planning

We next examine the effect of prior information and the optimization horizon in domains larger than the toy example presented above. Section 6.2.1 outlines the experimental setup, and Section 6.2.2 presents the results.

6.2.1. Experimental setup

Three environments as illustrated in Fig. 5 were examined: a maze (Fig. 5(a)), office-like (Fig. 5(b)), and an open outdoor environment (Fig. 5(c)). White and black cells are unoccupied and occupied, respectively, and gray cells indicate undefined or unobserved cells. The sizes of the environments varied from approximately 10-by-10 m (maze) to 250-by-250 m (outdoor). The office and outdoor maps were obtained from an online data repository.[4] The x and y coordinates of the robot's starting location in each environment were as follows. In the maze environment, $(x, y) = (7.9, 8.6)$; in the office environment, $(38.5, 58.0)$; and in the outdoor environment, $(90.0, 106.0)$.

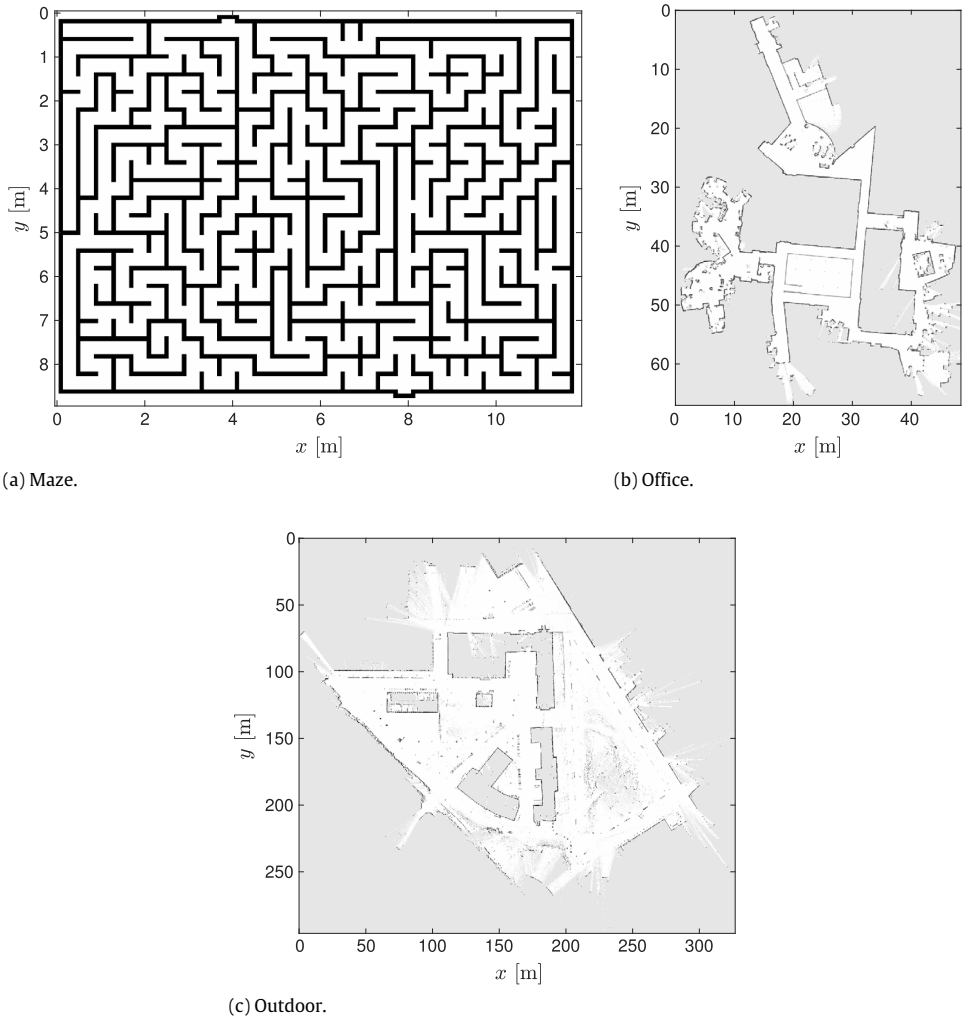
The simulated robot is able to turn in place, and it is controlled by applying a linear and angular velocity. For the maze and office environments, the robot was assigned a maximum linear velocity of 1 m per second, and for the outdoor environment 3 m per second. The angular velocity was constrained to be between -0.5 and 0.5 rad per second. The robot was equipped with a laser range finder with a maximum range of 4 m (maze and office environments), or 15 m (outdoor environment). Time was discretized into 1 s intervals per decision epoch, and robot trajectories were simulated applying the velocity motion model from [50, Ch. 5.3].

For planning with POMCP (Algorithm 1), the control space was discretized uniformly to 9 linear velocity and 7 angular velocity values, resulting in a total of 63 control actions. The exploration bonus e was set to 50 for the maze and office environments and 100 for the outdoor environment. A total of $N_s = 3000$ simulation episodes were executed when planning each action.

For planning with SMC (Algorithm 2), the control space does not need to be discretized, but the kernels for sampling control signals must be specified. It is more likely that the robot obtains more information the greater the amount of unexplored cells covered by its sensors. Thus, to increase the likelihood of observing more unexplored cells, the initial control sequences for iteration $l = 1$ were sampled assigning higher probability to linear velocities near the robot's maximum velocity. Angular velocities were sampled uniformly at random. The number of particles was $M = 100$, and a threshold value of $M_t = M/4$ was set to trigger resampling. For iterations $l > 1$, we applied a Gaussian kernel with variance proportional to $(1/l^2)$ times the full range of possible linear and angular velocity values. Control actions violating the aforementioned maximum and minimum values for the velocities were rejected. A cooling schedule $\nu_l = 2l + 5$ was applied, with $l_{\max} = 7$.

During the simulations, control actions were checked by examining their corresponding trajectories. If the trajectory entered

[4] Robotics Data Set Repository Radish [49]. We acknowledge Cyrill Stachniss (and Giorgio Grisetti) for providing the office (outdoor) environment data.



(a) Maze.

(b) Office.

(c) Outdoor.

Fig. 5. Environments examined in the simulation experiments.

cells with occupancy probability greater than 0.2, the control action was rejected. The robot was also not allowed to move outside the map area or enter unknown areas (grey cells in Fig. 5).

