

Near-optimal probabilistic search via submodularity and sparse regression

Kuo-Shih Tseng¹ · Bérénice Mettler²

Received: 21 May 2015 / Accepted: 7 November 2015 / Published online: 29 December 2015
© Springer Science+Business Media New York 2015

Abstract The goal of search is to maximize the probability of target detection while covering most of the environment in minimum time. Existing approaches only consider one of these objectives at a time and most optimal search problems are NP-hard. In this research, a novel approach for search problems is proposed that considers three objectives: (1) coverage using the fewest sensors; (2) probabilistic search with the maximal probability of detection rate (PDR); and (3) minimum-time trajectory planning. Since two of three objective functions are submodular, the search problem is reformulated to take advantage of this property. The proposed sparse cognitive-based adaptive optimization and PDR algorithms are within $(1 - 1/e)$ of the optimum with high probability. Experiments show that the proposed approach is able to search for targets faster than the existing approaches.

Keywords Probabilistic search · Coverage problem · Receding horizon control · Submodularity · Overlapping group LASSO

1 Introduction

Optimal search is one of the oldest topics in operation research. Today, search is receiving new attention due in part to applications in robotic search. In robotic search, a robot(s) has to autonomously search for a target(s) in an environment. It involves the interaction between the robot(s), target(s), and environment (Chung et al. 2011). How the problem of optimal search is solved depends on assumptions about these factors. There are three major approaches: searching in a graph-like environment, searching in a polygonal environment, and searching in a probabilistic environment (Bayesian search).

The graph-based model is based on the assumptions that the sensing and motions of robots are discrete, making it possible to model the problem using a graph (see Fig. 1a). The goal of this problem is to find optimal actions for catching the “enemy” (Nowakowski and Winkler 1983). The advantage of this model is that graph theory can be used. The disadvantage is that the motion and sensing models can easily be overly simplistic.

The polygonal model considers continuous motion and sensing (Aggarwal 1984). The assumptions of the polygonal model are that the environment can be described through polygons, the range of sensors is infinite and the field of view is 360° (See Fig. 1b). The goal is either to cover the whole environment using the fewest sensors or maximize the coverage using a fixed number of sensors. The advantage of the polygonal model is that it is possible to achieve theoretical bounds and approximate solutions based on the concept of set cover. However, it is generally impossible to obtain polygonal maps and perfect sensors in the real world.

Bayesian search focuses on how to estimate the target’s motion and position based on probability theory (Stone 1975). The assumptions of Bayesian search are that the search

✉ 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

area can be divided into finite cells/graphs and that each cell represents individual probability of detection (PD) (See Fig. 1c). The goal is to determine the optimal path to find the lost or moving target(s) according to the probability distribution function (PDF) (Richardson and Stone 1971; Stone et al. 2014). The three steps of Bayesian search for a target are as follows: The first is to compute prior PDF according to motion information (e.g. flight dynamics and drift data). The second is to compute posterior PDF according to sensor information (e.g. pinger and side-looking sonar). The third is to move to the highest probability cell, scan this area and update the posterior PDF as the prior PDF. The three steps are repeated until the target is found. Two optimization objectives are as follows: to maximize the PD or minimize the expected time to detection (ETTD). The advantages of Bayesian search are that the imperfect sensing/detection and target's motion can be modeled as a probability distribution and the probability of each cell is updated according to real-time sensing data. However, most Bayesian search applications are for outdoor environments. In such cases, the cell is larger than the sensors scanned area, so the coverage problem can be ignored. Interestingly, even though the three approaches rely on assumptions and simplifications, all three problems are still NP-hard.

To apply Bayesian search to indoor environments, a relaxed approach is to assume that the map is divided into several rooms and when the robot is in one of the rooms, its sensors cover the whole room (see Fig. 1a). Then, the robot computes the optimal search path based on Bayesian and graph theories (Lau et al. 2008, 2006). However, this type of discretization often precludes considering real sensing coverage and robot motion constraints (see Fig. 1d).

