Auton Robot (2018) 42:329–351
https://doi.org/10.1007/s10514-017-9616-2

Near-optimal probabilistic search using spatial Fourier sparse set

Kuo-Shih Tseng[1] · Bérénice Mettler[2]

Received: 29 February 2016 / Accepted: 11 January 2017 / Published online: 6 February 2017

Abstract Autonomous search is an essential research topic for rescue and other robotic applications. However, searching for targets efficiently is still an unsolved problem. To achieve this objective, a robot needs to simultaneously maximize environmental coverage, maximize probability of detection (PD) and minimize motion cost. The problems associated with these objectives are NP-hard. This research reformulates the three objective functions as a maximum cumulative PD problem with motion cost. Since the PD function depends on the environment, the robot needs to both learn the PD function and the cost-to-go (CTG) function. This research proposes a reinforcement learning algorithm to learn the PD and CTG functions simultaneously. Since the PD function is sparse in the Fourier domain under certain subgoal patterns, spatial Fourier sparse set is proposed to learn PD functions based on the compressed sensing technique. The learned PD and CTG functions can then be used to generate subgoals that achieve $(1 - 1/e)$ of the optimum due to the submodularity. Experiments conducted with this algorithm demonstrate that the robot can search for the target faster than prior learning approaches (e.g., PMAC and FSS) and the benchmark model (e.g., PD).

This is one of several papers published in *Autonomous Robots* comprising the Special Issue on Active Perception.

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10514-017-9616-2) contains supplementary material, which is available to authorized users.

✉ Bérénice Mettler
mettler@umn.edu

Kuo-Shih Tseng
kuoshih@cs.umn.edu

[1] Department of Computer Science and Engineering, College of Science and Engineering, University of Minnesota, Minneapolis, MN, USA

[2] Department of Aerospace Engineering and Mechanics, College of Science and Engineering, University of Minnesota, Minneapolis, MN, USA

Keywords Probabilistic search · Submodularity · Compressed sensing · Q-learning · Sparse learning

1 Introduction

Optimal search is concerned with simultaneously maximizing environment coverage, maximizing the probability of target detection and minimizing motion cost. However, each one of the three objective functions is NP-hard. Most of the related work focuses on only one of the objective functions. If the map is preprocessed as a graphical model or polygonal model, the robot can search for the target with theoretical bounds of time or probability (Hollinger et al. 2010; Gerkey et al. 2006; Lau et al. 2008, 2006). However, those algorithms do not consider sensing coverage, sensing uncertainty, and motion constraints simultaneously. In Tseng and Mettler (2015), the proposed algorithm considers three objective functions. Although it can give theoretical guarantees for two objective functions, it cannot give the overall search performance guarantee. To achieve overall performance guarantee, a unified objective function is necessary.

The coverage problem and probabilistic search in a graphical model can be solved by greedy algorithms (Nemhauser et al. 1978; Hollinger and Singh 2008). Since the objective functions of coverage and probabilistic search are submodular (see Definition S1 and S2), greedy algorithms give near-optimal guarantees (Nemhauser et al. 1978; Feige 1998). If the coverage problem can be reformulated as a probabilistic search problem, the objectives are reduce to maximize probability of detection (PD) and minimize motion cost. Then further consider the motion cost, maximizing PD with motion cost is a unified objective function. The problem definition and approach overview are in the following two sections.

Definition S1: (Submodularity) Given a finite set S = {1,2,...,N}, a submodular function is a set function $F : 2^N \rightarrow \mathbb{R}$ which satisfies the diminishing return property. For every $S_A, S_B \subseteq S$ with $S_A \subseteq S_B$ and every $s \subseteq S$, $F(S_A \cup s) - F(S_A) \geq F(S_B \cup s) - F(S_B)$ holds.

Definition S2: (Greedy algorithm) Given a finite set S = {1,2,...,N} and coverage function F, the greedy algorithm starts with the empty set $S_{G,0} = \emptyset$ and choose $S_{G,i+1} = S_{G,i} \cup \{\arg\max_{s \in S \setminus S_{G,i}} F(S_{G,i} \cup \{s\})\}$, where $S_{G,i}$ represents the 1st to i-th chosen elements by the greedy algorithm and and s is an added new element.

1.1 Problem definition

The efficient search problem can be defined as follows: Given a grid map, a mobile robot with velocity and accelation constraints ($[v, \omega, a, \alpha]_{max}$), a kinect sensor with a conditional probability table[1] of target detection, the objective is to find velocity commands $[v, \omega]_{0:T}$ such that T is minimal, where 0:T denotes the time from the beginning to the time when $P(A) > 90\%$, where A denotes High probability area (H-area[2]).

Figure 1 illustrates an efficient search scenario for a single robot, with position described by a grid map, that has to generate a search path to find a target. Since the detection outcome is with uncertainty, a Bayes filter is applied to estimate the probability of target detection in each cell. Once the $P(A) > 90\%$, the search is terminated. In this paper, the research focuses on determining a search path given as a sequence of subgoals. The robot is controlled via speed and turn rate (v, ω) commands which are generated based on the current subgoal and robot position using receding horizon control (Mettler et al. 2010; Tseng and Mettler 2015).

1.2 Approach overview

The general approach illustrated in Fig. 2a, b, is that search with a mobile robot platform can be decomposed into subproblems that are represented by finite subgoals. the subgoal

[1] The conditional probability table is computed based on the detection outcome. There are four cases: true positive, false negative, false positive, and true negative. For example, if the target is there and the sensor cannot detect, this is called false negative.

[2] The H-area is defined as the $20 \times 20\,\text{cm}^2$ area around the highest probability cell. If the target is in the H-area and $P(A) > 90\%$, it is a positive decision.

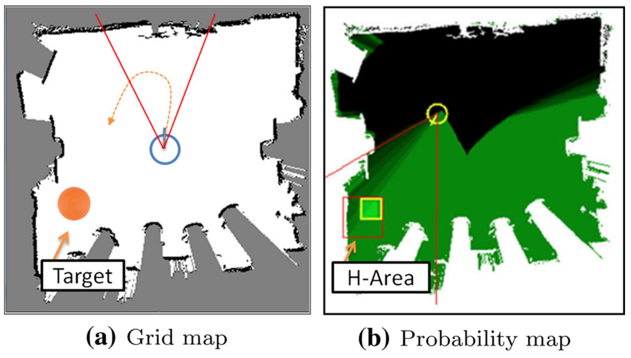


Fig. 1 Illustration of efficient search. **a** The *blue circle* represents the robot position, *red lines* are the sensing range, *black areas* are obstacles and *white areas* are unoccupied grids. **b** The *yellow circle* represents the robot position, the *black areas* represents low probability, *green areas* represents high probability and the *red square* is H-area

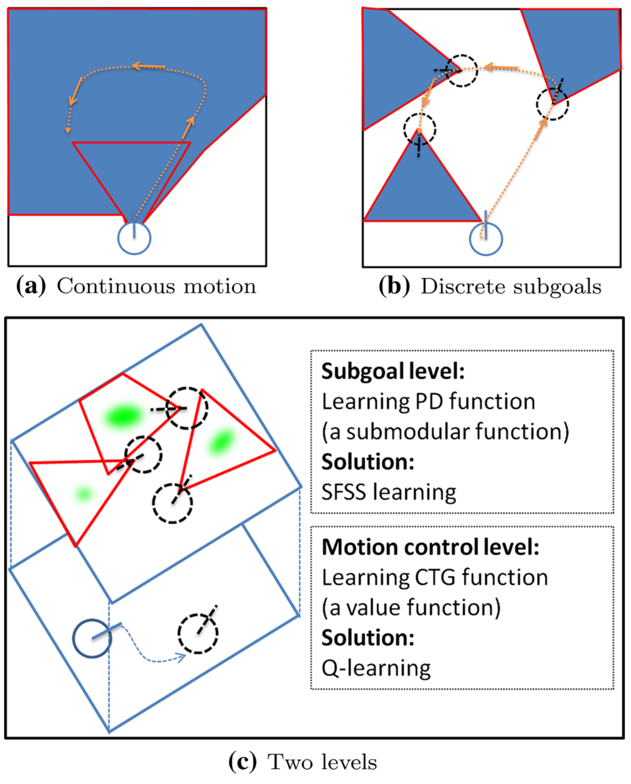


Fig. 2 Illustration of the subgoal concept. **a** The *blue circle* represents the current robot position, the *orange arrow* is search path, the *blue area* is covered area. The *triangle* represents the current coverage and *polygon areas* represent the coverage along the search path. **b** The *black circles* represent subgoal positions and *blue areas* are corresponding covered areas of each subgoal. **c** The *blue circle* represents the robot position, *black circles* are subgoal positions, *green areas* represents high probability, the *blue arrow* is the lowest cost path toward the subgoal

concept confers two main advantages. First, finding K subgoals with maximum coverage is a discrete optimization problem, which has solutions with theoretical guarantees (Nemhauser et al. 1978). Second, since the subgoals are the subset of the continuous motion, the covered area of continuous motion is always greater than the covered area of the subgoals (see the polygon area in Fig. 2a is bigger than the three triangles area in Fig. 2b). Hence, the covered area based on discrete subgoals is a lower bound of the covered area of the continuous motion. A key question then is, can such subgoals be learned? If the robot can learn the subgoals from information gathered through environment sensing, it can then sequentially visit the subgoals to cover the environment efficiently.

The goal of this research is to enable a mobile robot to learn the subgoals and then utilize those subgoals to search for the target (see Fig. 2c). At the motion control level, the robot learns the cost-to-go (CTG) function using Q-learning. The learned CTG function guides the robot to the lowest cost path toward the current subgoal. At the subgoal level, the robot learns the PD function through spatial Fourier sparse set (SFSS) learning. Since the search objective function is submodular, the subgoal selection based on a greedy policy is near-optimal with lower bound $(1 - 1/e)OPT$. After generating subgoals, the robot sequentially visits the learned subgoals to search for the target efficiently.

The contributions of this paper are as follows: First, a unified objective function considering coverage, probability of detection and motion cost is proposed. To the best of our knowledge, this is the first unified objective function considering these three objectives. Second, the proposed SFSS is able to learn a submodular function with m samples, where $m \leq O(klog(b))$, b is the Fourier basis number and k is the number of nonzero Fourier coefficients. To the best of our knowledge, the sparsity in the Fourier domain for learning submodular functions have not been investigated. The SFSS is the first algorithm exploiting this concept to dramatically reduce the computation. Third, the learned subgoals are shown to be within $(1 - 1/e)$ of the optimum with high probability. Finally, experimental demonstrations show that the robot can search for the target faster than prior learning approaches (e.g., PMAC and FSS) and the benchmark model (e.g., PD).

The paper is organized as follows. Section 2 describes the related work. Section 3 introduces the problem formulation. Sections 4 and 5 describe how the PD and CTG functions are learned and detail the proposed algorithm. Section 6 outlines the extensions of the proposed algorithms and the future work. Section 7 describes the experiments used to demonstrate its performance. Finally, Sect. 8 concludes the paper with a summary of the work.

2 Relevant work

The following discusses the three problems that make up optimal search problem: coverage, probabilistic search and path planning.

2.1 Coverage problem

For surveillance applications, a typical goal is to cover the whole room using the fewest sensors, or cover the most space using a fixed number of sensors. Both problems are NP-hard (ORourke and Supowit 1983). This problem is also called the “art gallery problem” (Aggarwal 1984). There are two major approaches to compute approximate solutions. First, the environment is modeled as a polygon, the range of the sensors and field of view are assumed to be infinite. Approximate algorithms solve the problem based on the set-covering concept. Second, real-time sensor data from the robots are collected for learning. Based on the learned coverage function, the robots move to the locations that maximize coverage. This procedure allows to find an approximate solution.

A polygonal environment is divided into several subsets. Choosing the least covered set provides a simple way to find a solution. Such greedy algorithms generate solutions with $O(\log N)$-approximation, where N is the number of sets. ϵ-net finder provides an $O(\log\log OPT)$-approximate algorithm for guarding a simple polygon with guards on the perimeter (King and Kirkpatrick 2011), where OPT is the minimal number of guards. To further consider the sensor can observe all orientations of the object, $\Omega(\sqrt{n})$ is necessary to cover a simple polygon with n vertices (Tokekar and Isler 2014). However, those theoretical bounds are based on highly simplified sensor and/or environmental models.

Cognitive-based adaptive optimization (CAO) is proposed for the optimization problems where the objective function is unknown but measurements are available (Kosmatopoulos 2009; Renzaglia et al. 2011). It is applied to maximize the coverage using a fixed number of robots equipped with sensors in 3D environments (Renzaglia et al. 2012). Given no prior information about a specific objective function and environments, the CAO framework has two advantages. First, the robots learn the objective function from real-time sensing data. Second, the CAO algorithm still works, even if the environment is non-convex. However, this approach cannot give a theoretical guarantee of coverage.

2.2 Coverage control

The technological advances in sensors and networking have transformed robots into mobile sensing networks. They can sample the environment more dynamically in space and time than static sensors. To do this successfully, robots must cover the environment and distribute control via communications. Lloyd algorithms achieve these objectives by finding solutions of Voronoi diagrams based on given constraints (Lloyd 1982). In Cortes et al. (2004), Lloyd descent algorithms can coordinate a mobile sensor network. In Pimenta et al. (2009), the proposed approach allows robot networks to simultaneously cover and track moving targets. In Schwager et al. (2009), UAVs with downward facing cameras are used to monitor an environment through a a distributed control algorithm. These applications and algorithms demonstrate the vast potential of mobile sensing networks.

2.3 Probabilistic search

Bayesian search is divided into two components, perception and decision-making. Perception is to compute the probability distribution of the target. Recursive Bayesian estimation techniques enable robot(s) to estimate the target using real-time motion and measurement data (Bourgault et al. 2003). To further consider multiscale search, probabilistic quadtrees are proposed for a UAV flying at different elevations (Chung and Carpin 2011). Decision-making is to compute the optimal actions toward the target. However, finding the optimal decisions in a probabilistic environment is NP-hard.

The decision-making problem was reformulated as Partially Observable Markov Decision Processes (POMDPs) (Eagle 1984; Kadane and Simon 1977), which is NP-hard. To allow larger horizons, maximizing the cumulative probability of detection (PD) was proposed (Stone 1975). There are two major assumptions in PD model. First, there is no false detections and no detection of the target along the search path. Second, the sensing coverage at each position is nonoverlapping and independent of one another. Based on these assumptions, the probability of detection is easily propagated for larger horizons through Bayes filter. Even if the propagation model is simplified, maximizing the cumulative PD is an NP-complete problem (Trummel and Weisinger 1986).