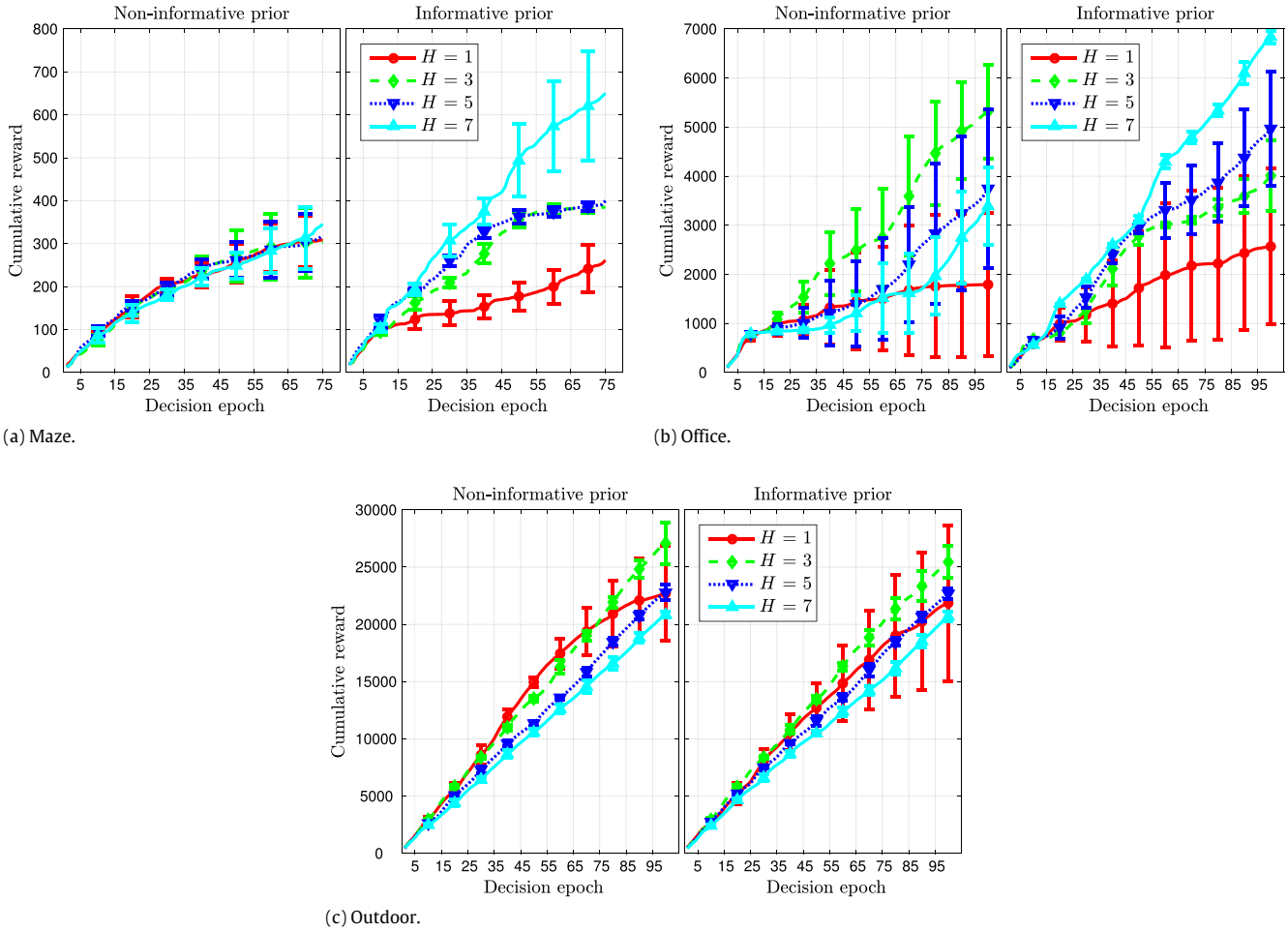
For prior information, a non-informative and an informative prior were considered. A non-informative prior corresponds to the case where the map area is initially unknown to the robot, with the exception of information provided by a single observation recorded at the robot's starting location. The informative prior was designed such that in the maze environment it corresponds to telling the robot the locations of all the maze walls (as indicated by the black cells in Fig. 5(a)) by assigning them occupancy probability of 0.99, and in the office and outdoor environments indicating the unknown area (the gray cells in Figs. 5(b) and (c)) by again assigning them an occupancy probability of 0.99. For each type of prior and each environment, the experiment was repeated five times.

6.2.2. Results

Fig. 6 shows comparisons of the cumulative mutual information reward collected as a function of the decision epoch when the POMCP algorithm was applied, in each of the environments studied and for optimization horizons $H = 1, 3, 5,$ and 7. The lines indicate the mean values, while the horizontal bars indicate the 95% confidence intervals of the mean values. For the non-informative prior, increasing the optimization horizon did not consistently result in an increased cumulative reward in any of the environments. This is due to the fact that in this case the map samples (and hence the corresponding observation samples) are drawn assuming a uniform prior on the map cells' occupancy, which poorly reflects the true configuration of the environment.

In contrast, for the informative prior, statistically significantly greater cumulative rewards were obtained in the maze and office environments when increasing H, as indicated by Figs. 6(a) and (b). A similar effect was however not observed in the outdoor environment (Fig. 6(c)). In the maze and office environments, there are dead-ends, i.e. locations where the robot has to travel through already-explored areas to reach new, unexplored areas. Choosing to traverse towards a dead-end may seem informative in the short term, but yields poor information gain later. Dead-ends may be avoided if sufficient prior information is available and can be exploited, i.e. the optimization horizon is great enough to indicate the presence of a dead-end. Turning a corner in the maze may severely limit the trajectory options available at subsequent decision epochs. In contrast, the outdoor environment is primarily open, and committing to a certain trajectory usually does not impose such limitations. The experimental results suggest that the effect of prior information on exploration performance is lesser in such an open environment.

Similar results were observed for the SMC algorithm, as shown for the office environment in Fig. 7. Here, the lines indicate the mean values, while the horizontal bars indicate the 95% confidence intervals of the mean values. Comparing the right panel of the figure to the right panel of Fig. 6(b), we note that for the informative prior the effect of increasing the horizon H for SMC does not seem



(a) Maze.

(b) Office.

(c) Outdoor.

Fig. 6. Cumulative reward in the maze, office, and outdoor environments as a function of optimization horizon H applying POMCP (Algorithm 1). The lines with the markers show the means over 5 simulation runs, while the horizontal bars indicate the 95% confidence intervals. In each subfigure, the left panels show results for a non-informative prior and right panels for an informative prior.