The goal of this research is to enable a robot to search for a target in confined environments considering sensing coverage, sensing uncertainty, and robot motion constraints. The original Bayesian search has been proposed for outdoor environments, so the major assumption is that the sensing area only covers the cell the robot occupies and the robot only moves to neighbor cells (see Fig. 1c). However, such an approach does not consider three critical factors for indoor environments. The first is coverage. The sensors cover multiple cells and the sensing range could be occluded by obstacles such that coverage becomes a complex set cover problem. The second is sensing overlap. The sensing areas at different positions could overlap. As Fig. 1d shows, the sensing areas of red and green positions are overlapping. The third is motion cost. If there are obstacles, the best decision is not always to move into the cell with the highest probability. As Fig. 1d shows, there is an obstacle between the robot and the purple position. Even if the purple position has the highest probability, its motion cost is also highest. These factors make planning in indoor environments more difficult than in most outdoor environments.

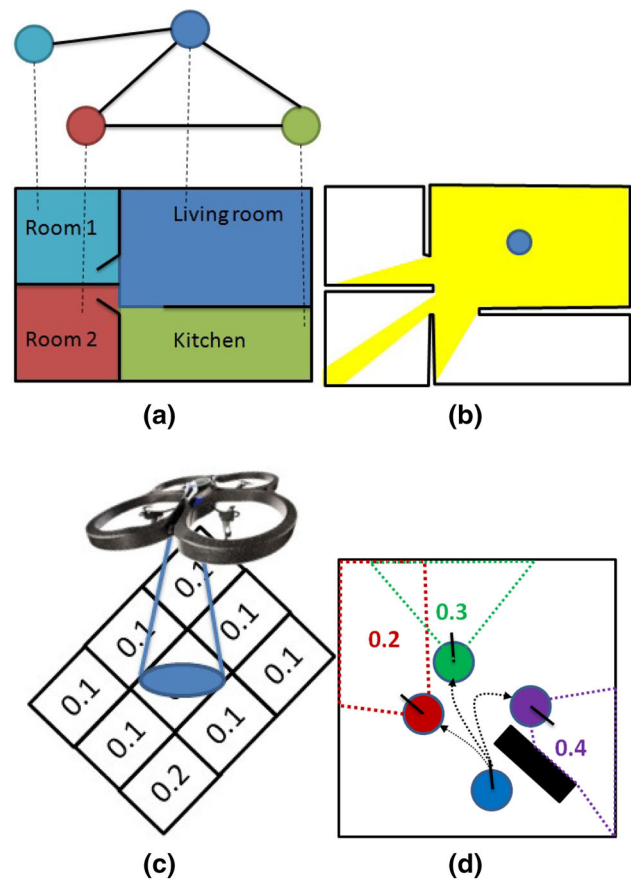


Fig. 1 Illustration of four search problem approaches. **a** The color circles and areas represent the robot positions and corresponding sensing areas respectively. **b** The blue circle and yellow areas represent the robot position and the covered area respectively. **c** The blue ellipse, cells, and decimal numbers represent the scanned cell, search space, corresponding probability respectively. **d** The blue circle, other circles, black rectangle and arrows represent the robot position, subgoal positions, obstacles and paths (Color figure online)

To overcome these issues, the proposed approach consists of dividing the search problem into three subproblems: a coverage problem with the fewest sensors; probabilistic search with maximal probability of detection rate (PDR); and minimum-time trajectory planning. Although the three subproblems are NP-hard, each of them can be solved efficiently using proper approximation algorithms.