Maximizing cumulative PD within finite horizons provides a way to find suboptimal actions. There are two major methods to find suboptimal paths. First, the maximization of the PD problem is formulated as a mixed-integer linear programming (MILP). The robots are able to find suboptimal search paths (Lo et al. 2012). Second, branch and bound (BNB) is used to reduce the computation. The principle of branch and bound is to generate the possible branches and prune the branches if the cost values exceed some prespecified costs. This concept enables a robot(s) to search for a moving target in indoor environments based on the PD model (Lau et al. 2008, 2006). Therefore, by choosing a suitable bound or horizon, suboptimal solutions of search paths can be obtained with an affordable computation.

2.4 Minimum-time trajectory planning

Minimum-time trajectory planning is NP-hard problem, which means that computational complexity grows exponentially with the order of the searcher's dynamics and the size of the map. Hence, the key challenge to solving this problem is finding the trade-off between reducing the computational load and increasing performance. There are two major approaches to compute approximate paths for a robot(s). The rapidly-exploring random tree (RRT) (LaValle and Kuffner 1999) works by randomly generating nodes from the free space once the connection between two nodes is obstacles-free. The robot's approximate path from start to goal is found by graph search.

Another approach, which was introduced in aerospace for UAV guidance, is receding horizon (RH) trajectory optimization with a cost-to-go (CTG) function (Bellingham et al. 2002). The CTG function is an approximation of the cost to reach a prespecified goal in the global environment. The Receding horizon solves a constraint optimal control problem over a finite horizon, subject to the terminal cost obtained by the CTG evaluated for the state value attained at the end of that horizon (Mettler et al. 2010). The state that minimizes the composite cost, i.e. the cost to reach the intermediate state (cost-to-come, CTC) and the cost from that state to the global goal (CTG). That states define the active waypoint (AWP). The AWP is decided based on the CTG function and the horizons of receding horizon control. For example, when the number of horizons is larger, the robot can choose farer AWP. With this approach, the original infinite-horizon optimization problem is divided into a sequence of smaller trajectory planning problems. More details on guidance techniques can be found in Goerzen et al. (2010) and Bellingham et al. (2002).

2.5 Informative path planning (IPP)

In the past 10 years, the robotics community has adopted the concept of informative path planing (IPP) (Singh et al. 2007) and submodular functions (Nemhauser et al. 1978; Krause et al. 2008; Singh et al. 2009). Instead of finding minimum-time trajectory, the goal of IPP is to find an optimal path which maximizes the amount of sensed information (Binney and Sukhatme 2012). IPP is a novel approach for active perception. It includes three relevant questions. First, where to place the sensors/robots so that most information can be collected? Second, how to construct a path satisfying certain objective functions with limited bounds? Third, what theoretical guarantee can be given about the performance of the approximated solutions?

The first problem is related to "adaptive sampling and feature selection" (Binney and Sukhatme 2012) and is based on machine learning techniques, such as Gaussian Processes (GP). This problem is to study where to place the sensor to maximize/minimize a quantity (e.g., PD, information gain and mutual information). The second problem is related to "path planning and probabilistic search" (Binney and Sukhatme 2012). Path planning is to find a path to a known goal. Probabilistic search is to find the target given the target distribution and motion model. The third problem is related to submodularity. If the objective function is submodular, greedy approaches give $(1 - 1/e)$ OPT guarantee (Nemhauser et al. 1978). Moreover, no polynomial time algorithms are strictly better than greedy approaches unless $P = NP$ (Feige 1998). Example applications include finding paths that enable robots to collect maximal mutual information (Singh et al. 2007), maximal variance reduction of Gaussian processes (Binney and Sukhatme 2012) and minimal distortion from wireless stations (Hollinger and Sukhatme 2013; Hollinger et al. 2013).

3 Problem reformulation

The successful applications of IPP and submodularity inspire a reformulation of the search problems. Section 3.1 describes the three objective functions associated with robotic search problems. Section 3.2, then introduces the near-optimal approach for the coverage problem. Section 3.3 proves that the probabilistic search objective function is submodular and that the coverage problem is a special case of probabilistic search. And, finally, Sect. 3.4 shows that if the objective function is maximizing cumulative PD with motion costs, greedy approaches generate near-optimal solutions.

3.1 Three objective functions of search problem

Definition 1 (*Optimal search with three objectives*) The ground set of subgoals is $S = \{1, 2, ..., N\}$ in a grid map. The objective is to choose K subgoals (S_g) which satisfy:

$$\text{(Objective 1: maximum coverage): } \max_{S_g \subseteq S} f_C(S_g)$$
$$\text{(Objective 2: maximum probability): } \max_{S_g \subseteq S} f_P(S_g)$$
$$\text{(Objective 3: minimum motion cost): } \min_{S_g \subseteq S} T(S_g)$$

where f_C is the coverage function, f_P is the PD function and T is the CTG function.

To illustrate the three functions and subgoals, assume there are four subgoals $S = \{1, 2, 3, 4\}$ (see Fig. 3a). $S_{g_1} = \{1, 2\}$ represents the two chosen subgoals. Figure 3b shows two subgoals cover 30% so $f_C(S_{g_1}) = 30\%$. Figure 3c shows two subgoals cover 20% probability so $f_P(S_{g_1}) = 20\%$. Figure 3d shows the motion cost of two subgoals is 8 s so $T(S_{g_1}) = 8$. Definition 2 describes how to compute f_C, Definition 6 describe hows to compute f_P and Definition 8 describes how to compute T.

The three objectives are in general coupled. For example, choosing 3 subgoals with the least motion cost could have small coverage. Even if Objective 1 and 2 are submodular (Tseng and Mettler 2015), it is difficult to give a

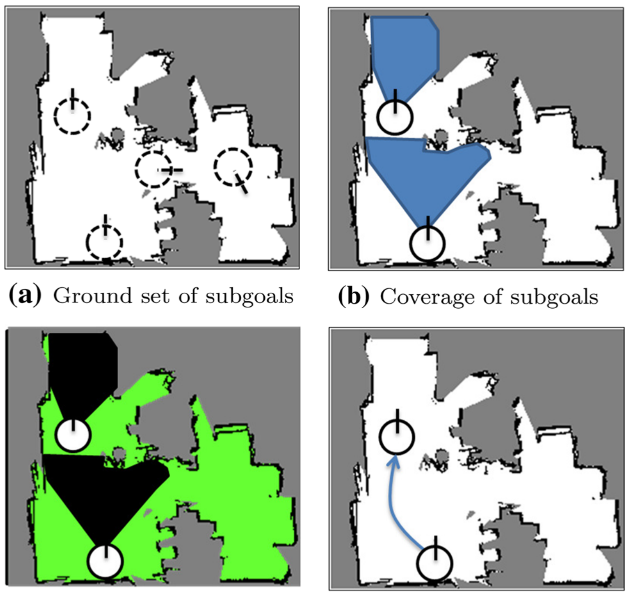
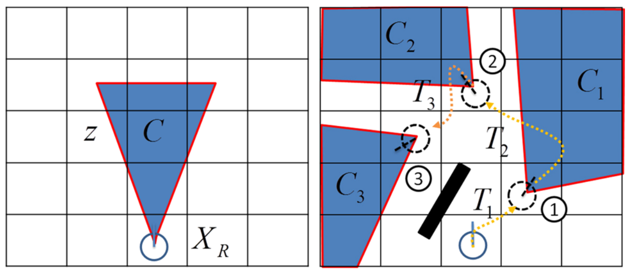

(a) Ground set of subgoals (b) Coverage of subgoals
(c) PD of subgoals (d) Motion cost of subgoals

Fig. 3 Illustration of three functions. **a** The *dash circles* represent the ground set of 4 subgoals. **b** The *black circles* represent chosen the two subgoals and *blue areas* are the corresponding covered areas of subgoals. **c** The *green areas* represents high probability and *black areas* represent low probability. **d** The *blue arrow* is the lowest cost path toward the subgoal



(a) Coverage (b) Coverage with motion

Fig. 4 Illustration of the coverage problem. **a** The *blue circle* represents the current robot position and the *blue area* is the corresponding covered area. **b** The *numbers* represent the subgoal index, *black circles* are subgoal positions, *orange arrows* are paths, *blue areas* are corresponding covered area of each subgoal and *black areas* are obstacles

performance guarantee for the overall search performance, therefore, to be able to provide a performance guarantee, a reformulation is necessary.

3.2 Coverage problem

Definition 2 (*Coverage*) As Fig. 4a shows, the robot locates at X_R and gets the sensing measurements $z = \{r_i, \theta_i\}$, where $i = 1, ..., N_z$. r is the range, θ is the angle and N_z is the number of sensor scans. Assume there are N_g unoccupied cells in a grid map. The robot's sensor covers N_c cells. The coverage $C(X_R)$ is defined as N_c/N_g.

Both ray-tracing and grid mapping techniques can compute the coverage efficiently (Charrow et al. 2015; Renzaglia et al. 2010; Konolige 1997). According to X_R and z, the covered area is computed. For example, there are 25 cells and 8 cells are covered by the robot (see Fig. 4a). Hence, $C(X_R)$ is 32%.

Definition 3 (*Maximum coverage problem*) As Fig. 4b shows, the ground set of subgoals is $S = \{1, 2, ..., N\}$. The objective is to choose K subgoals (S_g) such that $\max_{S_g \subseteq S} f_C(S_g)$.

As Fig. 4b shows, there are 25 cells and the heading resolution of the robot is 45°. So, the size of the ground set is 200 (25×8). If the robot tries to find 3 subgoals such that the coverage of subgoals is maximal, it needs to compute 200^3 combinatorial solutions and then choose the subgoal set with maximal coverage. Then, the robot visits the chosen subgoals to cover the environment. The computational complexity for this problem is $O(N^k)$. In fact, this is a NP-hard problem. Therefore, approximate solutions are needed.

Since the coverage function is submodular (see Lemmas 1 and 2), greedy approaches give $(1 - 1/e)$ OPT guarantee (Nemhauser et al. 1978). Moreover, no polynomial time algorithms are strictly better than greedy approaches unless $P = NP$ (Feige 1998). Hence, a greedy algorithm generates $(1 - 1/e)OPT$ solution for objective 1. As Fig. 4b shows, the robot chooses the first subgoal with maximal coverage from 200 candidates. Then, the robot repeats the same greedy approach for the next two subgoals. Finally, the robot only needs to compute 200×3 solutions.

Lemma 1 ($(1 - 1/e)$-Approximation (Nemhauser et al. 1978)) *Let F be a monotone submodular set function over a finite set S with $F(\emptyset) = 0$. Let A_G be the set of the first k elements chosen by the greedy algorithm and let $OPT = \max_{A \subset S, |A|=k} F(A)$. The lower bound of the greedy algorithm is $F(A_G) \geq (1 - 1/e)OPT$.*

Lemma 2 (Near-optimal guarantee (Feige 1998)) *The result of Lemma 1 is tight and there is no polynomial time algorithm can do strictly better than greedy algorithm if $P \neq NP$.*

3.3 Probabilistic search

Computing optimal decisions for probabilistic search is a partially observable Markov decision processes (POMDPs) problem, which is NP-hard. Hence, a probability of detection (PD) model is proposed that can be propagated for larger horizons.

Definition 4 (*Probability of detection (PD)*) As Fig. 5a shows, the robot locates at X_R and gets the measurements z.

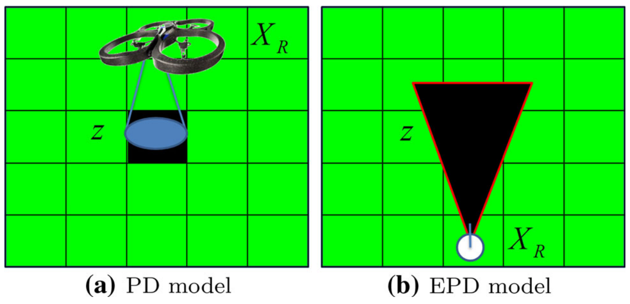


Fig. 5 Illustration of PD and EPD model. **a** The *blue areas* represent the covered area and *green areas* represent high probability. **b** The *blue circle* represents the robot position and *green areas* represent high probability

The assumptions about the robot's motion and sensing are as follows:

(i) there is no target detection along the path,
(ii) the robot only senses in its local cell and there is no sensing coverage overlap between different subgoal positions,
(iii) the robot only moves to the neighbor cells.

Based on these assumptions, the probability of detection (PD) is defined as $P(X_R) \cdot g$, where $P(i)$ is the probability of the i-th cell, g is glimpse function[3] and $0 \leq g \leq 1$.

The PD was originally proposed for airplanes and ships operating in large-scale environments (Stone 1975; McCue 1990). Hence, the 2nd and 3rd assumptions in Definition 4 are reasonable. However, they do not consider two critical factors. First, the sensing areas at different positions could overlap, and second, the robot can move to any cell in this environment. To make the PD model more realistic, an extended PD (EPD) model is proposed as follows:

Definition 5 (*Extended PD (EPD) model*) As Fig. 5b shows, the robot is located at X_R and gets the measurements z. The assumptions about the robot's motion and sensing is that there is no detection of the target along the path. Based on this assumption, the probability of detection (PD) is defined as $P(X_R) \cdot g$, where $P(X_R)$ is covered probability in X_R position.

According to Definition 5, EPD model deletes assumption (ii) and (iii) in PD model. Hence, EPD model supports sensing overlap and do not consider path constraints.

[3] The value of glimpse function can be decided from the conditional probability table of the given detector. $g = \frac{q-pq}{1-pq}$, where p denotes the probability of detected area. q denotes the probability of that the robot detects the target if the target is at the detected area. The derivation of glimpse function can be found in the appendix of Tseng and Mettler (2015).

Definition 6 (*Maximum cumulative PD*) As Fig. 4b shows, the ground set of subgoals is $S = \{1, 2, ..., N\}$. The cumulative probability of detection is defined as: $f_P(S_g) = \sum_{i=1}^{T} P(S_{g,i}) \cdot g$, where f_P is the cumulative PD along the path and $P(S_{g,i})$ is the probability of covered cells at the ith subgoal. The objective is to choose K subgoals from S such that the cumulative probability of detection (f_P) is maximal.

Theorem 1 *(Submodularity of probabilistic search (Tseng and Mettler 2015)) Probabilistic search is submodular if there are no detections along the search path.*

As Lemma 1 and 2 show, the greedy approach generates near-optimal solutions for submodular function. In Tseng and Mettler (2015), the authors already proved Theorem 1. However, they did not find the relationship between the coverage problem and probabilistic search, so the two problems are solved individually. The relationship between the two problems is as follows:

Theorem 2 *(Special case of probabilistic search) Maximum coverage is a special case of maximum cumulative PD under EPD model if the following conditions hold:*

(i) the target probability distribution is uniform,
(ii) $g = 1$.

Proof The cumulative PD is $f_P(S_g) = \sum_{i=1}^{T} P(S_{g,i}) \cdot g$. The coverage is $f_C(S_g) = \sum_{i=1}^{T} C(S_{g,i})$. There are two major difference between PD and coverage. First, the weight of each cell in the coverage problem is the same. Second, the g in PD model is between 0 and 1. Thus, if the target distribution is uniform and g equals 1, the objective 1 and 2 are the same.