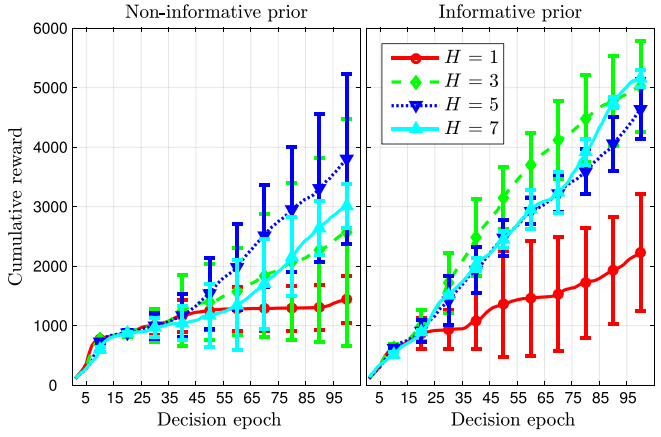


Fig. 7. Cumulative reward in the office environment applying SMC (Algorithm 2). The lines with the markers show the means over 5 simulation runs, while the horizontal bars indicate the 95% confidence intervals. The left panel shows results for a non-informative prior, and the left panel for an informative prior.

to be as significant as for POMCP. From the data, we determined that for POMCP the robot decided to move from its initial position at (38.5, 58.0) towards the large open area around coordinates (25, 50) (see Fig. 5(b) for the map with coordinate axes) in 1 out of 5 cases for $H = 5$, and in all of the five cases for $H = 7$. In contrast, for SMC, the robot instead moves towards the corridor around coordinates (45, 50) in 4 out of 5 cases for $H = 5$, and in 3 out of 5 cases for $H = 7$. Moving to the open area results in a much greater information gain over a long sequence of decisions, explaining the reason for the difference in favour of POMCP for $H = 7$. We do not believe this to be an indication of the superiority of the POMCP approach, but rather a byproduct of a statistical evaluation of the algorithms combined with the stochastic nature of the solution algorithms themselves.

Overall, the results of the experiments indicate that non-myopic planning ($H > 1$) is useful when the available prior information can be leveraged to find more informative local trajectories. This is the case in particular for the maze environment with an informative prior (Fig. 6(a)), and the office environment with an informative prior (Figs. 6(b) and 7).

The information the robot has about the map affects whether increasing the optimization horizon is useful. In the maze environment with a non-informative prior, the robot typically has information about the map only in its immediate vicinity as its view is blocked by nearby maze walls. Thus, most local trajectories, short or long, lead the robot outside this area, and increasing the optimization horizon is not useful if the prior information about these areas is not accurate (Fig. 6(a)). The office environment is more open, allowing the robot to observe the map in a larger area around it. The information gained in this way leads to improvements in exploration performance when increasing the optimization horizon, even in the case of an otherwise non-informative prior. Once the local trajectories considered are such that they bring the robot outside the area about which it has

accurate information, performance does not improve anymore: for instance, in the left panel of Fig. 6(b), this happens for $H > 3$. One further case where increasing the optimization horizon is not useful if the environment is such that all local trajectories are roughly equally informative. This is the case in the outdoor environment, which consists primarily of open areas (Fig. 6(c)).

6.3. Combining POMDP based and frontier based exploration

The proposed POMDP approach is capable of handling uncertainty in robot and environment states in a principled manner, allowing quantifiable trade-offs between uncertainty reduction and the cost of control actions. However, the optimization horizon has to be bounded to maintain computational feasibility. Thus only *local* reward information can be considered, leading to susceptibility to local minima.

A frontier-based exploration method, see e.g. [10], detects frontiers between free and unknown areas in the current map. One of the frontiers is selected as the exploration target, based on, e.g. the distance from the robot's current position to the frontier, or the size of the frontier. The robot is then commanded to move towards the selected target. Thus, frontier exploration can exploit *global* knowledge of frontiers towards unexplored areas over the whole map.

POMDP based exploration can fail when local information available within the optimization horizon is not sufficient to find an action with good exploration performance, for example if no unexplored area is reachable within the optimization horizon. To reduce the effect of these types of failures, we implemented an exploration method that combines POMDP based and frontier based exploration. The method applies POMDP based planning and executes the actions thus found until it reaches a situation where either (1) all local action sequences are below a given informativeness threshold, as measured by the expected total MI for them, or (2) all valid local action sequences correspond to trajectories with a length less than a given threshold value, indicating a possible dead-end. When either of the conditions triggers, a frontier exploration method is queried once for a frontier to be assigned as the next target for the robot. When this frontier is reached, the POMDP planning phase is again resumed.

We will argue that combining POMDP based and frontier based exploration in this way presents a stronger alternative to applying either approach alone. To support this argument, we executed a series of experiments comparing such an approach to only applying frontier exploration. The software used in this subsection was implemented in C++, and is available at https://goo.gl/ENGkIf.

6.3.1. Experimental setup

We chose to conduct the experiments in the office environment (Fig. 5(b)), as based on Section 6.2.2 increasing the optimization horizon H there improves performance both in the case of non-informative and informative prior. We implemented the method combining POMDP based and frontier based exploration applying the SMC algorithm and the basic frontier exploration algorithm as presented e.g. in [10]. The SMC method was chosen since it dynamically generates feasible control signals and trajectories, without need to manually define a fixed set of primitive control actions from which the trajectories are constructed. This method was then compared to only applying the basic frontier exploration method.

Each of the exploration experiments was repeated five times. Each repetition was terminated either at a timeout of 400 s, or if a failure happened that either caused the robot to get stuck, or the planner software to fail to produce a result. As in Section 6.2.1, the initial information provided to the robot consisted of a single observation recorded at the starting location, in addition to possible prior information.

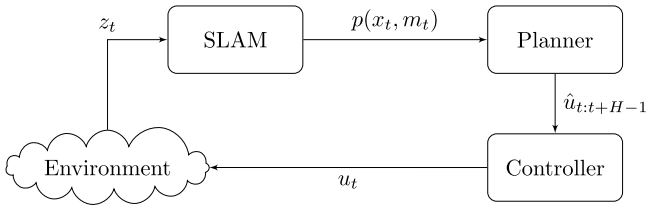


Fig. 8. A software implementation of the planner. The cloud-shaped block depicts the environment the robot is interacting with. The rectangular blocks indicate software modules, and arrows indicate propagation of signals between the modules, labelled by the mathematical symbol of the signal.

For the SMC algorithm, we set the number of particles to $M = 20$, and the resampling threshold to $M_t = M/4$. A cooling schedule $\nu_l = 2l + 5$ was applied, with $l_{\max} = 4$. The simulated robot and the kernels applied for sampling control actions were as described in Section 6.2.1. We applied maps with a resolution of 0.05 m per cell, and set the threshold for triggering frontier exploration at a total trajectory length of 0.5 m or expected MI of less than 50 bits. The 50 bit MI threshold corresponds to 50 completely unknown cells (occupancy probability 0.5) becoming completely known (occupancy probability 0 or 1), so, roughly speaking, if the robot expected to explore less than 0.125 square metres of new area, or move less than 0.5 m, it would trigger the frontier exploration method once instead of continuing with POMDP based exploration.