The main contributions of this paper are as follows: First, the proposed approach can search for the target considering coverage, probability, and motion cost. Second, PDR can handle the overlapping sensing areas and consider motion cost. Third, the sparse cognitive-based adaptive optimization (SCAO) algorithm solves maximum coverage and minimum number of k-sensors simultaneously via sparse regression techniques. Fourth, due to the submodularity of the objective function, the PDR algorithm gives the $(1 - 1/e)$ lower bound guarantee with high probability while

SCAO gives the $(1 - 1/e)$ lower bound guarantee, which is near-optimal (Nemhauser et al. 1978). Unless $P = NP$, no polynomial time algorithm is strictly better than this approach (Feige 1998). To the best of our knowledge, the PDR and its objective function submodularity have not been investigated, and the SCAO is the first algorithm to apply sparse regression to the coverage problem.

The rest of the paper is organized as follows. Section 2 describes the related work. Section 3 gives the overview of the proposed approach. Section 4 details the benchmark algorithms and proposed algorithms for the three subproblems. In Sect. 5, experiments demonstrate the performance, optimality and sensitivity analysis of the proposed algorithms. Section 6 discusses the limitation and potential for extensions of the proposed algorithms. Finally, Sect. 7 provides conclusions.

2 Related work

This section describes related work associated to the three subproblems and search algorithms for confined environments.

2.1 Coverage problem

For surveillance applications, a typical goal is to monitor an entire room using the fewest sensors, or cover the most space using a fixed number of sensors. Both problems are NP-hard. 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 of robots is collected for learning. Based on the learned coverage function, the robots move to the locations such that the coverage is maximal. Hence, an approximate solution can be found.

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. However, those theoretical bounds are based on oversimplified 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 shows the result, which are different from previous work. First, the robots learn the objective function from real-time sensing data. Second, even if the environment is non-convex, the CAO algorithm still works. More details on the CAO algorithm are provided in Sect. 4.1.

2.2 Probabilistic search

Bayesian search is divided into two components. The first, perception, is to compute the probability distribution of the target. Recursive Bayesian estimation enables robot(s) to estimate the target using real-time motion and measurement data (Bourgault et al. 2003). The second, decision-making, is to compute the optimal actions toward the target. However, finding the optimal decisions in a probabilistic environment however is NP-hard.

To further consider larger horizons, the problem was reformulated as partially observable Markov decision processes (POMDPs) (Eagle 1984; Kadane and Simon 1977), which is NP-hard. Hence, maximizing the cumulative PD is proposed. Two assumptions had to be considered in the PD model. First, the propagation model assumes that the sensors are without false detections and without detection of the target at each sampling. Second, that the sensing coverage at each position is non-overlapping and independent of another. Based on these assumptions, the PD 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 alleviate the computational issues. The principle of BNB is to generate the possible branches and prune the branches if the cost values exceed some pre-specified 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, suboptimal solutions of search paths can be obtained by choosing a suitable bound or horizon with an affordable computation.

2.3 Minimum-time trajectory planning

Minimum-time trajectory planning problem is NP-hard, which means that if the robot tries to arrive at the goal as soon as possible, computational complexity grows exponentially

with the order of the searcher's dynamics and the size of the map. Hence, trade-off between reducing the computational load and increasing the performance is the key challenge to solving this problem. There are two major approaches to compute approximate paths for a robot(s). The Rapidly-Exploring Random Tree (RRT) is one of the approaches in robotic planning (LaValle and Kuffner 1999). It works by randomly generating nodes from the free space once connections obstacles-free edges between two nodes, they are connected. The robot's approximate path from start to goal is found by graph search.

Another approach is receding horizon (RH) optimization with a cost-to-go (CTG) function (Bellingham et al. 2002) in the aerospace field. The CTG function is an approximation of global cost to the robot in the global environment. The receding horizon is able to give accurate solutions of constrained optimization within finite horizons. If the goal is not approachable within a finite horizon, the robot will search an approachable active waypoint (AWP) from the CTG function (Mettler et al. 2010). The AWP minimizes the composed cost (CTGC) considering the cost to reach AWP (cost-to-come, CTC) and the cost from the AWP to the global goal (CTG). So, the original optimization problem is relaxed to many small problems. In other words, to reduce the computational load, the robot only considers optimal solutions within the finite horizons. More detail on guidance techniques can be found in Goerzen et al. (2010) and Bellingham et al. (2002). Hence, both RRT and RH are able to find suboptimal path.