Since coverage is a special case of probabilistic search under EPD model, achieving objective 2 also satisfies objective 1. According to Lemmas 1, 2 and Theorem 1, greedy approaches give near-optimal solutions for objective 2. The three objective functions are reduced to two objective functions– maximum probability and minimum motion cost. The incorporation of objective 2 and 3 are described next.

3.4 Probabilistic search with motion costs

Considering only objective 2 would be insufficient for robotic applications, since the robot's motion cost to visit those subgoals sequentially could be excessive. As Fig. 4b shows, the total motion time is $T_1 + T_2 + T_3$. Hence, considering both probability and motion cost is necessary to achieve an efficient search. The unified objective function is formulated as follows:

$$\max_{S_g \subseteq S} f_P(S_g) \qquad s.t. \quad T(S_g) \le T_{th}, \tag{1}$$

where T_{th} is an assigned motion cost threshold. This problem can be seen as a submodular maximization problem with different item costs, which is also NP-hard (Khuller et al. 1999). Fortunately, a near-optimal solution of Eq. 1 can be found taking advantage of submodularity.

Lemma 3 *(Submodularity with non-uniform item costs (Khuller et al. 1999)) Given a submodular function $F : 2^N \to \mathbb{R}$, a finite set S = {1,2,...,N} and the corresponding item cost $\tau(s) \in \mathbb{R}$, the greedy algorithm is $S_{G,i+1} = S_{G,i} \cup \{\arg\max_{s \in S \setminus S_{G,i}} \frac{F(S_{G,i} \cup \{s\}) - F(S_{G,i})}{\tau(s)}\}$. The greedy algorithms using a partial enumeration technique give $(1 - 1/e)OPT$ guarantee.*

Lemma 3 is similar to Lemma 1 but each item has a different cost. Even if each item cost is different, the submodularity holds and greedy approaches generate near-optimal solutions (Khuller et al. 1999).

However, Lemma 3 cannot apply to motion costs because the $\tau(s)$ depends on the given set S_A or S_B. As Fig. 4b shows, assume $S_A = \{1\}$, $S_B = \{1, 2\}$ and $s = \{3\}$. $\tau(s|A)$ denotes the motion cost adding subgoal s from the subgoal set S_A. $T_{i,j}$ denotes the motion cost from i-th subgoal to j-th subgoal. In this example, $\tau(s|A)$ is the motion cost when the robot at subgoal 1 moves to subgoal 3. $\tau(s|A)$ is $T_{1,3}$ and $\tau(s|B)$ is $T_{2,3}$. Hence, $\tau(s|A) \neq \tau(s|B)$. To prove the submodularity with motion costs, a probabilistic bound is introduced as follows:

Theorem 3 *(Optimality of maximizing PD with motion costs (Tseng and Mettler 2015)) Given a set of subgoals S_G chosen from a ground set of subgoals S by a greedy algorithm, the following equation holds: $f_P(S_G) \ge (1-1/e) f_P(S_{opt})$ with high probability.*

where S_{opt} represents the optimal subgoals chosen from S.

Theorem 3 shows that the greedy approach for maximizing PD with motion costs (Eq. 1) generates the near-optimal solution with high probability. The approximation factor is $(1 - 1/e^{\frac{T(S_g)}{T_{th}}})$. If $T(S_g) = T_{th}$, the approximation factor is $1 - 1/e$. Since the motion cost is always over than 1 s and the coverage/probability is always less than 1, the motion cost dominates parts of decisions. In Tseng and Mettler (2015), the proof shows that the $T(S_g)$ is very close to T_{th} with high probability. Hence, the performance of the selected subgoals is close to $1 - 1/e$ with high probability.

This search problem consists of two levels: motion control and subgoal determination (see Fig. 2c). If the robot knows f_P and T through precomputation, it can find near-optimal solutions with high probability. f_P can be computed online through ray-tracing techniques if the number of subgoals (N) is smaller than 100 and a map is provided. T can be computed online through the Dijkstra algorithm if the number of subgoals (N) is smaller than 100 and the grid size is smaller than 10,000. However, if these conditions are not satisfied, the robot must learn the two functions as the agent interacts with the environment. The following section explains motivations for learning two functions.

3.5 Motivations for learning PD and CTG functions

Motivations for learning the PD function are as follows: First, when a map is not provided or does not include accurate data (e.g., some pieces of furniture are moved), the robot must learn to search in unknown environments. In such scenarios, the robot cannot use ray-tracing techniques to compute f_P directly and it needs to explore the coverage values of each subgoal. Then, the robot can learn the f_P based on what it observed. Furthermore, the experiments show that simultaneous exploration and coverage approach outperforms exploration-only approaches by utilizing the submodularity of coverage problems (Heng et al. 2015). Second, exploring the invariant property of search problems is also important. A robot should be able to reuse information gained from previously searching an environment when later searching a new environment. The robot could utilize reused information to reduce the search computation in various environments. Third, the learning approach could be more computationally efficient than ray-tracing techniques. Although ray-tracing techniques can compute coverage efficiently in 2D environments, computing PD requires the application of Bayes rule to each grid cell. Hence, batching some data for learning is an efficient way to approximate PD functions. In Heng et al. (2015), the authors also indicated that ray-tracing techniques in 3D environments are infeasible for real-time motion planning of UAVs.

Fourth, data storage and communication are important for multi-agent search. For example, if two robots search cooperatively, they can share their submodular functions. The number of submodular values is 2^N, which is infeasible for communication. Therefore, compression techniques must be used. The compressed sensing techniques already execute sensing and compression simultaneously (Hayashi et al. 2013). Moreover, they can simultaneously reconstruct the submodular function via distributed compressed sensing (Chen et al. 2011). Finally, submodular functions can be applied to many domains. For example, combinatorial auctions for coalition formation (Vig and Adams 2007), feature selection problems in graphical model (Krause and Guestrin 2005) and clustering problems (Narasimhan and Bilmes 2007) are popular in various robotic applications. The objective functions of these problems are submodular. Hence, learning submodular functions can also apply to these problems.

Receding horizon (RH) trajectory optimization with a CTG function is proposed to solve guidance problems for a UAV (Mettler et al. 2010). In the precomputation stage, the CTG function is computed based on Dijkstra algorithm and motion primitive automaton (MPA). In the motion planning stage, the robot computes velocity commands using nonlinear programming to achieve minimum-time trajectory planning. However, there is only one goal in guidance problems while there are N subgoals in search problems. The amount of precomputation is therefore much heavier for search tasks. For example, if the map is $10\,\text{m} \times 10\,\text{m}$ and CTG resolution is $10\,\text{cm} \times 10\,\text{cm} \times 45^\circ$, computing a CTG takes about 1 min. If $N = 100$, computing all CTG functions will take over 1 h. Hence, the motivation of learning the CTG function is as follows: Learning from motion data is a way to efficiently approximate CTG functions. If a small set of simple features can approximate CTG functions, the robot can learn CTG functions using reinforcement learning approaches. This can save hours of precomputation.

If the robot learns both the CTG function at the motion control level and the PD function at the subgoal level, then the robot will greedily chooses the subgoals according to the learned functions until arriving at the Kth subgoal or finding the target. Hence, learning the two functions is the key to solving efficient search. The definition of the two learning problems and the collection of the training data are introduced in the two following sections.

4 Learning PD function

Learning a submodular function is a challenging problem since there are 2^N values based on different discrete sets. In Balcan and Harvey (2011), approximating a submodular function over $\sqrt{N}/\log(N)$ bound can be simplified as a classification problem. The authors proposed probably mostly approximately correct (PMAC) learning. However, approximating a submodular function within $\sqrt{N}/\log(N)$ bound requires an exponential number of samples. In Stobbe and Krause (2012), the author proposed Fourier sparse set functions (FSS) to learn submodular functions using compressed sensing techniques. This approach requires fewer samples. However, it needs more computation. Section 4.1 introduces the FSS concept. Since FSS only works for a small number of samples ($N < 100$) due to the time complexity $O(N^4)$, exploring the sparisty in the Fourier domain could reduce the unnecessary computation. Section 4.2 explores the sparsity in the Fourier domain that can be exploited to reduce the number of bases. Section 4.2.2 introduces the proposed Spatial Fourier Sparse Set (SFSS) for learning PD function. Before introducing how to learn a submodular function, this section details the learning PD function problem and how to collect samples.

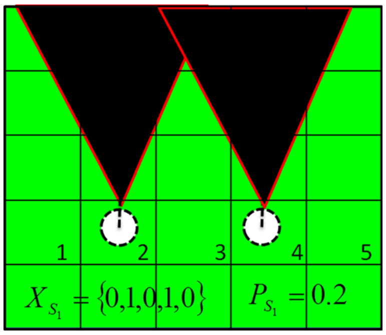


Fig. 6 Illustration of the PD learning problem. The *numbers* represent the feasible subgoals, *black circles* are subgoal positions, *green areas* represent hight probability and *black areas* represent low probability. There are five feasible subgoals ($n = 5$). If the subgoals are at the 2nd and 4th cells (X_{S_1}={0,1,0,1,0}), the corresponding PD (P_{S_1}) is 0.2

Definition 7 (*Learning PD functions*) The ground set of subgoals is $S = \{1, ..., N\}$. Given subgoal-PD data $X_s = \{X(S_i), P_{S_i}\}$, where $X(S_i) \in \{0, 1\}^N$ represents selected subgoals, P_{S_i} is the corresponding PD and $1 \leq i \leq m$, where m is the sample size. The objective is to find an approximate PD function $f_P : \{0, 1\}^N \to \mathbb{R}_+$.

Figure 6 illustrates the number of subgoal ground sets is five. The measurements z_2 and z_4 at 2nd and 4th cells are assumed to be known. Given this information the subgoal set $S_1 = \{2, 4\}$ and corresponding PD (P_{S_1}) are computed. The subgoal-PD data is $X_s = \{X(S_1), P_{S_1}\} = \{\{0, 1, 0, 1, 0\}, \{0.2\}\}$. As the robot collects enough subgoal-PD data, f_P can be approximated.

The approach to collect the subgoal-PD data is the following: First, once the robot visits the ith-cell, it saves the z_i into the database. Second, the robot randomly chooses $\{n_1, n_2, ..., n_{k_s}\}$ from the database, where $1 \leq k_s \leq K_s$ and K_s denotes the maximal number of sampled subgoals. According to $S_i = \{n_1, .., n_{k_s}\}$ and $z_{n_{1:k_s}}$, the robot computes the corresponding probability P_{S_i}. Third, the robot saves the subgoal-PD data (X_{S_i}, P_{S_i}) into X_s. After repeating the three steps, the robot has a batch of subgoal-PD data. The following subsection describes how to utilize subgoal-PD data to learn a PD function.

4.1 Fourier sparse set (FSS)

As described in Stobbe and Krause (2012), the compressed sensing technique can be used to learn submodular functions using fewer samples. As Fig. 7a shows, the robot first acquires a signal $X_{(n,1)}$ via a sensing matrix $\Phi_{(m,n)}$ and collect $f_M := f_{(m,1)}$ for learning, where $m << n$ and $X_{(n,1)}$ is the submodular function. The robot has to recover the signal X (see Fig. 7b). Notice that this is an ill conditioned linear inverse problem. However, if the signal is sparse in certain domains, the robot can recover X using sparse regression. As

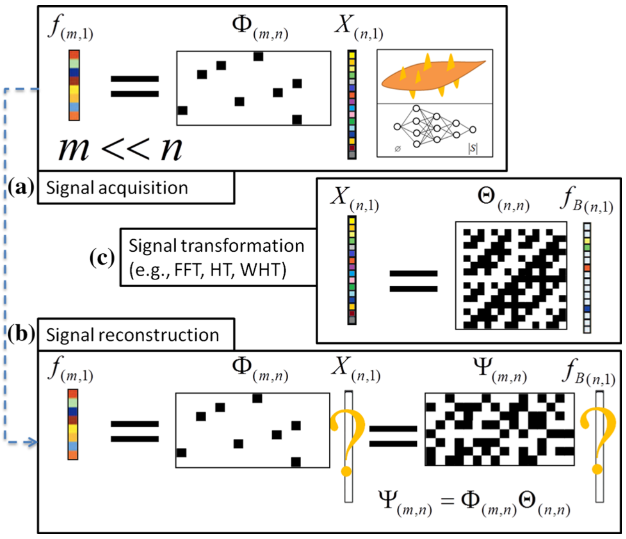


Fig. 7 Illustration of the compressed sensing concept. **a** $f_{(m,1)}$ is collected by the robot after taking measurements from a signal $X_{(n,1)}$. The *color cells* represent real values and *black/white* cells represent binary values (0 and 1 in Φ while 1 and -1 in Θ.) **b** The robot has $f_{(m,1)}$ and tries to recover $X_{(n,1)}$. **c** The signal X is sparse in the Fourier domain. In this example, m is 8, n is 16 and k is 4. Given $\Phi_{m,n}$ and $f_{(m,1)}$, it's impossible to recover $X_{(n,1)}$ ($m < n$). But, given $\Psi_{(m,n)}$ and $f_{(m,1)}$, $f_B(n, 1)$ can be recovered ($k < m$)

Fig. 7c shows, X is the inner product of the transform matrix $\Theta_{(n,n)}$ and coefficient $f_{B(n,1)}$. $f_{B(n,1)}$ has only k nonzero values (so called k-sparsity). Since Θ and Φ are known, the reconstruction matrix Ψ can be computed. Although directly recovering X is impossible, the robot can recover $f_{B(n,1)}$ if $k < m$, and then recover X. The signal recovery formulation is given as:

$$\hat{f}_B = \arg\min_{f_B} \frac{1}{2}||f_M - \Psi f_B||^2 + \lambda||f_B||_1 \tag{2}$$

where f_B is the submodular function in the Fourier doman, f_M is a measurement vector of the submodular function, Ψ is a reconstruction matrix, $\Psi = \Phi\Theta$, Φ is a sensing matrix and Θ is a transform matrix.

The key assumption is that the signal is sparse in certain domains. Hence, it is necessary to find a Fourier domain where the coefficients of submodular functions are sparse. In Stobbe and Krause (2012), the submodular function is represented as a pseudo-Boolean function. Since basis vectors are composed of binary numbers (e.g., +1 and −1), this transform is a special kind of Fourier transform called Hadamard transform.