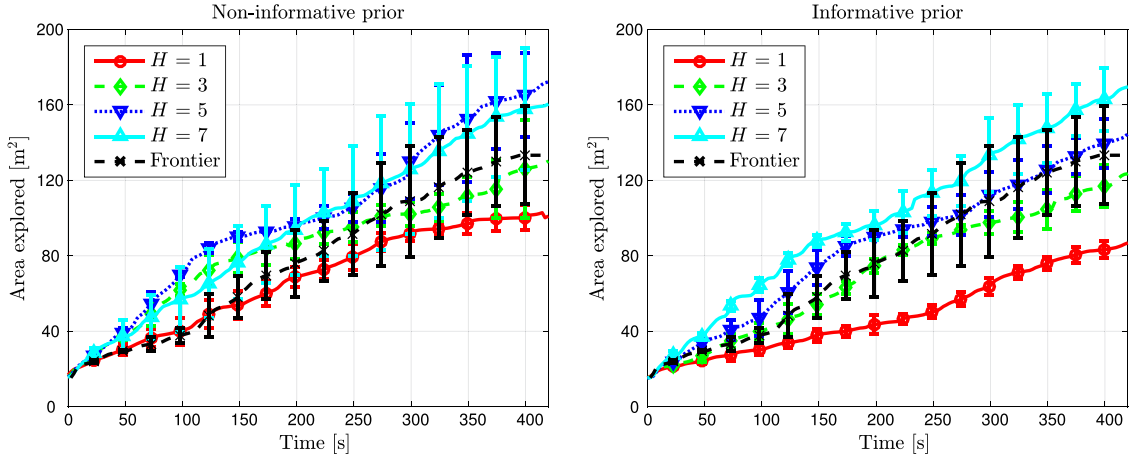
A conceptual overview of our software implementation is shown in Fig. 8. The robot is interacting with the environment, shown on the bottom left hand side of the figure. The outputs from the environment, i.e., observations z_t, are processed by the SLAM algorithm to revise the belief state $b_t = p(x_t, m_t)$. Based on the current belief state, the planner module shown on the top right of the figure computes an optimized sequence of control actions $\hat{u}_{t:t+H-1}$. A controller module shown on the bottom right of the figure decides which control action u_t to finally apply.

The simulation was implemented in the Stage robot simulator [51], integrated with the Robot Operating System (ROS) framework [52] where the exploration algorithms were implemented. The SLAM module we applied (see Fig. 8) was the RBPF SLAM algorithm based on [24]. As the frontier exploration method, we applied an open source implementation provided at http://wiki.ros.org/frontier_exploration.

We remark that there exist multi-robot frontier-exploration techniques that make use of prior information, e.g., in the form of semantic information about types of areas in the map [53,54]. Suitable applications for these methods include for example search and rescue, where semantic information about the types of rooms, e.g., office or lobby, can help guide the robots to promising search areas. However, in the experiments here, no such semantic information was available.

6.3.2. Results

Overall, in the 40 exploration runs applying the proposed method (5 runs for each of the 4 horizons, with 2 cases for prior information), we observed 10 failures with the experiment terminating before the timeout of 400 s. There were 8 cases of planner failure, either due to being unable to detect a frontier after trying POMDP based exploration, or due to inability to find a feasible path towards the requested exploration target. There were 2 cases in which the robot got stuck in the simulator after a collision. Among all failure cases, the earliest time of occurrence was at 195 s, while the average time of failure occurrence was at 273 s. In the 5 runs with the pure frontier exploration method, there was 1 failure at 197 s due to inability to find any frontiers.



(a) Non-informative prior.

(b) Informative prior.

Fig. 9. Comparison of POMDP based exploration (with optimization horizon H) combined with frontier based exploration with pure frontier exploration in the office environment. The lines with the markers show the mean area explored over 5 simulation runs, while the horizontal bars indicate the 95% confidence intervals. Fig. 9(a) shows the results for a non-informative prior, and Fig. 9(b) for an informative prior.

Since frontier exploration does not consider mutual information in selecting exploration targets, we present the comparison results in terms of the total area explored as a function of the time spent exploring. Fig. 9 shows the mean area explored and its 95% confidence interval over each of the five experiments, for each of the methods applied. The results shown take into account that not every experiment ran until the timeout of 400 s.

There seems not to be a significant difference in the area explored between the cases of non-informative or informative prior. However, we can see that the consistency of the results is better in case of the informative prior: the mean area explored increases monotonically as a function of H, and the confidence intervals are smaller than for the non-informative prior.

We note that with the exception of the myopic case $H = 1$, the proposed method does not perform significantly worse than pure frontier exploration during any time interval. It is interesting to note that although the area explored at the end of the experiment is not significantly greater for the proposed method than for pure frontier exploration, there are time intervals where the difference is significant in favour of the proposed method. Such intervals can be seen for example from 75 s to about 200 s for the non-informative prior, $H = 5$, and from 75 s to about 175 s for the informative prior, $H = 7$.

When we examined the trajectories the robot chose in each of these cases more closely, we discovered that this difference is primarily due to a different choice of initial exploration target. Recall that the robot started at coordinates $(38.5, 58.0)$ (see Fig. 5(b)). The proposed method, especially with an informative prior, favours moving towards the corridor at coordinates $(40, 50)$, as moving instead to the room at coordinates $(40, 60)$ is noted to quickly lead to a dead-end providing no further information gain. This is however not considered by frontier exploration, which favoured moving first to the frontier in the room, and then returning to explore other areas once the dead-end was discovered.

A further example of the usefulness of prior information can be seen in Fig. 2. The prior information indicates the outer edges of the environment, shown in the figure by the black lines bordering the unknown area. Initially, the trajectories sampled are distributed evenly leading towards the corridor on the left hand side of Fig. 2(a) and the area on the right hand side of the figure. Availability of prior information however indicates that a smaller unknown area will be visible when the robot moves to the left hand side corridor, compared to moving to the potentially large open area on the right hand side. As the total MI related to either trajectory choice is evaluated, the SMC algorithm eventually discards trajectories towards the corridor and converges on a trajectory bringing the robot towards the open area instead, as seen in Fig. 2(c). In cases with a non-informative prior, both trajectory options result in roughly equal expected MI.

Table 2
The average number of function calls in the proposed method over five experiments to either the POMDP based or the frontier based exploration method. Results are shown as a function of the optimization horizons H, and for both the case of non-informative and informative prior information.