2.4 Search in confined environments

Since indoor environments are cluttered, searching for a target efficiently is a challenge. Under certain assumptions, there are several algorithms that are able to search for the target in indoor environments. In Hollinger et al. (2010), if the map is preprocessed as a graphical model, the proposed algorithm is able to capture the target within guaranteed time. In Gerkey et al. (2006), if the map is preprocessed as a polygonal map, the proposed algorithm is able to search the target using limited field of view sensors. In Lau et al. (2008, 2006), if the map is preprocessed as a graphical model, the proposed algorithm is able to search for the target with a tighter bound than previous probabilistic search. However, those algorithms do not consider sensing coverage, sensing uncertainty, and motion constraints simultaneously. The overview of the proposed algorithms are in next section.

3 Approach overview

In the past 10 years, the robotics community has been paying more attention to the concept of informative path planning (IPP) (Singh et al. 2007) and submodular func-

tions (Nemhauser et al. 1978; Krause et al. 2008; Singh et al. 2009). The goal of IPP is to find an optimal path which maximizes the amount of sensing information (Binney and Sukhatme 2002). 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 the performance of the approximated solutions?

The first problem is related to “adaptive sampling and feature selection” (Binney and Sukhatme 2002). Some novel techniques of machine learning are adopted, such as Gaussian Processes (GP). The second problem is related to “path planning and probabilistic search” (Binney and Sukhatme 2002). 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). For example, robots are able to find paths that enable a robot to collect maximal information using IPP (Singh et al. 2007). A robot is able to find paths which maximize variance reduction of Gaussian processes (Binney and Sukhatme 2002). A robot is also able to find paths which get maximal information or minimal distortion from wireless stations (Hollinger and Sukhatme 2013; Hollinger et al. 2013). The successful examples of IPP problems and submodularity inspire a reformulation of search problems.

The probabilistic search is reformulated as follows: In a known grid map (m), the initial position of the robot (X^{R_0}) is known. The robot acquires the real-time motion data (u_k) and measurement data (z_k) from sensors, where k is time index. The goal is to find the optimal velocity ($v_{1:k}$) and angular velocity ($\omega_{1:k}$) of the robot such that the actions satisfy the following objectives:

Objective 1 Coverage with the fewest subgoals: Given the sensor specifications, the robot finds the minimal number of subgoals to cover the most space in the environment (see Fig. 2a).

Objective 2 Probabilistic search: According to the given subgoal positions, the robot finds the optimal path over the graph to maximize probability of detection rate (PDR) (see Fig. 2b).

Objective 3 Minimum-time trajectory planning: Given the current robot and subgoal positions, the robot finds the optimal velocity commands to minimize travel time under motion constraints (see Fig. 2c).

The majority of existing approaches for search only deal with O1 or O2 (Lau et al. 2008, 2006; Gerkey et al. 2006; Hollinger et al. 2010). Since the three objectives are NP-hard, each subproblem has to be solved using an approximate algorithm.