The problem of learning submodular functions can be formulated as follows: Given the measurement sets A, the corresponding submodular values f_M and basis sets B, the goal is to recover the coefficients $\hat{f}_B$ in the Fourier domain. Once $\hat{f}_B$ is known, the submodular function can be reconstructed from any input set S. The steps of FSS learning are as follows:

(1) Compute reconstruction matrix:

$$\Psi[i,j] = \psi_{B_j}(A_i) = (-1)^{|A_i \cap B_j|}$$

(2) Recover Fourier coefficients:

$$\hat{f}_B = \arg\min_{f_B} \frac{1}{2}||f_M - \Psi f_B||^2 + \lambda||f_B||_1$$

(3) Recover submodular functions:

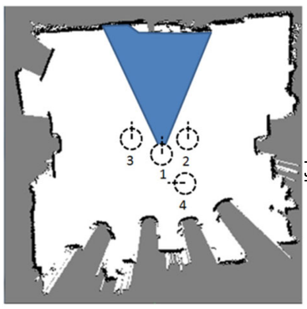
$$f(S) = \sum_{B \in 2^n} \hat{f}_B \psi_B(S)$$

Time and sample complexity are important for learning. The major computation takes place at step 2, which is solved by an accelerated gradient descent approach (Beck and Teboulle 2009). Thus, the time complexity of FSS is $O(Ib^2)$, where I is the iteration number and b is the number of bases. According to the Restricted Isometry Property (RIP), the sample complexity is $m = O(klog(b))$, where k is the number of nonzero Fourier coefficients (Candes et al. 2006).

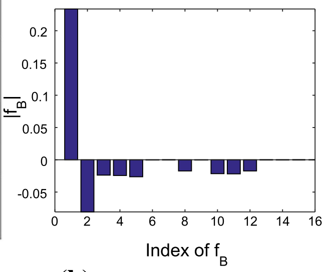
The size of b is $\sum_{l=0}^{d} \binom{n}{l}$, which is exponentially increasing in presence of higher-order coefficients in the pseudo-Boolean function. In Stobbe and Krause (2012), the authors found that second-order pseudo-Boolean functions can approximate submodular functions. Thus, the size of b is $\sum_{l=0}^{2} \binom{n}{l}$. However, since the number of second-order bases is $b = O(N^2)$ and the time complexity is $O(Ib^2)$, the time complexity is $O(IN^4)$, which is infeasible for online learning. Hence, this approach only works for $N < 100$. For example, if $N = 100$, $b = 5101$, it cannot be applied to online learning. In Stobbe and Krause (2012), the authors do not address the relationship between sparsity in spatial and Fourier domains, which could be useful to reduce the number of bases. The relationship is described in the following section.

4.2 Sparsity in the Fourier domain

To explore the sparsity in the Fourier domains, the coverage of 4 subgoals is transferred to the Fourier domain for the observations (see Fig. 8). Notice that the covered area for the 1st, 2nd and 3rd subgoals is highly overlapping. But, there is no overlap between the covered area of 4th subgoals and the other 3 subgoals. The total number of possible subgoal sets is 16. Next, the coverage (f_M) for the 16 subgoal sets is computed. The associated basis table is shown in Table 1. f_B can be computed by $\Psi^{-1} f_M$ and as shown in Fig. 8b, the 6th, 7th, 9th, 13rd, 14th, 15th and 16th coefficients are zero. The common characteristics of these bases is that The B_4 values of these bases are 1 (see Table 1). In other words, 4th subgoal is one of these bases' choosing subgoals.



(a) Subgoals



(b) Fourier coefficients

Fig. 8 Illustration of sparsity in the Fourier domain. **a** The *black dash circles* represent the subgoal positions and the *blue area* represents the covered area by subgoal #1. **b** The Fourier coefficients shows that the 6th, 7th, 9th, 13rd, 14th, 15th and 16th coefficients are close to zero

Table 1 Basis Table. For example, the 7th basis represents choosing the 2nd and 4th subgoals. The table also indicates the order of the bases. $i-th$ order means choosing i subgoals

	Index	B_1	B_2	B_3	B_4
0th order	1	0	0	0	0
1st order	2	0	0	0	1
	3	0	0	1	0
	4	0	1	0	0
	5	1	0	0	0
2nd order	6	0	0	1	1
	7	0	1	0	1
	8	0	1	1	0
	9	1	0	0	1
	10	1	0	1	0
	11	1	1	0	0
3rd order	12	1	1	1	0
	13	1	1	0	1
	14	1	0	1	1
	15	0	1	1	1
4th order	16	1	1	1	1

Based on this example, the relationship between sparsity in the spatial and Fourier domains is as follows: First, if the subgoals result in no overlap, the Fourier coefficients of the corresponding bases is zero. Second, the coefficient values of higher-order is close to zero with high probability. According to this finding, if these 7 basis vectors are removed from B, f_B still can predict f_M accurately. In other words, unnecessary basis vectors can be removed before processing with sparse regression. Based on the sparsity in the Fourier domains, the invariant property and interaction patterns in the Fourier domain are introduced in the following subsections.

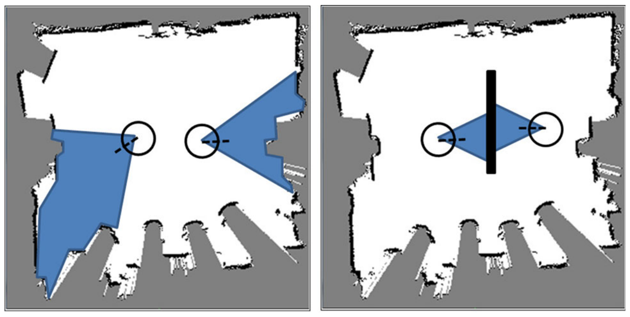


Fig. 9 Illustration of nonoverlapping coverage. **a** The overlapping coverage of two subgoals is zero due to the subgoal configuration. **b** The overlapping coverage of two subgoals is zero due to the obstacle

4.2.1 Invariant property in the Fourier domain

According to the sparsity in the Fourier domain, skipping two nonoverlapping subgoals can reduce the number of the second-order bases. However, there are two cases of nonoverlapping subgoals. They correspond to the scenarios with and without obstacles between subgoals. These two cases are illustrated in Fig. 9. The 2nd case is difficult to compute since it depends on the subgoal and map configuration. The 1st case is easy to compute since it only involves two subgoal positions and is environment-independent. Thus, before learning, the bases for all subgoals that fall in the 1st case can be discarded. After learning, the basis coefficients of the 2nd case are close to zero.

To illustrate the invariant property, Figure 10a, b demonstrate that two different maps share the same four subgoals. Their 16 coverage values (X^{M_1} and X^{M_2}) are vary in spatial domain. After the Fourier transformation, their 1st, 2nd, 3rd, 4th, 5th and 8th values in the Fourier domain are different, while the other values are zero. In other words, they have the same sparsity even in different environments.

Since only one area will be covered by two subgoals, the coefficient of 2nd order term (the 8th value) is non-zero. If an obstacle occludes the visibility of a subgoal (see Fig. 10c), there is no overlap between two subgoals, therefore the 8th value in the Fourier domain becomes zero. In other words, the 1st case is environment-independent while the 2nd case is dependent on the environment.

4.2.2 Interaction patterns in the Fourier domain

Search is the interaction among the searcher(s), target(s) and environment(s). Their interaction patterns affect the learnability of submodular functions. According to compressed sensing theory, the robot needs $O(k\log b)$ samples to recover the submodular functions. The interaction patterns in learning are as follows:

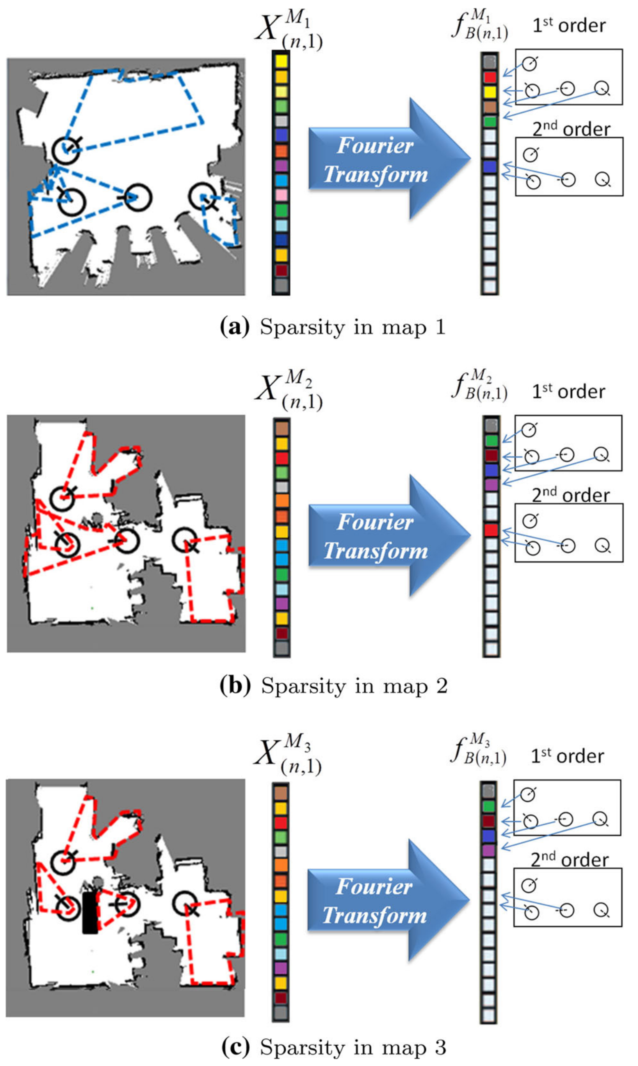


Fig. 10 Illustration of sparsity in different maps. The *white color* represents zero value and the other *colors* represent different values in X and f_B. There are the same 4 subgoals in two maps. After Fourier transform, the zero values appear in the same positions

Searcher: The subgoal patterns will affect the number of b. The less overlap between subgoals' coverage generates more sparse coefficients of bases and more basis vectors can be removed. Environment: The number of k is decided by the subgoal and map configuration. If there is overlap between two subgoals' coverage and there is an obstacle blocking the overlap areas (see Fig. 10c), this coefficient of basis is zero. Target: The target's motion and prior distribution affect the coefficient values in f_B. If the target distribution is 99% in a certain area, this distribution will generate more sparsity in f_B. According to these findings in the Fourier domain, Spatial Fourier Sparse Set (SFSS) is introduced next.

4.3 Spatial Fourier sparse set (SFSS)

The major difference between FSS and SFSS is the reduction of the basis number achieved by exploiting the sparsity relationship between the spatial and Fourier domains. The steps to generate a basis matrix (B) are as follows:

(1) 0 order basis: assign a zero vector.
(2) 1st order bases: pick $\binom{N}{1}$ 1st order bases.
(3) 2nd order bases: pick $\binom{N}{2}$ 2nd order bases.
(4) Reduce bases: compute nonoverlapping subgoals and remove them from 2nd order bases.

Steps 1 and 2 take $\binom{N}{0}+\binom{N}{1}$ terms. Step 3 and 4 takes $\beta\binom{N}{2}$ terms, where $0 \leq \beta \leq 1$, which depends on the subgoal patterns. Hence, the size of b is $O(\beta N^2)$. This approach can be extended to nth order basis. The advantages of the SFSS are as follows:

– Reduced basis number: The FSS needs $\sum_{l=0}^{2}\binom{N}{l}$ bases while SFSS only needs $\sum_{l=0}^{1}\binom{N}{l}+\beta\binom{N}{2}$.
– Reduced computation: The computational complexity of f_B is $O(Ib^2)$, where I is the number of iterations and b is the number of bases. Thus, the complexity of FSS and SFSS are $O(Ib^2)$ and $O(Ib^2\beta^2)$, where $\beta \leq 1$.
– Reduced number of samples: The sample complexity of FSS and SFSS are $O(klog(b))$ and $O(klog(\beta b))$.
– Computational efficiency: Since the elements in Ψ consist of $+1/-1$ and $f(S) = \Psi f_B$, computing a submodular value is the summation of elements in f_B.
– Data storage and communication efficiency: To save and transfer 2^N submodular values is infeasible. Due to the sparsity, the number of f_B coefficients is only b. The robot only needs to save f_B and send it to the other robots. In other words, this approach already compressed the data into a sparse format.
– Invariant property: If the subgoal ground set is the same, the learned f_B can be reapplied to the other environments. The robot only needs to update the nonzero values in f_B instead of 2^N values in the spatial domain.

5 Learning CTG function

Section 5.1 introduces the Q-learning. Section 5.2 introduces the proposed reinforcement learning algorithm for search.

Definition 8 (*Learning CTG functions*) Given the current motion cost Δt, the robot position (X_R) at time k, and the subgoal position (X_G), the objective is to find an approximate CTG function $T(X_R, X_G)$.

As shown in the scenario described in Fig. 11a, the robot has five possible actions and chooses the third action. After arriving at the cell, the robot gets the motion cost (Δt). The CTG function is approximated through the sequential actions and motion costs using Q-learning.

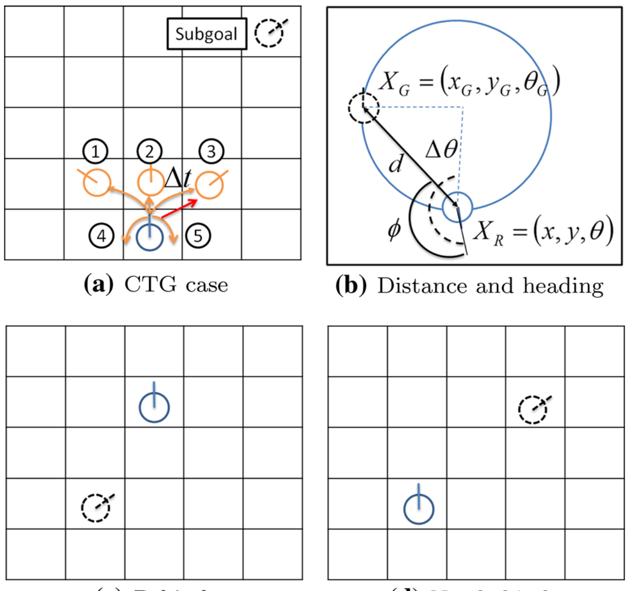


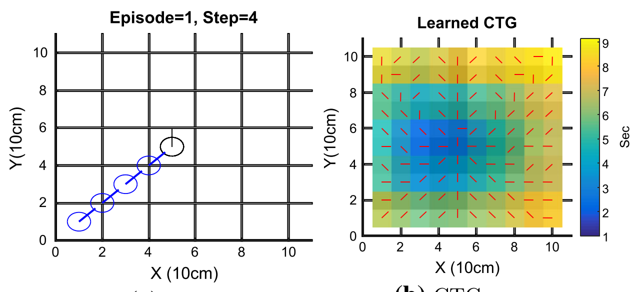
Fig. 11 Illustration of CTG learning problem and selected features. **a** The *blue circles* represent the current robot position, *orange circles* are next possible actions, the *black circle* is subgoal position, the *red arrow* is the optimal action. The robot has five possible actions, (*1*) upper left, (*2*) upper, (*3*) upper right, (*4*) turn left and (*5*) turn right. It chooses the 3rd action with lowest motion cost (Δt). **b** The *black circles* are subgoals, the *blue circle*is robot position and the *black line* is the distance between the robot and subgoal. The *dash curve* represents the heading difference and the *solid curve* represents the direction angle. **c** The subgoal is behind. **d** The subgoal is not behind

5.1 Q-learning

For the guidance problem, the goal is known, therefore the CTG map for simple motion primitives is usually solved offline via Dijkstra's algorithm (Bellingham et al. 2002; Mettler and Kong 2008). For search problems, there are N subgoals. It is infeasible to compute all CTG functions. The robot has to learn CTG functions online. In Richards and Boyle (2010), the author adopted reinforcement learning for CTG function based on a tubular Q-function, which represents the function values by a table. This approach required hundreds of episodes[4] to achieve a good solution. In the present research, a linear Q-function composed of four features is proposed to reduce the number of episodes.