	H	POMDP	Frontier
Non-informative prior	1	28.2	10.0
	3	22.2	5.0
	5	20.0	3.0
	7	23.2	3.4
Informative prior	1	25.2	5.0
	3	25.2	3.4
	5	19.6	2.4
	7	18.6	1.8

Since the proposed method combines POMDP based and frontier based exploration, it is illustrative to consider how often either of the exploration techniques is applied. Table 2 indicates the average number of times the proposed method applied either POMDP based or frontier based exploration to select the next target where the robot should move to explore the environment. In the majority of the cases, POMDP based exploration is preferred. As expected, shorter optimization horizons H lead the robot more frequently to situations where no informative local trajectories can be found, and subsequently frontier exploration is applied more frequently. We also note that applying the informative prior seems to result in less calls to frontier exploration, indicating further the usefulness of prior information for POMDP based exploration. Overall, there is a slightly decreasing trend in the total number of calls to either method as a function of H, since the robot tends to traverse a longer trajectory before considering the next exploration target for longer optimization horizons.

Based on the experimental results, our proposed method can outperform frontier based exploration in terms of area explored in the initial part of exploration when the environment is still largely unknown. We note this is e.g., due to the ability of the proposed method to avoid dead-ends. Complete exploration of an environment is the goal in many applications, meaning that also dead-ends should eventually be explored.

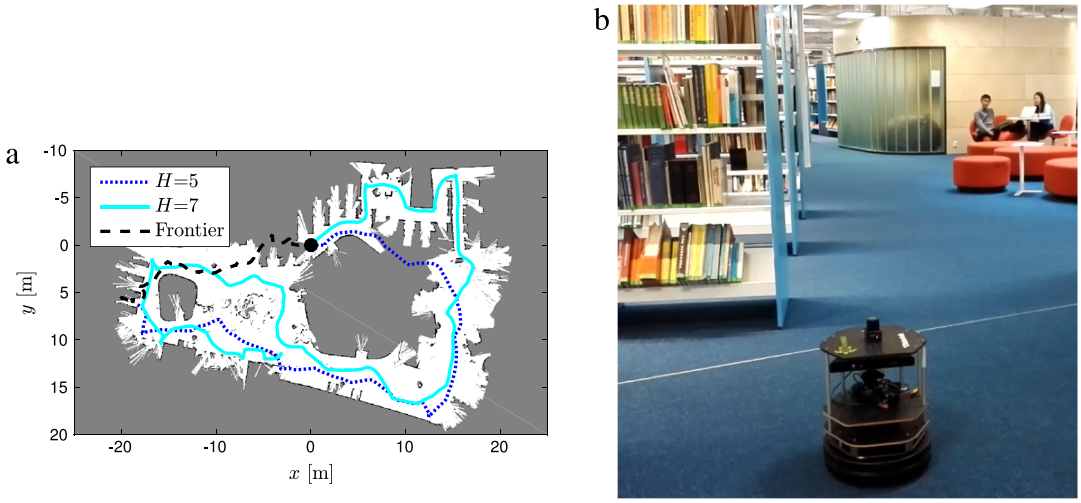


Fig. 10. (a) A partial map of the campus library environment. Occupied, free, and unknown areas are indicated by black, white, and gray cells, respectively. The robot's starting location is indicated by a filled black circle marker. Overlaid on the map are examples of typical trajectories taken applying the proposed exploration method or frontier based exploration. The map shown corresponds to the data collected while traversing the trajectory for $H = 7$. The unknown areas through which the other trajectories travel were observed to be free in the corresponding experiments, but not shown here. (b) A view of the environment from the robot's starting location. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

We note that although the proposed method combining POMDP based and frontier based exploration will eventually achieve this, the strategy it applies may not be optimal: considering the objective of complete exploration, it might be worthwhile to explore nearby dead-ends immediately rather than returning to them later after exploring other parts of the environment. When this strategy can be improved upon depends at least on the availability of prior information, and remains to be studied more carefully. In conclusion, we believe the proposed method is preferable in applications where quickly exploring the local environment for a high information gain is required, and completeness of exploration in the short term is not crucial.

7. Real-world exploration

To verify the feasibility of applying our proposed planning approach in a real application, in this section we present results on exploration tasks in a real environment. The experimental setup is described in Section 7.1, and results are reported in Section 7.2.

7.1. Experimental setup

The experiments were executed at a university campus library. A partial map of the environment with the robot's starting location indicated is shown in Fig. 10(a). The environment features open areas and narrow corridors between bookshelves. In all experiments, the robot started at the location indicated by the black circle marker in the figure. A view of the environment showing the robot at its starting location is shown in Fig. 10(b). The photograph was taken such that it shows the environment towards the negative values of the x-axis in the map of Fig. 10(a). The robot had no prior information about the environment beyond a single observation recorded at the starting location.

The experiments were carried out with a robot such as described in Section 6.2.1. The robot's velocities were equal to those of the simulated robot of Section 6.2.1. The robot was equipped with a laser range finder with a maximum range of 4 m. The robot can also be seen in Fig. 10(b). The software setup and algorithm parameters were as described in Section 6.3.1. The robot's on-board computer, equipped with an Intel i7-4500U multicore processor, 4 GB of RAM and running a Linux operating system, was applied to run all of the required software. We applied the proposed exploration method combining POMDP based and frontier based exploration, with the software implementation described in Section 6.3, and a frontier based exploration method. Optimization horizons $H = 5$ and $H = 7$ were applied in the POMDP based exploration method. With each method, the experiment was repeated 5 times, and each experiment ran until a timeout of 400 s or until a failure caused termination of the experiment.

Table 3
The minimum, average and maximum planning times for the SMC algorithm as a function of the optimization horizon H.

H	Planning time [s]		
	Min	Avg	Max
5	2.60	4.24	6.14
7	2.03	5.21	7.61

7.2. Results

For the proposed method, we observed 4 failures with $H = 5$ occurring at 240, 255, 265, and 376 s, and 1 failure with $H = 7$ at 390 s. With frontier exploration, we observed 2 failures at 355 and 380 s. The failures were due to the robot colliding with an undetected obstacle and getting stuck, or the planner failing to find an exploration target. Failures to find an exploration target most often happened at a narrow corridor, where the robot's sensors erroneously indicated that there was no possible path without a collision to exit the corridor.

The minimum, average and maximum planning times required for the SMC algorithm are reported as a function of the optimization horizon H in Table 3. Longer trajectories and larger open areas tend to increase the amount of map cells for which an occupancy value must be sampled to obtain an estimate of the MI, thus increasing the required planning time. The shortest planning times were observed when the robot was near a wall or in a narrow corridor. During the planning time, the robot remained in place.