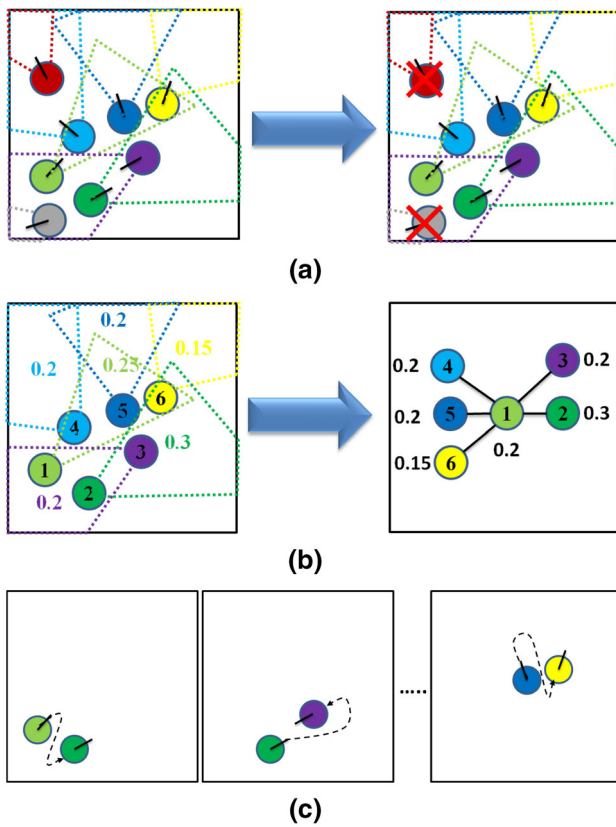


Fig. 2 Illustration of the three subproblems. **a** The color circles, dash lines and red cross marks represent the subgoal positions, corresponding covered area and discarded subgoals respectively. **b** The decimal numbers and black lines represent the probability in covered area and path between nodes respectively. **c** The dash lines represent the trajectory from one subgoal to the other (Color figure online)

3.1 Coverage with the fewest subgoals

Coverage with the fewest subgoals can be seen as coverage with the fewest sensors/robots, which are moving in the environment to maximize the total coverage. Given the number of robots (M), the robots' positions ($X^{R_{1:M}}$), the sensors' range (z_R), the field of view (z_θ), real-time sensing data (z^i) and a grid map (m), the goal is to find the number (M_{eff}) and position (X_{eff}) of the fewest robots to cover the environment, where $i = 1, \dots, M$. If M_{eff} is known, then X_{eff} is solved by the CAO algorithm. However, computing M_{eff} is another NP-hard problem. Hence, the proposed sparse CAO (SCAO) is to find M_{eff} and X_{eff} simultaneously. The basic concept of SCAO is to drop enough robots ($X^{R_{1:M}}$) in the map. Some of the robots cover very small areas. As Fig. 2a shows, if those robots are discarded, M_{eff} and X_{eff} are found. The formulation of the objective function is the following:

$$X_{eff} = \arg \max_{X^{R_{1:M}} \in \mathcal{E}} F_C(X^{R_{1:M}}) \tag{1}$$

where \mathcal{E} is the space of possible allocations for a robot and F_C is the coverage function.

3.2 Probabilistic search

Given the measurement data (z_k), the probability distribution of measurement model $P(z_k | X_k^t, m)$, the motion model probability distribution $P(X_k^t | X_{k-1}^t)$ and the cells of the map ($m = \{m_1, \dots, m_N\}$), the goal is to find the target distribution conditioned on sensing information and map $P(X_k^t | z_{1:k}, u_{1:k}, m)$ and the optimal search path P_{path}^* (see Fig. 2b). The first step is to compute the probability in the covered area (Pd_i) with Bayes filter (see ‘‘Bayes filter’’ section in Appendix). where $i = 1, \dots, M_{eff}$. The second step is to maximize cumulative PDR using BNB to determine the P_{path} over the graphs as in the following:

$$P_{path}^* = \arg \max_{P_{path}} F_{PDR}(P_{path}) \tag{2}$$

s.t. $F_{PDR}(P_{path}) \geq B$

where F_{PDR} is the PDR, P_{path} is the path sequence and B is the assigned bound.

3.3 Minimum-time trajectory planning

The subgoal positions ($X_g^l, l = 1, \dots, M_{eff}$) and visiting order of subgoals (P_{path}^*) are given by SCAO and PDR respectively. The current position of the robot ($X^{R,k}$), the maximal velocity (v_{max}) and angular velocity (ω_{max}) are given. The goal is to find the velocity commands (v, ω) of a mobile robot at each time step such that the robot visits the subgoal sequentially within minimum-time (see Fig. 2c). Minimum-time trajectory planning is modeled using mixed-integer nonlinear programming (MINLP). The formulation is based on receding horizon control (RH) (Bellingham et al. 2002).

$$V^* = \arg \min_{V \in \Lambda} F_m(b, t) = \sum_{i=1} b_i t_i \tag{3}$$

$$F_M = \sum_{l=1}^{M_{eff}} F_{m,l} \tag{4}$$

where V is the velocity vector, Λ is the space of the robot's motion and velocity constraints, and b is a binary decision vector, t_i is the time at which the goal is reached and F_m is the function of t and b .