The CTG function is $\hat{Q} = T(X_R, X_G) = \sum_i w_i f_i$, where f_i is the ith feature function, w is the weighting vector, X_R is the robot position and X_G is the subgoal position. If X_G is fixed, the CTG function is a 3D function. Given any robot position ($X_R = \{x, y, \theta\}$), the motion cost is computed through the CTG function. The four features are as follows:

[4] The number of episodes means the number of trials in reinforcement learning.



(a) Actions **(b)** CTG map

Fig. 12 Illustration of the learning CTG function process. **a** The *blue* and *black circles* represent the robot trajectories and goal position. **b** The *red lines* and *color cells* represent the optimal direction and cost of CTG function for each cell. The robot is at (1, 1, 45°) and the goal position is (5, 5, 90°). The CTG value in goal cell (5, 5, 90°) is minimal. The *red line* in this cell shows the optimal direction

Distance (d)**:** As Fig. 11b shows, the distance between the robot and the goal is an important feature in computing the motion cost. d is computed by the robot position X_R and subgoal position X_G.

Heading difference ($\Delta\theta$): As Fig. 11b shows, the heading difference between the robot and the goal is another important feature.

Direction angle (ϕ): As Fig. 11b shows, the angle between robot heading and the vector $\overrightarrow{X_R X_G}$ is defined as direction angle. If the robot is heading to the subgoal, the motion cost should be less. Hence, ϕ is also an important feature.

Behind: If the subgoal is behind (see Fig. 11c), the robot needs to turn and move toward subgoal. It will take more motion cost. In this case, this feature value is 1. If the subgoal is behind (see Fig. 11d), the robot only needs to move toward to subgoal. It will take less motion cost. In this case, this feature value is 0.

Figure 12a shows, the robot in a 10 × 10 grid with 45° heading resolution. The number of feasible states is 800. The robot chooses the optimal actions (a_t) with the minimal motion cost according to the initial CTG function ($\hat{Q}$). Upon arrival at the next cell (s_{t+1}), it gets the motion cost (Δt) as a reward R and uses it to update the weighting vector of CTG through equation 3. After 5 steps, it arrives at the goal. The learned CTG function is used to compute the motion cost for the 800 states. This 3D cell map is called CTG map. It gives the direction with minimal motion cost for each cell (see Fig. 12b). After 3–5 episodes, the CTG function is approximated by Q-learning.

$$\begin{aligned} a_t &= \min_{a_t} \hat{Q}(s_t, a_t) \\ \Delta R &= [R(s_t, a_t) + \gamma \min_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1}) - \hat{Q}(s_t, a_t)] f_i \\ w &\leftarrow w + \alpha_1 \Delta R \end{aligned} \tag{3}$$

where s_t is the state, a_t is action at time t, γ is the discount factor, R is the reward and α_1 is the learning rate.

5.2 Reinforcement learning algorithm

```
1: while not arriving $K^{th}$ subgoal ($S_{G,K}$)
   or not finding the target do
2:   if not arriving current subgoal ($S_{G,k}$) then
3:     $a_t = \min_{a_t} \hat{Q}(s_t, a_t)$
4:     execute an action using receding horizon control
5:     $R(s_t, a_t) = \Delta t$
6:     $\Delta R = [R(s_t, a_t) + \gamma \min_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1}) - \hat{Q}(s_t, a_t)] f_i$
7:     $w \leftarrow w + \alpha_1 \Delta R$
8:     if the cell is not visited then
9:       save $z_i$ to database.
10:    end if
11:    generate subgoal-PD data $X_{s,i}$
12:    batch $X_{s,i}$ into $X_s$
13:  else
14:    randomly choose a subgoal with $\epsilon$ probability.
15:    $S_{G,k+1}$=SFSS_subgoal($X_s$, $S_{G,k}$, $w$) with $1-\epsilon$
       probability.
16:  end if
17: end while
```

Algorithm 1: Reinforcement learning for subgoals

Algorithm 1 shows one episode of the RL algorithm. Lines 3–7 show the Q-learning for the CTG function. The CTG function $\hat{Q}$ is composed of four features and one constant term. The CTG weighting vector (w) is updated at each step. Lines 8–10 show the robot saving the measurement z_i into the database if this cell is not visited. Lines 11–12 show the subgoal-PD data randomly generated from the database. Lines 14–15 show determination of the next subgoal once the robot arrives at the subgoal, the robot chooses the subgoal by SFSS_subgoal algorithm with $1-\epsilon$ probability or randomly chooses the subgoal with ϵ probability for exploration. ϵ is a multiplicative inverse of the number of visited subgoals. When the robot has not explored any subgoals, $\epsilon = 1$. After arriving at the Kth subgoal, the robot completes this episode and saves (w, f_B) for next episode.

Algorithm 2 shows how to compute near-optimal subgoals through SFSS. Lines 2–3 show the subgoal-PD data are assigned to A and f_M for sparse regression. Lines 5–9 show the computation of Ψ based on A and B. Lines 12–13 show the computation of f_B using fast iterative soft-threshold algorithm (FISTA) (Beck and Teboulle 2009), where α_2 is the step size of gradient descent, which is decided by a line search algorithm. Lines 17–19 show the computation of the submodular values f_P and motion cost T using a greedy algorithm. Line 20 shows the greedy selection of subgoals based on f_P and T. Once the set with maximum cumulative PD is chosen, it is set as the next subgoal.

```
1: $X_{sg,k+1}$=SFSS_subgoal($X_s$, $S_{G,k}$, $w$)
2: $A_{(m,1:n)} = X_{s(m,1:n)}$
3: $f_{M(m,1)} = X_{s(m,n:n+1)}$
4: // Compute basis matrix
5: for i=1,..,m do
6:     for j=1,..,b do
7:         $\Psi_{i,j} = (-1)^{A_i \cap B_j}$
8:     end for
9: end for
10: // Reconstruct $f_B$
11: while not converge do
12:     $f_{B,k+1} = \tau_\lambda(f_{B,k} - 2\alpha_2 \Psi^T(\Psi f_B - f_M))$
13:     where $\tau_\lambda(f_B) = max(|f_B| - \lambda, 0) sign(f_B)$
14: end while
15: // Reconstruct a submodular function
16: for j=1,..,n do
17:     $S \leftarrow s_j \cup S_{G,k}$
18:     $\hat{f}_P(S) = \sum_B f_B(B)\psi_B(S)$
19:     $\hat{T}(X_R, X_{s_j}) = \sum_i w_i f_i(X_R, X_{s_j})$
20:     $s_G \leftarrow arg\max_s \frac{\hat{f}_P(S) - \hat{f}_P(S_{G,k})}{\hat{T}(s_j)}$
21: end for
22: $S_{G,k+1} \leftarrow s_G \cup S_{G,k}$
23: return $S_{G,k+1}$
```

Algorithm 2: SFSS learning for subgoals

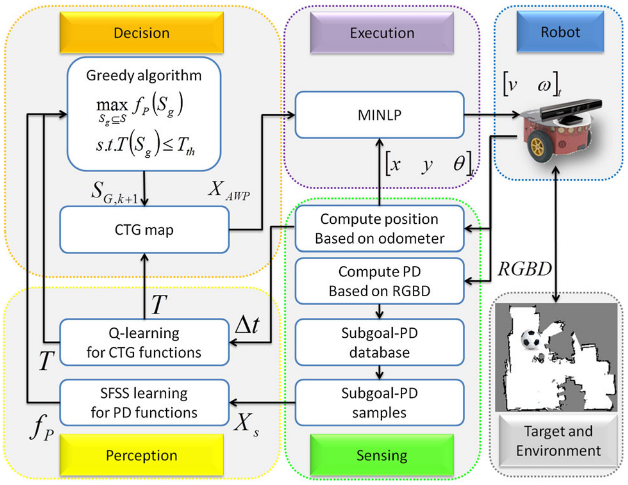


Fig. 13 The data flow of the search system

The data flow of the search system is illustrated in Fig. 13. The system consists of sensing, perception, decision and execution. The robot perceives RGBD and odometer data. In the sensing stage, the RGB data is used to compute and batch subgoal-PD data (X_s). The odometer is used to compute the robot position (x, y, θ). In the perception stage, (X_s) is used to approximate a submodular function using SFSS. The motion time Δt is used to approximate a CTG function using Q-learning. In the decision stage, a greedy algorithm computes the next near-optimal subgoal $S_{G,k+1}$ according to Eq. 1. Once the next subgoal is known, the robot computes a CTG map for receding horizon control. In the execution stage, MINLP computes (v, ω) commands according to active waypoint (X_{AWP}) and subgoal ($S_{G,k+1}$) and the current robot position (x, y, θ).

6 Experiments

To verify the proposed algorithms, three experiments with different objective functions were conducted (see Table. 2). In Sect. 6.2, the experiments only consider maximum coverage. The experiment is terminated when 12 subgoals are found. At that point, the coverage computed with the greedy, PMAC, FSS and SFSS algorithms are compared. In Sect. 6.3, the experiments consider maximum coverage with motion cost. In this case, the experiment is terminated when the robot finds 20 subgoals.[5] The coverage of PMAC+Q and SFSS+Q are compared, where PMAC+Q presents that the submodular function is approximated by PMAC while CTG is approximated by Q-learning. In Sect. 6.4, the experiments consider maximizing cumulative PD with motion cost. In this case, the experiment is terminated when the probability of a H-area is over 90%. The search time of PD and SFSS+Q are compared. In Sect. 6.5, the complexity and accuracy of learning approaches are discussed (e.g., PMAC, FSS and SFSS). In Sect. 6.6, the performance of non-learning models are discussed (e.g., PD and EPD). The setup is described in the following section.

6.1 Experimental setup

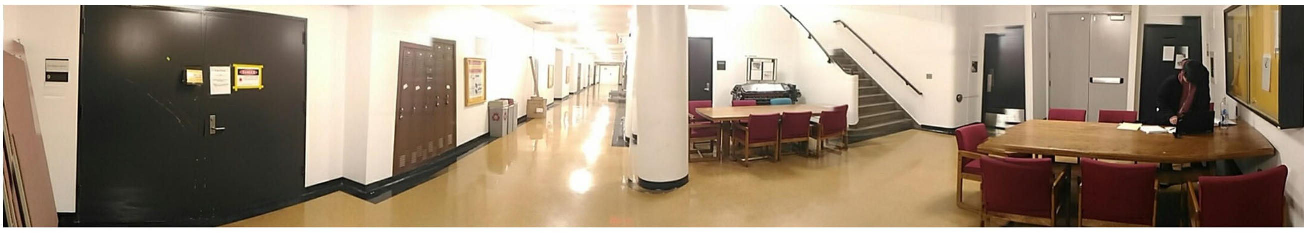
The experiments were conducted at the University of Minnesota's Interactive Guidance and Control Lab (IGCL) hallway (see Fig. 14). The computational platform is a Lenovo X230 with 2.6 GHZ CPU and 8 GB RAM. The mobile robot is a Pioneer P3DX equipped with a Microsoft Kinect. The non-holonomic platform is controlled via the speed (v) and turn rate (ω). The robot gets the odometer from the encoders and RGBD data from Kinect. The odometer data is used for computing the robot position. The horizontal field of view and maximal range of Kinect are 57° and 4 (m), resepctively. The 3D RGBD data is projected to a 2D grid map for computing the sensing coverage and probability. The target is an orange soccer ball. It is detected through Hough circle transform in HSV spaces. The target probability distribution is updated based on Bayes filter (Tseng and Mettler 2015).

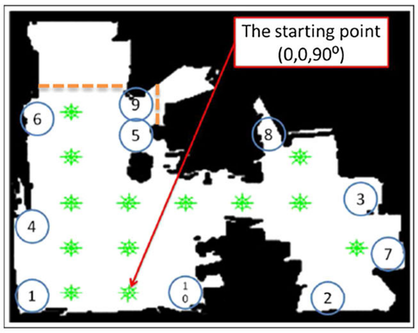
At the subgoal level, the greedy, FSS, SFSS and PMAC approaches generate subgoals for the motion control level. b^2_{FSS} denotes FSS takes 2nd order basis. In the EX1, b^2_{FSS} is $\binom{104}{0} + \binom{104}{1} + \binom{104}{2} = 5461$. Since the parts of 2nd order

[5] In EX1, 12 subgoals can cover the environments over 80%. In EX2, since the motion cost is considered, the robot needs more subgoals to cover over 80%.

Table 2 Objective functions of three experiments

	Section 6.2 Objective EX1	Section 6.3 Objective EX2	Section 6.4 Objective EX3
	$\max_{S_g \subseteq S} f_C(S_g)$	$\max_{S_g \subseteq S} f_C(S_g)$	$\max_{S_g \subseteq S} f_P(S_g)$
		$s.t.\ \ T(S_g) \leq T_{th}$	$s.t.\ \ T(S_g) \leq T_{th}$
Terminal condition	$\lvert S_G \rvert = 12$	$\lvert S_G \rvert = 20$	$P(A) \geq 0.9$
Parameters		$T_{th} = 150$	$T_{th} = 150$
			$g = 1$


(a) 180° panoramic view



(b) Map, subgoal set and target locations

Fig. 14 IGCL hallway environment. **a** IGCL hallway picture. **b** The *green circles* represent the subgoal positions, lines are heading directions and *numbers* indicate the target locations. The target is not beyond the *dashed lines*

bases are discarded, the b^2_{SFSS} is 1491. In the EX2 and EX3, b^3_{FSS} is $\binom{104}{0}+\binom{104}{1}+\binom{104}{2}+\binom{104}{2}+\binom{104}{3} = 187,565$, which is infeasible for online learning. Hence, FSS approaches is only feasible at 2nd order basis. Since the parts of 2nd and 3rd order bases are discarded, the b^3_{SFSS} is 3566. PMAC is solved using the perceptron learning algorithm. FSS and SFSS are solved using FISTA algorithms (Beck and Teboulle 2009). The number of iterations for perceptron and FISTA algorithms is 500.