A summary of the experimental results is presented in Fig. 11, showing the mean area explored and its 95% confidence interval as a function of time, for each of the exploration methods considered. For the proposed method with $H = 5$, results are only shown up to around 265 s, as only two of the experiments ran for a longer

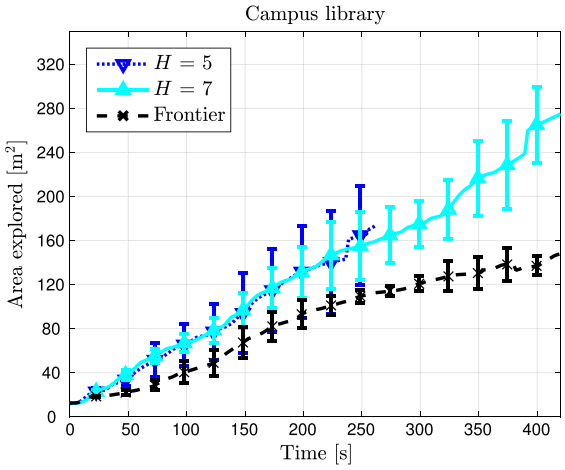


Fig. 11. Area explored in the campus library environment as a function of time. Results are shown for the proposed method with optimization horizon $H = 5$ or $H = 7$, and for frontier exploration. The lines with the markers show the mean area explored over 5 runs, while the horizontal bars indicate the 95% confidence intervals.

time. In all cases, the confidence intervals drawn take into account the number of active experiments remaining. The results suggest that in this environment, it is beneficial to apply the proposed exploration approach instead of frontier exploration. We observed the proposed approach with $H = 7$ to result in a significantly greater total area explored than frontier exploration after 200 s.

To understand why this is the case, it is useful to study the trajectories traversed by the robot applying each of the exploration methods. Fig. 10(a) shows typical examples of trajectories for each of the methods. Frontier exploration, indicated by the black dashed line, consistently chose to move towards the negative x axis from the starting location. In this direction, there was a row of bookshelves aligned as shown in Fig. 10(b). As indicated by the curves in the initial part of the trajectory, frontier exploration would find the nearest frontier behind the corner of the closest bookshelf to the robot. The robot would then move to it, and repeat in a similar manner until reaching the end of the row of shelves. Frontier exploration does not attempt to quantify the amount of information that may be gained by such a strategy, but rather deterministically moves the robot towards the closest frontier found.

In contrast, we observed the proposed method to adopt a different strategy. Since the narrow corridors between bookshelves can often be observed before it is necessary to move into them, this information may be applied to select more informative trajectories. The robot would generally prefer to avoid narrow corridors, as the proximity of the walls made it unlikely that moving there would provide as much information as moving towards a potentially open area. Corridors were typically only preferred when there was no alternative, such as shown in the trajectory for $H = 7$ indicated by the cyan line in the top right part of Fig. 10(a). Trajectories with $H = 5$ were similar, as indicated by the blue line in Fig. 10(a) and the results in Fig. 11. Based on similar results in the previous experiments, we thus believe that increasing the optimization horizon further would not result in better performance in this environment.

We remark that the results could be different if the robot's starting position were to be changed, as the location to explore first would be different. For the proposed method, this is due to a different set of feasible local trajectories considered. For pure frontier exploration, the first location to explore depends on the map information obtained from sensor readings at the starting position. If the starting position is changed such that the map information remains roughly the same, the same exploration location would likely be selected.

Overall, the experiments show that in the environment studied, combining non-myopic POMDP based planning with frontier exploration improves performance over pure frontier exploration. Although frontier exploration can apply all global information to select the next exploration target, it performs the selection in a heuristic manner, ignoring the expected information gain. POMDP based methods can evaluate the information available locally within the optimization horizon to select exploration targets that lead to quantifiably greater expected information gain. The weakness of the method, susceptibility to local minima due to the finite horizon, can be alleviated by combining it with traditional frontier exploration.

8. Conclusion

We formulated a robotic exploration problem as a partially observable Markov decision process (POMDP), applying mutual information as reward function. We solved an open loop approximation to the POMDP applying forward simulation and the receding horizon control principle. We derived a new sampling-based approximation for mutual information, and presented an efficient method to draw samples for the approximation when an occupancy grid map representation is applied.

The usefulness of non-myopic decision-theoretic planning for exploration was demonstrated in simulation and real world experiments. Non-myopic planning can help avoid situations where initial information gain related to a control action seems high, but ultimately the action leads to a dead-end where further exploration is not possible. However, basing exploration decisions purely on a finite horizon look-ahead was found to perform poorly in situations where a sequence of control actions longer than the look-ahead horizon is required to reach an unexplored area. To alleviate this susceptibility to local minima, we suggested combining POMDP based and frontier based exploration. Experimental results show that, depending on the environment to be explored, this combination can improve exploration performance when compared to only applying frontier exploration.

There are two main weaknesses in our approach. First, an open loop approximation was required to deal with the very large and possibly continuous action and observation spaces, resulting in a performance loss compared to the optimal closed loop solution. Secondly, to increase computational effectiveness we ignore the SLAM problem while solving the open loop control problem. Thus if the possibility of losing the consistency of the real process SLAM filter cannot be ignored [25,9], actions that lead to such a failure state may be selected.

More prior information about the environment was noted to improve exploration performance. In future work, a database of typical map features for different types of environments, for instance, office, outdoor, etc. could be maintained, and map samples drawn from the database instead, see e.g. [55]. As forward simulation methods are applicable to very general underlying models, more realistic environment models that lift the assumption of spatial independence between occupancy grid cells could be applied, for instance higher order Markov random fields [56].

Acknowledgements

We thank Mr. Joonas Melin and Mr. Eero Heinänen for their support in executing the experimental work. This work was partly supported by the TUT Graduate School and TUT Strategic funding for robotics and intelligent machines.

References

[1] L. Kaelbling, M. Littman, A. Cassandra, Planning and acting in partially observable stochastic domains, Artificial Intelligence 101 (1–2) (1998) 99–134.

[2] C. Stachniss, G. Grisetti, W. Burgard, Information gain-based exploration using Rao-Blackwellized particle filters, in: Proc. Robotics: Science and Systems Conf., RSS, Cambridge, MA, USA, 2005.

[3] G. Hitz, A. Gotovos, M. Eve, A. Krause, R.Y. Siegwart, Fully autonomous focused exploration for robotic environmental monitoring, in: Proc. IEEE Intl. Conf. on Robotics and Automation, ICRA, Hong Kong, China. ISBN 9781479936847, 2014, pp. 2658–2664.

[4] B. Charrow, V. Kumar, N. Michael, Approximate representations for multi-robot control policies that maximize mutual information, Auton. Robots 37 (4) (2014) 383–400. http://dx.doi.org/10.1007/s10514-014-9411-2.