3.4 The proposed search approach

The complete flowchart of the proposed probabilistic search is shown in Fig. 3. In the offline stage, the SCAO algorithm

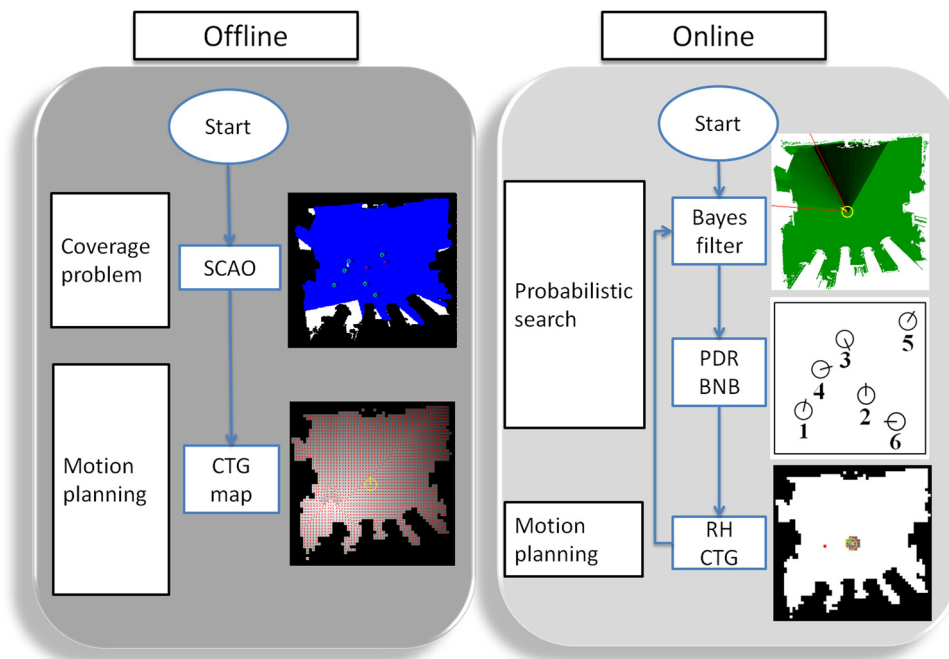


Fig. 3 The flowchart for the offline and online processes. *SCAO* The green points, blue area, and white area represent the subgoal positions, covered area, and uncovered area respectively. *CTG map* The yellow circle and red lines represent the robot position and optimal direction at each cell respectively. The cost in each cell is represented as grey scale. The brighter and darker color represent lower cost and higher cost respectively. The subgoal is at the lower left corner so the lower left corner is brighter and the upper right corner is darker. *Bayes filter*

The green area and black area represent higher and lower probability in the logarithmic form. The red line is the field of view (FOV) of the robot. *PDR BNB* The circles and number represent the subgoal nodes and index. *RH CTG* The red point, green point, red line, and yellow circle represent the subgoal, AWP, directions of CTGC cells, and robot position respectively. The grey area represents the CTGC cost within fixed distance (Color figure online)

determines the subgoals to cover the environment. The cost-to-go (CTG) maps store the cost from any position to each subgoal. After the offline state, the robot knows where the subgoals are and what the cost is from any position to each subgoal.