At the motion control level, given the current robot position and AWP, the mixed-integer nonlinear program (MINLP) computes the optimal velocity commands (v, ω) to reach AWP. The horizon (H) of MINLP has 8 steps. The neighbor cells around the robot are candidates for AWP. The robot chooses the cell with the lowest cost as AWP according to the learned CTG function. If the distance between the obstacle and robot is smaller than 0.5 m, the robot needs to avoid it. The robot will choose a safe AWP based on VFH (Vector Field Histogram) (Borenstein and Koren 1991). The velocity commands are updated at 1 Hz and sensor data is updated 4 Hz. Once the robot arrives at the current subgoal, the robot will move toward the next subgoal until the robot visits all subgoals or finds the target. More details about MINLP formulation can be found in Tseng and Mettler (2015).

The map parameters reviewed in Table 3 are as follows: The grid map of IGCL is built using FastSLAM (Montemerlo et al. 2003). The map is discretized using 5 by 5 cm cells. The CTG map resolution is related to the minimal planning length of a horizon. Based on the velocity constraints, 10*cm* is a feasible resolution. The grid map resolution is decided by the environmental size (10×10 m^2) and the grid map size (300×300). The subgoal resolution decides the number of subgoal ground set (N). If the resolution is higher, N is larger. Consideration of computation, the 1.5m $\times$ 1.5 m $\times 45°$ is feasible. There are 13 subgoal locations and each one has 8 heading directions. Hence, the number of feasible subgoal (N) is 104 (see Fig. 14b).

Table 3 Map parameters

IGCL hallway map size	10 m× 10 m
CTG map resolution	10 cm× 10 cm
Grid map resolution	5 cm× 5 cm
Subgoal resolution	1.5 m × 1.5 m ×45°

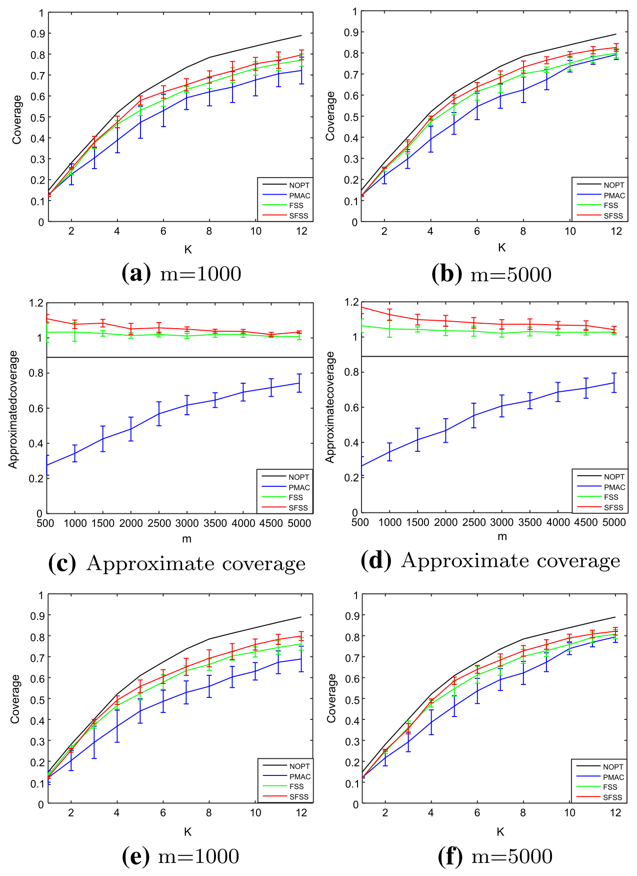
Table 4 Experiment parameters

N	γ	α_1	λ	K_s
104	0.9	0.01	5×10^{-4}	12
H	V_{max}	ω_{max}	a_{max}	α_{max}
8	20 (cm/s)	22 $(°/s)$	5(cm/s^2)	11 (°/s^2)
b^2_{FSS}	b^2_{SFSS}	b^3_{SFSS}		
5461	1491	3566		

The experiment parameters in Table 4 are as follows: γ is the discount factor of a MDP model. It is the trade-off between myopic and nonmyopic behavior when pursuing rewards. When $\gamma = 0$, the robot is myopic to current rewards. When $\gamma = 1$, the robot considers receiving long-term rewards. Hence γ is set as 0.9. α_1 is the learning rate of Q-learning. Higher λ values decide the sparisty of f_B. K_s is the maximal number of sampling subgoals. K_s cannot be too small (< 2) or too high (> 40). If K_s is too high, the coverage values are close to 1. If K_s is too low, the coverage values are close to 0. In both cases, the sampling data is not useful for learning. $H = 8$ is decided based on the computation cost of MINLP (< 0.2 s). The maximal velocity and acceleration ($V_{max}, \omega_{max}, a_{max}, \alpha_{max}$) are the trade-off between the speed and detection rate. For example, higher speeds lead to image blur, which decreases the detection rate.

6.2 EX1: Coverage experiments

The greedy algorithm generates 12 near-optimal (NOPT) subgoals offline. These results are used as benchmark. Ten subgoal-coverage data are collected for evaluation. The coverage obtained with greedy, PMAC, FSS and SFSS are compared using different numbers of samples ($m = 500$–5000). The coverage is computed based on the subgoals chosen by the learned submodular functions of PMAC, FSS, and SFSS. Figure 15a shows that when $m = 1000$, the SFSS outperforms FSS and PMAC. Since FSS have 5461 unknown variables in f_B, 1000 samples are not enough to achieve convergence. PMAC needs large samples to convergence as claimed in Balcan and Harvey (2011). Figure 15b shows that when $m = 5000$, the SFSS and FSS outperform PMAC. Since FSS and SFSS have enough samples, their performance is close to the NOPT. Figure 15c shows that the predicted error of SFSS and FSS is less than 10% while that



(a) m=1000 **(b)** m=5000 **(c)** Approximate coverage **(d)** Approximate coverage **(e)** m=1000 **(f)** m=5000

Fig. 15 Coverage and approximate coverage of NOPT, PMAC, FSS and SFSS approaches. **a** The greedy coverage based on PMAC, FSS and SFSS estimation versus subgoal number K when m = 1000. **b** The greedy coverage based on PMAC, FSS and SFSS estimation versus subgoal number K when m = 5000. **c** Approximate coverage of PMAC, FSS and SFSS versus sample number (m = 500–5000) at $K = 12$. **d–f** The measurements are with Gaussian noise

of PMAC is around 10–50%. Since these approaches choose subgoals based on relatively largest coverage values, better approximation does not guarantee better a subgoal selection. For example, the approximation of FSS is closer than that of SFSS (see Fig. 15c). However, Fig. 16a, b show that SFSS coverage is higher than FSS and PMAC coverage.

To evaluate the performance under noisy measurements, the measurement model is as follows:

$$f_c = f_C(S_g) + \eta, \quad \eta \sim N(0, \sigma^2) \tag{4}$$

where, f_c is noisy coverage measurement, $f_C(S_g)$ is the ground true coverage given subgoal set S_g, η is a Gaussian noise with 3%[6] standard deviation (σ).

Figure 15d shows that the approximate coverage of SFSS and FSS are affected by the noisy data while the approxi-

[6] 3% is a noisy measurement since the maximal coverage of one subgoal is less than 15%.

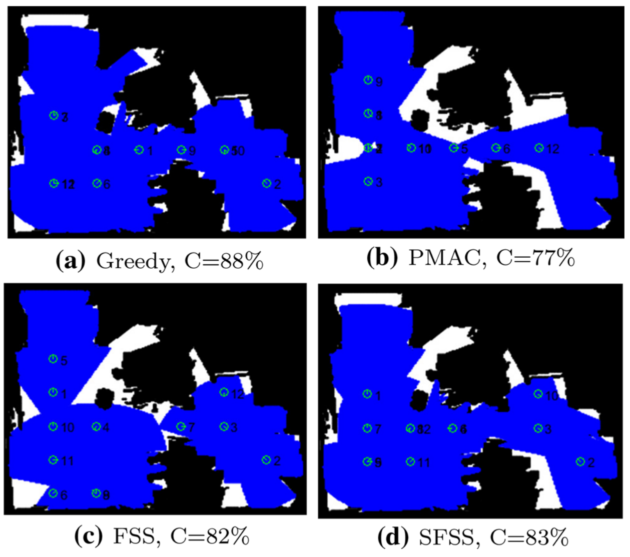

(a) Greedy, C=88% **(b)** PMAC, C=77% **(c)** FSS, C=82% **(d)** SFSS, C=83%

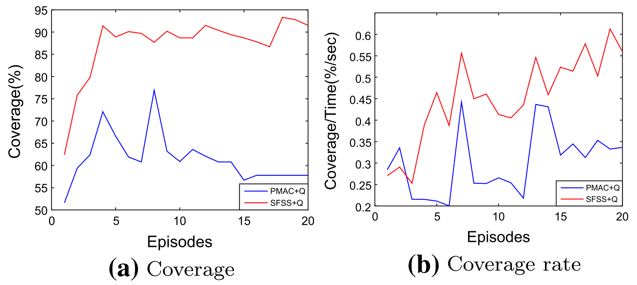
Fig. 16 Coverage and subgoals of greedy, PMAC, FSS and SFSS approaches ($m = 5000$, $K = 12$). The *green circles* represent the chosen subgoals, *blue areas* are corresponding covered area, *white areas* are uncovered area, *black areas* are obstacles and *numbers* represent the order of subgoals

mate coverage of PMAC does not be affected. Figure 15e, f show that the noisy data does not significantly affect coverage of the three approaches. Hence they are robust to noisy measurements. Since FSS and SFSS are solved based on FISTA, which is a least square method, they are robust to noisy data. PMAC method involves projecting submodular data to upper/lower bounds, then use a classifier to approximate a submodular function. This explains why the noise resistance of PMAC is good (see Fig. 15d). However, it also implies that its performance will not be improved even if accurate data is provided.

EX1 shows that the coverage and mean predicted error of FSS and SFSS outperform that of PMAC using the same size of samples. Since SFSS discards parts of 2nd order bases, it converges faster than FSS. Since PMAC chooses the relative maximal coverage from subgoal ground set even if its prediction is not accurate, the result shows that PMAC coverage still can generate subgoals with 77% coverage. Although FSS and SFSS have similar coverage performance, SFSS computation is 10 times faster than FSS (see Sect. 6.5).

6.3 EX2: Coverage with motion cost experiments

To further consider the motion cost, the objective function EX2 shown in Table 2 is used. The experiments are terminated when the robot visits 20 subgoals. Since the computation of FSS is too high for online learning, the PMAC+Q and SFSS+Q coverage are compared and both approaches are executed for 20 episodes. The learning algorithms are shown in Algorithm 1 and 2. In the first episode, the robot only



(a) Coverage **(b)** Coverage rate

Fig. 17 Coverage and coverage time of PMAC+Q and SFSS+Q approaches

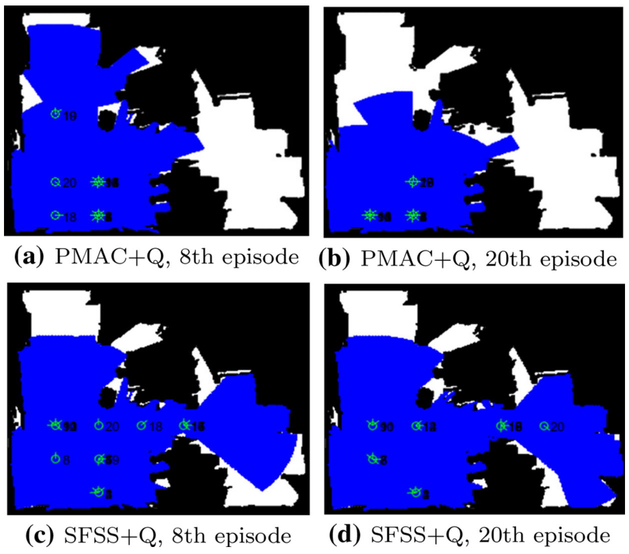

(a) PMAC+Q, 8th episode **(b)** PMAC+Q, 20th episode **(c)** SFSS+Q, 8th episode **(d)** SFSS+Q, 20th episode

Fig. 18 Comparison of coverage and subgoal configuration for PMAC+Q and SFSS+Q approaches at 8th and 20th episodes

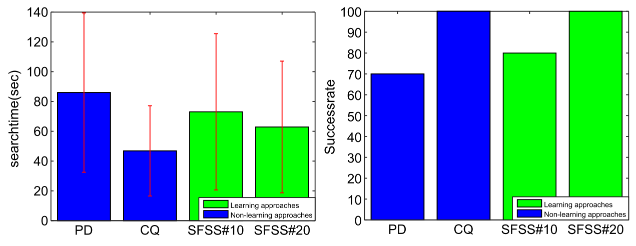
knows the subgoal-coverage data in the starting point and needs to explore new subgoals. The robot chooses the nearest unexplored subgoal for the exploration mode and chooses the next subgoal based on learned functions for the exploitation mode. The probability of exploration decreases as the number of the episodes increases. After 10 episodes, the PMAC+Q and SFSS+Q already explored all subgoals.

During episodes 1–10, since the robot explores new subgoals with high probability, the PMAC+Q and SFSS+Q coverage increase as the number of episodes increases (see Fig. 17a). Since the SFSS approximation is more accurate than PMAC, it chooses better subgoals than PMAC+Q when in exploitation mode. Figure 18a, c show that SFSS+Q coverage is higher than PMAC+Q coverage at 8th episode.

During episodes 11-20, since the robot is always in exploitation mode, the subgoal selection is based on learned functions. SFSS approximation is accurate, therefore its coverage is over 85% after 10 episodes (see Fig. 17a). Since PMAC approximation is not accurate, and has lower values than true coverage (see Fig. 15c), the motion cost dominates the subgoal selection procedure. Hence, PMAC+Q chooses

Table 5 Overview of the algorithms tested in the EX3

Approaches	The ways to compute f_P and T
PD	f_P is computed through ray-tracing techniques offline and T is computed through Q-learning offline PD model considers the maximal PD within 4 steps
CQ	f_P is computed through ray-tracing techniques offline and T is computed through Q-learning offline
SFSS#10	f_P is computed through SFSS online and T is computed through Q-learning online
SFSS#20	f_P is computed through SFSS online and T is computed through Q-learning online



(a) Search time **(b)** Success rate

Fig. 19 Search time and success rate of PD, CQ, SFSS#10 and SFSS#20 approaches

low motion cost subgoals (see Fig. 18b), as a result the coverage rate of PMAC+Q is around 0.3 while that of SFSS+Q is around 0.5 (see Fig. 17b. The coverage of PMAC+Q selected subgoals is lower than 60% (see Fig. 17a).

This experiment shows that the inaccurate approximation leads PMAC+Q to choose neighbor subgoals, which could result in low coverage (see Fig. 18b). In other words, PMAC approximation accuracy does not affect coverage when the motion cost is discarded but it seriously affects coverage when motion cost is accounted for.