[5] N. Atanasov, J. Le Ny, K. Daniilidis, G. Pappas, Information acquisition with sensing robots: Algorithms and error bounds, in: Proc. IEEE Int. Conf. on Robotics and Automation, ICRA, Hong Kong, China, 2014, pp. 6447–6454.

[6] M. Lauri, R. Ritala, Optimal sensing via multi-armed bandit relaxations in mixed observability domains, in: Proc. IEEE International Conference on Robotics and Automation, ICRA, Seattle, WA, 2015, pp. 4807–4812. http://dx.doi.org/10.1109/ICRA.2015.7139867.

[7] C.H. Papadimitriou, J.N. Tsitsiklis, The complexity of Markov decision processes, Math. Oper. Res. 12 (1987) 441–450. http://dx.doi.org/10.1287/moor.12.3.441.

[8] H. Durrant-Whyte, T. Bailey, Simultaneous localisation and mapping (SLAM): Part I: The essential algorithms, IEEE Robot. Autom. Mag. 13 (2) (2006) 99–110.

[9] L. Carlone, J. Du, M. Ng, B. Bona, M. Indri, An application of Kullback–Leibler divergence to active SLAM and exploration with particle filters, in: Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS), IEEE, Taipei, Taiwan, ISBN: 978-1-4244-6674-0, 2010, pp. 287–293. http://dx.doi.org/10.1109/IROS.2010.5652164.

[10] B. Yamauchi, A frontier-based approach for autonomous exploration, in: Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97 'Towards New Computational Principles for Robotics and Automation', 1997, pp. 146–151. http://dx.doi.org/10.1109/CIRA.1997.613851.

[11] H.H. Gonzalez-Banos, J.C. Latombe, Navigation strategies for exploring indoor environments, Int. J. Robot. Res. 21 (10–11) (2002) 829–848. http://dx.doi.org/10.1177/027836490202101083 4.

[12] M. Juliá, A. Gil, O. Reinoso, A comparison of path planning strategies for autonomous exploration and mapping of unknown environments, Auton. Robots 33 (2012) 427–444. http://dx.doi.org/10.1007/s10514-012-9298-8.

[13] F. Amigoni, Experimental evaluation of some exploration strategies for mobile robots, in: Proc. IEEE Int. Conf. on Robotics and Automation, (ICRA), IEEE, Pasadena, CA, ISBN: 9781424416479, 2008, pp. 2818–2823. http://dx.doi.org/10.1109/ROBOT.2008.4543637.

[14] F. Amigoni, V. Caglioti, An information-based exploration strategy for environment mapping with mobile robots, Robot. Auton. Syst. 58 (5) (2010) 684–699. http://dx.doi.org/10.1016/j.robot.2009.11.005.

[15] R. Sim, N. Roy, Global a-optimal robot exploration in SLAM, in: Proc. IEEE Int. Conf. on Robotics and Automation, (ICRA), IEEE, Barcelona, Spain, ISBN: 078038914X, 2005, pp. 661–666. http://dx.doi.org/10.1109/ROBOT.2005.1570193.

[16] S. Huang, N.M. Kwok, G. Dissanayake, Q.P. Ha, G. Fang, Multi-step look-ahead trajectory planning in SLAM: Possibility and necessity, in: Proc. IEEE Intl. Conf. on Robotics and Automation, (ICRA), IEEE, Barcelona, Spain, ISBN: 078038914X, 2005, pp. 1091–1096. http://dx.doi.org/10.1109/ROBOT.2005.1570261.

[17] T. Kollar, N. Roy, Trajectory optimization using reinforcement learning for map exploration, Int. J. Robot. Res. 27 (2) (2008) 175–196. http://dx.doi.org/10.1177/0278364907087426.

[18] B. Tovar, L. Muñoz-Gómez, R. Murrieta-Cid, M. Alencastre-Miranda, R. Monroy, S. Hutchinson, Planning exploration strategies for simultaneous localization and mapping, Robot. Auton. Syst. 54 (4) (2006) 314–331. http://dx.doi.org/10.1016/j.robot.2005.11.006.

[19] R. Martinez-Cantin, N. De Freitas, E. Brochu, J. Castellanos, A. Doucet, A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot, Auton. Robots 27 (2) (2009) 93–103. http://dx.doi.org/10.1007/s10514-009-9130-2.

[20] V. Indelman, L. Carlone, F. Dellaert, Planning under uncertainty in the continuous domain: a generalized belief space approach, in: Proc. IEEE Intl. Conf. on Robotics and Automation, ICRA, Hong Kong, China. ISBN 9781479936847, 2014, pp. 6763–6770. http://dx.doi.org/10.1109/ICRA.2014.6907858.

[21] V. Indelman, L. Carlone, F. Dellaert, Planning in the continuous domain: a generalized belief space approach for autonomous navigation in unknown environments, Int. J. Robot. Res. (2015) http://dx.doi.org/10.1177/0278364914561102.

[22] H. Moravec, Sensor fusion in certainty grids for mobile robots, AI Mag. 9 (2) (1988) 61–74.

[23] F. Bourgault, A. Makarenko, S. Williams, B. Grocholsky, H. Durrant-Whyte, Information based adaptive robotic exploration, in: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, (IROS), IEEE, Lausanne, Switzerland, ISBN: 0-7803-7398-7, 2002, pp. 540–545. http://dx.doi.org/10.1109/IRDS.2002.1041446.

[24] G. Grisetti, C. Stachniss, W. Burgard, Improved techniques for grid mapping with Rao-Blackwellized particle filters, IEEE Trans. Robot. 23 (1) (2007) 34–46. http://dx.doi.org/10.1109/TRO.2006.889486.

[25] J. Blanco, J. Fernandez-Madrigal, J. Gonzalez, A novel measure of uncertainty for mobile robot SLAM with Rao Blackwellized particle filters, Int. J. Robot. Res. 27 (1) (2008) 73–89. http://dx.doi.org/10.1177/0278364907082610.

[26] M.L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, John Wiley & Sons, Inc., New York, NY, 1994.

[27] M.H. DeGroot, Optimal Statistical Decisions, in: Wiley Classics Library Edition, John Wiley & Sons, Inc., Hoboken, New Jersey, 2004.

[28] T. Cover, J. Thomas, Elements of Information Theory, second ed., John Wiley & Sons, ISBN: 9780471241959, 2006.

[29] D.P. Bertsekas, Dynamic Programming and Optimal Control, Vol. 1, Athena Scientific, Belmont, MA, 1995.

[30] R.D. Smallwood, E.J. Sondik, The optimal control of partially observable Markov processes over a finite horizon, Oper. Res. 21 (5) (1973) 1071–1088.