6.4 EX3: Search with motion cost experiments

To further consider the sensing uncertainty, the objective function EX3 shown in Table 2 is used. Non-learning approaches (e.g., PD and CQ) and learning approaches (e.g., SFSS#10 and SFSS#20) are compared (see Table 5). Non-learning approaches utilize map information to compute f_P. Learning approaches do not utilize the map information to compute f_P. The PD method considers the maximal PD values within 4 steps and only move to the 8 neighbor subgoals (see Definition 4). The coverage+Q (CQ) method adopts the proposed objective function (see Eq. 1). The SFSS#10 and SFSS#20 are the SFSS+Q approaches at 10th and 20th episodes. To evaluate the search performance of four approaches, the search time and successful rate are compared. For each approach, the robot searches for the target 100 times. The target is put at different locations for each time. If the robot cannot find the target before T_{th}, the search time is T_{th}.

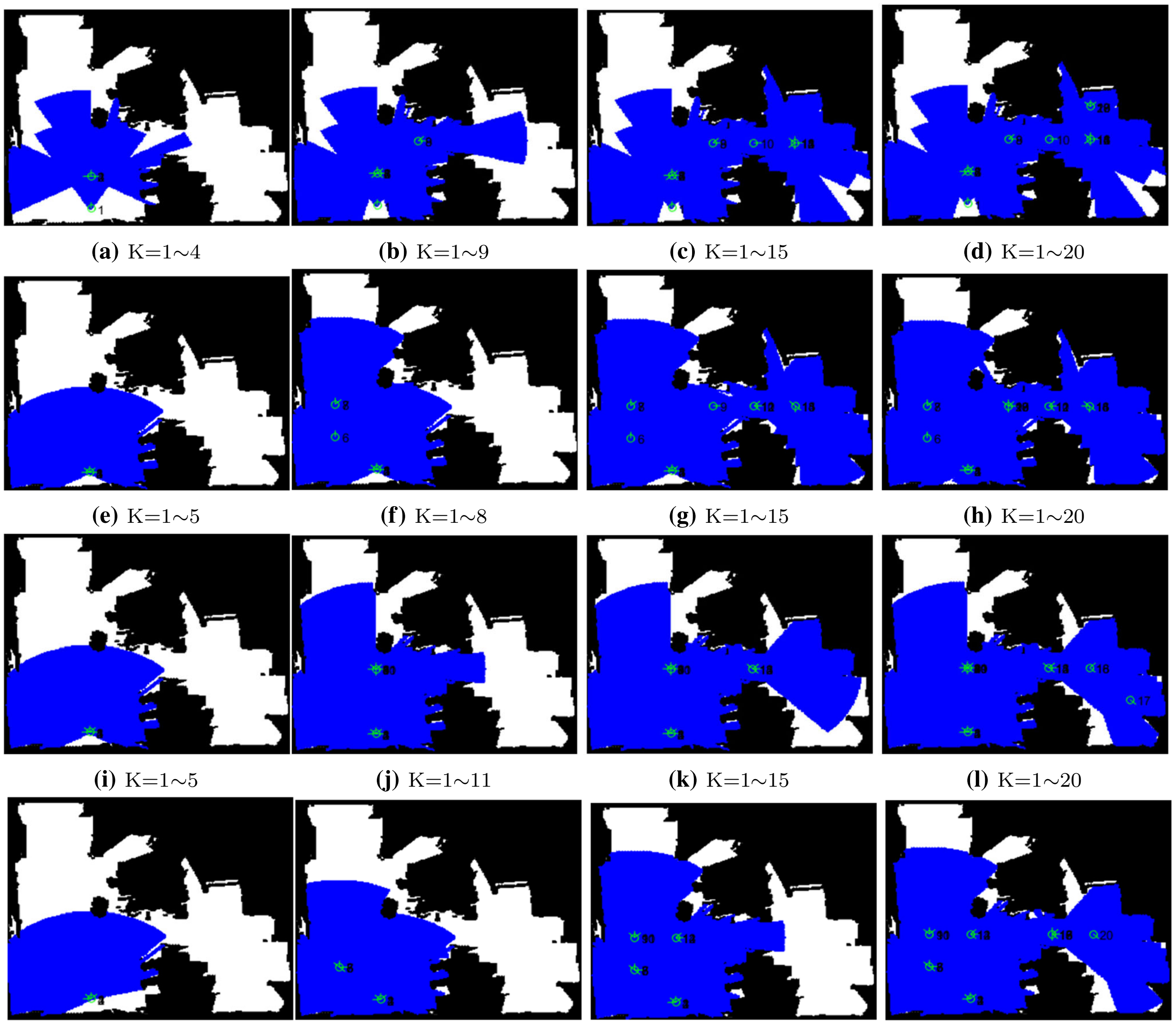
Figure 20a–d show that the PD approach reaches maximal coverage within 4 steps. Due to the motion constraints of the PD model (e.g., moving to neighbor subgoals), the robot does not cover the upper left side of the room, therefore, it cannot detect the target at 5th, 6th and 9th locations, and the success rate is 70% (see Fig. 19b). Figure 20e–h show that the CQ's behavior involves turning on the spot of original position, and then searches the left and right side of the room. The robot can cover over 90% area using only 15 subgoals (see Fig. 20c). Figure 19 shows that for non-learning approaches, the CQ's search time is almost 2 times faster than that of PD.

For learning approaches, Figure 20i–l show that the SFSS#10's behavior is to turn on the spot at the original position, move forward and then search the right side of the room. Since the f_P is not accurate enough, the robot cannot detect the target at 5th and 9th locations and the success rate is 80% (see Fig. 19b). Figure 20m–p show that the SFSS#20's behavior is to turn on the spot at the original position, then search the left and right sides of the room. Figure 19 shows that for learning approaches, the search time of SFSS#20 is faster than that of SFSS#10, since the f_P is more accurate.

The CQ demonstration for target at location 7 is illustrated in Fig. 21. At time 13 (s), the robot visits the 5th subgoal (see Fig. 21a, e, i). Since the robot already searched for most area from the original position, moving to another position will give more coverage. At time 47 (s), the robot explores most space on the left side and starts to search the new space on the right side (see Fig. 21b, f, j). At time 66 (s), the robot visits the 8th subgoal and explores the space on the right side (see Fig. 21c, g, k). Since the target is beyond the Kinect sensing range (< 4 m), the target cannot be detected. At time 73 (s), the robot detects the target at location 7 several times before arriving at 10th subgoal (see Fig. 21d, f, l). The probability of H-area is over 90%. Based on these results, the robot has been able to successfully find the target.

These experiments demonstrate that for non-learning approaches, CQ outperforms PD. Meanwhile, learning approaches SFSS#10 and SFSS#20 outperform PD even if the map information is not utilized. SFSS#20 shows that the robot can search for the target by selecting near-optimal subgoals based on approximated f_P and T, and it can converge within 10+ episodes.[7]

[7] SFSS can converge within 2000–3000 samples but it needs to explore the environment. Hence, it takes 10+ episodes.



(a) K=1∼4 **(b)** K=1∼9 **(c)** K=1∼15 **(d)** K=1∼20

(e) K=1∼5 **(f)** K=1∼8 **(g)** K=1∼15 **(h)** K=1∼20

(i) K=1∼5 **(j)** K=1∼11 **(k)** K=1∼15 **(l)** K=1∼20

(m) K=1∼4 **(n)** K=1∼8 **(o)** K=1∼14 **(p)** K=1∼20

Fig. 20 Subgoals coverage of PD, CQ, SFSS#10 and SFSS#20. **a**–**d** PD. **e**–**h** CQ. **i**–**l** SFSS#10. **m**–**p** SFSS#20. The *blue area* is the covered area, *green circles* are subgoals positions, and the number is the index of subgoals

6.5 Complexity and accuracy analysis

Table 6 shows the time and sample complexity of PMAC, FSS, and SFSS. In the EX1, N is 104, b is 5461, β is 0.27, k is 1400, ϵ is 0.1 and δ is 0.1. The time complexity of the three approaches is as follows: Since $b = O(N^2)$, b is bigger than N. Based on time complexity, PMAC is faster than SFSS and FSS. Since $\beta < 1$ holds, SFSS is faster than FSS.

The sample complexity of three approaches is as follows: As Table 6 shows, given these parameters, the needed sample number for PMAC, FSS and SFSS are about 248166, 5232 and 4442, respectively. As EX1 shows that PMAC needs large amount of samples to converge while SFSS and FSS only need a few thousand. Since the number of bases used by SFSS is fewer than for FSS, its convergence is faster.

In summary, the complexity of the three approaches is as follows: First, PMAC is the most efficient algorithm to approximate submodular functions since it only uses $N + 1$ variables to describe them. On the other hand, it is impossible to describe 2^N values using $N + 1$ variables so its prediction is not accurate. It also needs large training samples. Second, SFSS and FSS need $O(N^2)$ bases, hence its time complexity is more than PMAC complexity. But the two approaches need fewer samples to approximate submodular functions. Moreover, at the same sample number, the prediction achieved with FSS and SFSS are more accurate than that of PMAC.

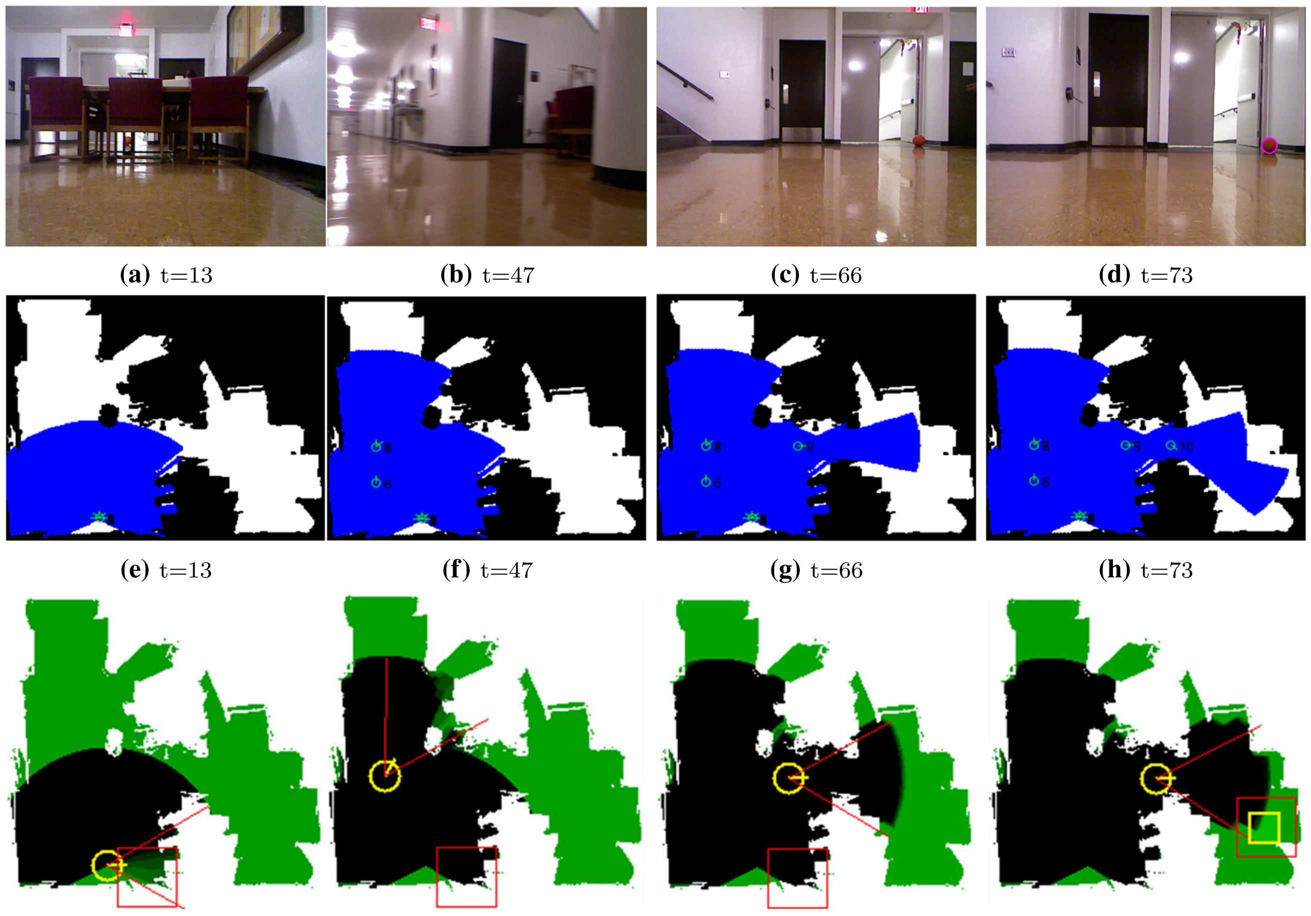


Fig. 21 Search using CQ. **a–d** RGB image frames at times t = 13, 47, 66 and 73 s. *Pink circles* represent the detection of the target. **e–h** Coverage human machine interface (HMI) and **i–l** PD HMI at selected frames. The *green area* and *black area* represent higher and lower probability in the logarithmic form. The *red rectangle* represents the H-area, the *yellow rectangle* is detected area, *red line* is the field of view (FOV), the *yellow circle* is robot position. The image frames show: t = 13: the robot arrives at 5th subgoal; t = 47: the robot arrives at 8th subgoal; t = 66: the robot arrives at 9th subgoal but the target is too far to detection; and finally at t = 73: the robot detects the target and the probability of H-area is over 90% before arriving at 10th subgoal

Table 6 Complexity of PMAC, FSS and SFSS

	PMAC	FSS	SFSS
Computational complexity	$O(MN^2)$	$O(Mb^2)$	$O(M\beta^2 b^2)$
Sample complexity	$O(\frac{48N}{\epsilon} log \frac{9N}{\delta\epsilon})$	$O(Klogb)$	$O(Klog(\beta b))$
Notation	M: the number of iterations N: the number of subgoals δ: the probability of that the Sample is not correctly classified ϵ: the probability of that The estimated submodular value is Not within $\sqrt{N}/\log(N)$ bound	b: the number of bases K: the number of nonzero coefficients	β: reduction rate

Finally, since SFSS requires fewer bases than that FSS, the time and sample complexity of SFSS is less than that of FSS. In the EX1, SFSS only needs 85% of the samples needed by FSS, and SFSS's computation is 13 times faster than FSS. Hence, considering accuracy, sample number and computation, SFSS is the best approach for learning submodular functions. The results of three experiments also support this complexity analysis.