[31] M. Hauskrecht, Value-function approximations for partially observable Markov decision processes, J. Artificial Intelligence Res. 13 (1) (2000) 33–94.

[32] M.T. Spaan, N.A. Vlassis, Perseus: Randomized point-based value iteration for POMDPs, J. Artificial Intelligence Res. 24 (2005) 195–220.

[33] J. Pineau, G. Gordon, S. Thrun, Anytime point-based approximations for large POMDPs, J. Artificial Intelligence Res. 27 (1) (2006) 335–380.

[34] H. Kurniawati, D. Hsu, W. Lee, SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces, in: Proc. Robotics: Science and Systems Conf. (RSS), Zürich, Switzerland, 2008.

[35] M. Araya, O. Buffet, V. Thomas, F. Charpillet, A POMDP extension with belief-dependent rewards, in: Advances in Neural Information Processing Systems, Vol. 23, Vancouver, Canada, 2010, pp. 64–72.

[36] S. Ross, J. Pineau, S. Paquet, B. Chaib-Draa, Online planning algorithms for POMDPs, J. Artificial Intelligence Res. 32 (2008) 663–704.

[37] D. Bertsekas, Dynamic programming and suboptimal control: A survey from ADP to MPC, Eur. J. Control 11 (4–5) (2005) 310–334. http://dx.doi.org/10.3166/ejc.11.310-334.

[38] C. Yu, B. Gerkey, J. Chuang, G.J. Gordon, A. Ng, Open-loop plans in multi-robot POMDPs. Tech. Rep., Stanford University, CS Dept., 2005.

[39] E. Chong, R. Givan, H.S. Chang, A framework for simulation-based network control via hindsight optimization, in: Proc. 39th IEEE Conf. on Decision and Control, Vol. 2, Sydney, Australia, 2000, pp. 1433–1438. http://dx.doi.org/10.1109/CDC.2000.912059.

[40] D. Silver, J. Veness, Monte-Carlo planning in large POMDPs, in: Advances in Neural Information Processing Systems, Vol. 23, Vancouver, Canada, 2010, pp. 2164–2172.

[41] C.B. Browne, E. Powley, D. Whitehouse, S.M. Lucas, P.I. Cowling, P. Rohlfshagen, et al., A survey of Monte Carlo tree search methods, IEEE Trans. Comput. Intell. AI Games 4 (1) (2012) 1–43. http://dx.doi.org/10.1109/TCIAIG.2012.2186810.

[42] P. Auer, N. Cesa-Bianchi, P. Fischer, Finite-time analysis of the multiarmed bandit problem, Mach. Learn. 47 (2002) 235–256. http://dx.doi.org/10.1023/A:1013689704352.

[43] L. Kocsis, C. Szepesvári, Bandit based Monte-Carlo planning, in: J. Fürnkranz, T. Scheffer, M. Spiliopoulou (Eds.), Machine Learning: ECML 2006, in: Lecture Notes in Computer Science, vol. 4212, Springer, Berlin, Heidelberg, ISBN: 978-3-540-45375-8, 2006, pp. 282–293.

[44] N. Kantas, J. Maciejowski, A. Lecchini-Visintini, Sequential Monte Carlo for model predictive control, in: L. Magni, D. Raimondo, F. Allgöwer (Eds.), Nonlinear Model Predictive Control, in: Lecture Notes in Control and Information Sciences, vol. 384, Springer, Berlin, Heidelberg, ISBN: 978-3-642-01093-4, 2009, pp. 263–273.

[45] A.M. Johansen, A. Doucet, M. Davy, Particle methods for maximum likelihood estimation in latent variable models, Stat. Comput. 18 (1) (2008) 47–57. http://dx.doi.org/10.1007/s11222-007-9037-8.

[46] P. Del Moral, A. Doucet, A. Jasra, Sequential Monte Carlo samplers, J. R. Stat. Soc. Ser. B Stat. Methodol. 68 (3) (2006) 411–436. http://dx.doi.org/10.1111/j.1467-9868.2006.00553.x, arXiv:0212648v1.

[47] J. Liu, R. Chen, Sequential Monte Carlo methods for dynamic systems, J. Amer. Statist. Assoc. 93 (443) (1998) 1032–1044. http://dx.doi.org/10.1080/01621459.1998.10473765.

[48] C.P. Robert, G. Casella, Monte Carlo Statistical Methods, Springer-Verlag, Inc., New York, NY, 1999.

[49] A. Howard, N. Roy, The Robotics Data Set Repository (Radish), 2003. URL: http://radish.sourceforge.net/.

[50] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, The MIT Press, Cambridge, MA, 2006.

[51] R. Vaughan, Massively multi-robot simulation in stage, Swarm Intell. 2 (2–4) (2008) 189–208.

[52] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, et al. ROS: an open-source robot operating system, in: ICRA Workshop on Open Source Software, 2009.

[53] C. Stachniss, Ó. Martínez Mozos, W. Burgard, Efficient exploration of unknown indoor environments using a team of mobile robots, Ann. Math. Artif. Intell. 52 (2) (2009) 205–227. http://dx.doi.org/10.1007/s10472-009-9123-z.

[54] A. Quattrini Li, R. Cipolleschi, M. Giusto, F. Amigoni, A semantically-informed multirobot system for exploration of relevant areas in search and rescue settings, Auton. Robots 40 (4) (2015) 581–597. http://dx.doi.org/10.1007/s10514-015-9480-x.

[55] D.P. Strom, F. Nenci, C. Stachniss, Predictive exploration considering previously mapped environments, in: Proc. IEEE International Conference on Robotics and Automation, ICRA, Seattle, WA, USA. ISBN 978-1-4799-6923-4, 2015, pp. 2761–2766. http://dx.doi.org/10.1109/ICRA.2015.7139574.

[56] B. Nabbe, M. Hebert, Extending the path-planning horizon, Int. J. Robot. Res. 26 (10) (2007) 997–1024. http://dx.doi.org/10.1177/0278364907084100.

Mikko Lauri Mikko Lauri received the master's degree in electrical engineering and the doctoral degree in automation science from Tampere University of Technology (TUT), Tampere, Finland, in 2010 and 2016, respectively. His research interests include active sensing, optimal control for information acquisition, and decision-making under uncertainty.

Risto Ritala received the master's and doctoral degrees in engineering physics from Helsinki University of Technology, Espoo, Finland, in 1981 and 1986, respectively. Since 2003, he has been a Professor of measurement information technology with the Department of Automation Science and Engineering, Tampere University of Technology, Tampere, Finland. His research interests are optimization of the operation of sensing degrees of freedom in autonomous mobile robots and related estimation, prediction, data analysis, modelling, and optimal control.