6.6 PD and EPD model

The proposed objective function in Eq. 1 adopts EPD model (see Definition 5). The major difference between the original PD model and EPD model is the path constraint. The PD model only considers neighbor subgoals while the EPD model considers all subgoals. As Fig. 14 shows, there are 8 directions for each subgoal. If the robot is at the original subgoal and tries to find 10 steps, such that the PD is maximal using branch and bound, the computation is 8^{10}. If the robot adopts greedy algorithms and the path constraints are skipped, the computation is 104×10. In the EX3, PD takes 1 hour to find 20 subgoals while CQ using the EPD model takes 5 seconds to find 20 subgoals. Moreover, the EPD approach gives a near-optimal guarantee. In other words, breaking the path constraints will generate more branches, but the greedy approach will give near-optimal solution with linear time complexity due to the submodularity.

7 Discussion and future Work

7.1 Extensions to different robots and sensors

The proposed method is applied to a mobile robot with Kinect in a 2D grid map. To apply the method to different robots with different sensors in a 3D environment, some parameters need to be changed as follows:

First, changing the sensors affects the sensing range and FOV of the sensors. Since coverage computation is through a ray-tracing technique, this technique can be applied to 2D or 3D sensors (e.g., UTM-30LX laser scanner, Asus Xtion and downward facing cameras) (Renzaglia et al. 2012; Charrow et al. 2015). The covered area is computed based on the range and FOV of sensors. Hence, the users only need to modify the range and FOV parameters.

Second, changing the robot affects parameters of MINLP. CTG and receding horizon control approaches are originally proposed for aerial robots (Mettler and Kong 2008). The velocity, acceleration constraints and kinematic models in MINLP need to be changed according to the new robots' parameters. The processes of learning PD and CTG functions are still the same. Therefore, the proposed method can be applied to different robots and sensors.

7.2 Future work

Since search encompasses the interaction between the searcher, target, and environments, future work will consider the following questions: First, finding better subgoal patterns can reduce the number of basis and training samples. One possible way is to analyze human subjects' subgoal patterns. Their subgoal patterns could inspire new algorithms to generate subgoal patterns which have sparsity in the Fourier domain. Second, it would be even better if what is learned during search could be reused by the robot in a new environment. Transfer learning technologies could give some clues about this question. Third, if the target is moving, the PD function becomes a time-varying submodular function. Analyzing the PD function in the Fourier domain could find the invariant properties for dynamic targets. Finally, based on existing technologies, human recognition ability still outperforms robots (e.g., detect and classify targets). Forming efficient human-robot search teams requires integrating them according to their respective strengths. Thus, how to allocate subtasks and design user interfaces are potential topics for cooperative search.

8 Conclusion

This paper presents a learning framework for efficient robotic search. The core element of the framework is a reformulated objective function combining maximal cumulative PD with motion cost. The robot planning is formulated as a sequence of trajectories segments defined by a finite set of subgoals with associate cost-to-go (CTG) functions. Since the PD and CTG functions depend on the environment, the robot has to learn the two functions. Learning is formulated using SFSS and Q-learning through sequential actions and perceptions. The robot chooses near-optimal subgoals based on the two functions until the target is detected.

The main contributions of this research are as follows: First, a reinforcement learning algorithm is proposed for solving search tasks. In other words, the robot can improve its search performance using data from its past experience. To the best of our knowledge, this is the first online learning approach for solving search tasks. Second, the proposed SFSS approach utilizes the sparse property of Fourier coefficients to reduce the number of bases without decreasing accuracy. To the best of our knowledge, the sparsity in the Fourier domain for learning submodular functions have not been investigated. The SFSS is the first algorithm exploiting this concept. Third, due to the submodularity, the learned subgoals give $(1-1/e)$ of the optimum with high probability. Finally, the experiments show that the proposed method (e.g., SFSS and CQ) outperforms existing learning approaches by a significant margin (e.g., PMAC and FSS) and benchmark models (e.g., PD).

Acknowledgements This research was completed thanks to the financial support from ONR Grant 1361538 and NSF CAREER CMMI 1254906. Kuo-Shih would like to thank his daughter, Chin-Chun Tseng. The hide-and-seek game they played inspires the learning concept for search problems.

References

Aggarwal, A. (1984). The art gallery theorem: Its variations, applications, and algorithmic aspects,. Ph.D. thesis, Johns Hopkins University.

Balcan, M. F., & Harvey, N. J. (2011). Learning submodular functions. In *Proceedings of the 43rd annual ACM symposium on theory of computing*.

Beck, A., & Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, *2*, 183–202.

Bellingham, J., Richards, A., & How, J. P. (2002). Receding horizon control of autonomous aerial vehicles. *American Control Conference*, *5*, 3741–3746.

Binney, J., & Sukhatme, G. S. (2012). Branch and bound for informative path planning. In *IEEE international conference on robotics and automation* (pp. 2147–2154).

Borenstein, J., & Koren, Y. (1991). The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, *7*(3), 278–288.

Bourgault, F., Furukawa, T., & Durrant-Whyte, H. F. (2003). Coordinated decentralized search for a lost target in a bayesian world. In *IEEE/RSJ international intelligent robots and systems* (pp. 979–1000).

Candes, E., Romberg, J., & Tao, T. (2006). Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transaction on Information Theory*, *52*(2), 489–509.

Charrow, B., Liu, S., Kumar, V., & Michael, N. (2015). Information-theoretic mapping using Cauchy–Schwarz quadratic mutual information. In *IEEE international conference on robotics and automation* (pp. 4791–4798).

Chen, W., Rodrigues, M. R. D., & Wassell, I. J. (2011). Distributed compressive sensing reconstruction via common support discovery. In *IEEE international conference on communications* (pp. 1–5).

Chung, T. H., & Carpin, S. (2011). Multiscale search using probabilistic quadtrees. In *IEEE international conference on robotics and automation* (pp. 2546–2553).

Cortes, J., Martinez, S., Karatas, T., & Bullo, F. (2004). Coverage control for mobile sensing networks. In *IEEE international conference on robotics and automation* (pp. 243–255).

Eagle, J. N. (1984). The optimal search for a moving target when the search path is constrained. *Operations Research*, *32*(5), 1107–1115.

Feige, U. (1998). A threshold of ln n for approximating set cover. *Journal of the ACM*, *45*(4), 634–652.

Gerkey, B. P., Thrun, S., & Gordon, G. (2006). Visibility-based pursuit-evasion with limited field of view. *The International Journal of Robotics Research*, *25*(4), 299–315.

Goerzen, C., Kong, Z., & Mettler, B. (2010). A survey of motion planning algorithms from the perspective of autonomous UAV guidance. *Journal of Intelligent and Robotic Systems*, *57*, 65. doi:10.1007/s10846-009-9383-1.

Hayashi, K., Nagahara, M., & Tanaka, T. (2013). A user's guide to compressed sensing for communications systems. In *IEICE transactions on communicationsE96-B*(3), 685–712.

Heng, L., Gotovos, A., Krause, A., & Pollefeys, M. (2015). Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments. In *IEEE international conference on robotics and automation* (pp. 1071–1078).

Hollinger, G., Choudhuri, C., Mitra, U., & Sukhatme, G. S. (2013). Squared error distortion metrics for motion planning in robotic sensor networks. In *Proceedings of the international workshop wireless networking for unmanned autonomous vehicles* (pp. 1426–1431).

Hollinger, G., Kehagias, A., & Singh, S. (2010). Gsst: Anytime guaranteed search. *Autonomous Robots*, *29*(1), 99–118.

Hollinger, G., & Singh, S. (2008). Proofs and experiments in scalable, near-optimal search by multiple robots. *Robotics: Science and Systems*,. doi:10.15607/RSS.2008.IV.027.

Hollinger, G., & Sukhatme, G. S. (2013). Sampling-based motion planning for robotic information gathering. *Robotics: Science and Systems Conference*,. doi:10.15607/RSS.2013.IX.051.

Kadane, J. B., & Simon, H. A. (1977). Optimal strategies for a class of constrained sequential problems. *The Annals of Statistics*, *5*(2), 237–255.

Khuller, S., Moss, A., & Naor, J. (1999). The budgeted maximum coverage problem. *Information Processing Letters*, *70*(1), 39–45.

King, J., & Kirkpatrick, D. (2011). Improved approximation for guarding simple galleries from the perimeter. *Journal Discrete and Computational Geometry*, *46*(2), 252–269.

Konolige, K. (1997). Improved occupancy grids for map building. *Autonomous Robots*, *4*, 351–367.

Kosmatopoulos, E. B. (2009). An adaptive optimization scheme with satisfactory transient performance. *Automatica*, *45*(3), 716–723.

Krause, A., & Guestrin, C. (2005). Near-optimal nonmyopic value of information in graphical models. In *Twenty-First conference on uncertainty in artificial intelligence* (pp. 324–331).

Krause, A., Singh, A., & Guestrin, C. (2008). Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *The Journal of Machine Learning Research*, *9*, 235–284.

Lau, H., Huang, S., & Dissanayake, G. (2006). Probabilistic search for a moving target in an indoor environment. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 3393–3398).

Lau, H., Huang, S., & Dissanayake, G. (2008). Discounted mean bound for the optimal searcher path problem with non-uniform travel times. *European Journal of Operational Research*, *190*(2), 383–397.

LaValle, S. M., & Kuffner, J. J. (1999). Randomized kinodynamic planning. *IEEE International Conference on Robotics and Automation*, *1*, 473–479.

Lloyd, S. P. (1982). Least-squares quantization in pcm. In *IEEE transactions on information theory* (pp. 129–137).

Lo, N., Berger, J., Noel, M. (2012). Toward optimizing static target search path planning. In *IEEE symposium on computational intelligence for security and defence applications* (pp. 1–7).

McCue, B. (1990). *U-boats in the bay of biscay: An essay in operations analysis*. Washington: National Defense University Press.

Mettler, B., & Kong, Z. (2008). Receding horizon trajectory optimization with a finite-state value function approximation. In *American control conference* (pp. 3810–3816)

Mettler, B., Tehrani, N. D., & Kong, Z. (2010). Agile autonomous guidance using spatial value functions. *Control Engineering Practice*, *18*(7), 773–788.

Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B. (2003). Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceed-*

ings of the sixteenth international joint conference on artificial intelligence (pp. 1151–1156).

Narasimhan, M., & Bilmes, J. (2007). Local search for balanced submodular clusterings. In *Proceedings of the 20th international joint conference on artifical intelligence* (pp. 981–986).

Nemhauser, G. L., Wolsey, L. A., & Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions. I. *Mathematical Programming*, *14*(1), 265–294.

ORourke, J., & Supowit, K. (1983). Some NP-hard polygon decomposition problems. *IEEE Transactions on Information Theory*, *29*(2), 181–190.

Pimenta, L. C. A., Schwager, M., Lindsey, Q., Kumar, V., Rus, D., Mesquita, R. C., et al. (2009). *Simultaneous coverage and tracking (scat) of moving targets with robot networks*, Algorithmic foundation of robotics VIII (pp. 85–99). Berlin: Springer.

Renzaglia, A., Doitsidis, L., Martinelli, A., Kosmatopoulos, E. B. (2010). Cognitive-based adaptive control for cooperative multi-robot coverage. In *IEEE/RSJ international intelligent robots and systems* (pp. 3314–3320).

Renzaglia, A., Doitsidis, L., Martinelli, A., Kosmatopoulos, E. B. (2011). Adaptive-based distributed cooperative multi-robot coverage. In *American control conference* (pp. 468–473).

Renzaglia, A., Doitsidis, L., Martinelli, A., & Kosmatopoulos, E. B. (2012). Multi-robot three dimensional coverage of unknown areas. *The International Journal of Robotics Research*, *31*(6), 738–752.

Richards, A., Boyle, P. (2010). Combining planning and learning for autonomous vehicle navigation. In *AIAA guidance, navigation and control conference*.

Schwager, M., Julian, B. J., & Rus, D. (2009). Optimal coverage for multiple hovering robots with downward facing cameras. In *IEEE international conference on robotics and automation* (pp. 3515–3522).

Singh, A., Krause, A., Guestrin, C., Kaiser, W., & Batalin, M. (2007). Efficient planning of informative paths for multiple robots. In *Proceedings of the 20th international joint conference on artificial intelligence* (pp. 2204–2211).

Singh, A., Krause, A., & Kaiser, W. (2009). Nonmyopic adaptive informative path planning for multiple robots. In *International joint conference on artificial intelligence* (pp. 1843–1850).

Stobbe, P., & Krause, A. (2012). Learning Fourier sparse set functions. In *Proceedings of the fifteenth international conference on artificial intelligence and statistics* (pp. 1125–1133).

Stone, L. D. (1975). The theory of optimal search. Operations Research Society of America.

Tokekar, P., & Isler, V. (2014). Polygon guarding with orientation. In *IEEE international conference on robotics and automation* (pp. 1014–1019).

Trummel, K. E., & Weisinger, J. R. (1986). The complexity of the optimal searcher path problem. *Operations Research*, *34*(2), 324–327.

Tseng, K. S., & Mettler, B. (2015). Near-optimal probabilistic search via submodularity and sparse regression. *Autonomous Robots*,. doi:10.1007/s10514-015-9521-5.

Vig, L., & Adams, J. A. (2007). Coalition formation: From software agents to robots. *Journal of Intelligent and Robotic Systems*, *1*, 85–118.

Kuo-Shih Tseng received the B.S. degree in Mechanical Engineering from Chung Yuan Christian University, Taiwan, in 2002, the M.S. degree in Bio-Industrial Mechatronics Engineering from National Taiwan University, Taiwan, in 2004. From 2004 to 2009, he was an associated researcher with the Intelligent Robotics Technology Division, Mechanical and System Research Laboratories, Industrial Technology Research Institute (ITRI), Hsinchu, Taiwan. From 2009 to 2011, he was a researcher with KYH Co., Ltd., Taiwan. He received his second M.S. degree in the Computer Science and Engineering from University of Minnesota, in 2013. He is currently a Ph.D. candidate in the Department of Computer Science and Engineering at University of Minnesota. His current research interests include robotic search, human search behavior analysis and reinforcement learning.

Bérénice Mettler received the Diploma in mechanical engineering from ETH, Zurich, Switzerland, in 1996 and her Ph.D. degree from Carnegie Mellon University, Pittsburgh, PA, USA, in 2001. Until 2004, she was a Postdoctoral Researcher and then a Research Scientist at MITs Laboratory for Information and Decision Systems. In 2006, she joined the Department of Aerospace Engineering and Mechanics, the University of Minnesota, Minneapolis, MN, USA, where she is currently an Associate Professor. Her current research interests include guidance of agile vehicles in challenging tasks and environments. She runs the Interactive Guidance and Control Lab (IGCL). Some of her current research projects include benchmarking autonomous systems guidance performance and modeling human guidance skills, with an emphasis on modeling techniques to capture the control and planning mechanisms used to achieve versatile and adaptive performance. Her interdisciplinary approach combines methods from dynamics and controls with machine learning, along with current perspectives and knowledge from human factors and cognitive sciences. She has published articles in the areas of modelling, control design, and autonomous guidance of autonomous aerial vehicles, including IdentificationModeling and Characteristics of Small-Scale Rotorcraft (Kluwer, 2003).