In the online stage, the Bayes filter computes the target distribution according to real-time sensing data and detection (see “Bayes filter” section in Appendix). According to the target distribution, the PDR BNB algorithm plans the optimal path (P_{path}^*). Finally, the RH CTG algorithm computes the robot velocity commands ($V^* = [v_{k:k+H}, w_{k:k+H}]$) to guide the robot to the subgoals. In summary, the robot senses the environment using Bayes filter, computes the optimal visiting order of subgoals using PDR BNB and moves toward the subgoal using RH CTG. Once the target probability density of a certainty area is higher than the threshold, the robot terminates the search.

4 Proposed algorithms

Since some benchmark approaches (e.g. CAO, PD and CTG) are highly related to proposed algorithm, they are reviewed

in Sects. 4.1.1, 4.2.1, and 4.3.1. The proposed algorithms are described in Sects. 4.1.2, 4.2.2 and 4.3.2. The optimality of SCAO, PD and PDR is proved in Sect. 4.4. Finally, Sect. 4.5 summarizes the relationship between the proof and corresponding algorithm.

4.1 Coverage problem

CAO is proposed to solve the coverage problem via learning (Renzaglia et al. 2012, 2011). However, the number of robots needs to be predefined. To solve maximal coverage problem with the fewest subgoals, sparse cognitive-based adaptive optimization (SCAO) is proposed in Sect. 4.1.2.

4.1.1 Cognitive-based adaptive optimization (CAO)

The CAO algorithm is divided into three stages: sensing, learning, and decision-making. Before the first stage, the initial robots positions (X^{R_0}), the range of sensors (z_R), and the field of view (z_θ) are given.

In the sensing stage, the robots acquire the sensing data (z_k). Then, the coverage rate of robots (y_k) in the map is computed (see “Computing coverage” section in Appendix).

In the learning stage, the robots learn the coverage function using the robots position data and coverage data. The coverage function is formulated as follows:

$$\hat{C}(X^R, W) = w_1 \cdot 1 + w_2 \phi_2(X^R) + \dots + w_L \phi_L(X^R) = \sum_{i=1}^L \phi_i(X^R) w_i \tag{5}$$

where \hat{C} is the estimated coverage, W is the weighting vector, X^R is the robot positions, $\phi(\cdot)$ is an approximator, and w is the weighting. The robots need to keep at least L data points in memory for learning. The coverage function is written as a matrix form in Eq. 6. It is reformulated as a linear regression problem (Eq. 7).

$$\begin{bmatrix} c_{k-p} \\ c_{k-p+1} \\ \vdots \\ c_k \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^L \phi_i(X_{k-p}^R) w_i + \epsilon_k \\ \sum_{i=1}^L \phi_i(X_{k-p+1}^R) w_i + \epsilon_{k+1} \\ \vdots \\ \sum_{i=1}^L \phi_i(X_k^R) w_i + \epsilon_{k+N} \end{bmatrix} \tag{6}$$

$$C_{(P,1)} = X_{(P,L)} W_{(L,1)} + \epsilon, \quad \epsilon \sim N(0, \Sigma) \tag{7}$$

where P is the number of data points for learning ($P \geq L$), C is a batch of coverage data, X is a batch of approximator data, ϵ is the Gaussian noise, and Σ is the covariance matrix of noise. The goal is to find the least square error between true coverage and estimated coverage function (Eq. 8). The computational complexity of the least square algorithm is $O(P^3)$.

$$\hat{W} = \arg \min_W \left\{ \|C - XW\|_2^2 \right\} \tag{8}$$

In the decision-making stage, the robots generate N samples of positions (Eq. 9). According to the learned weighting (Eq. 8), the robots choose the sample which generates maximal coverage for their next actions (Eq. 10).

$$x_k^{i,j} = x_k^{(i)} + \alpha_k \zeta_k^{i,j}, \quad i \in \{1, \dots, M\}, j \in \{1, \dots, N\} \tag{9}$$

$$x_k^{opt} = \arg \max_{x_k^{i,j} \in \mathcal{E}} \{X' \hat{W}\} \tag{10}$$

where $x_k^{i,j}$ is the j th sampling position of i -th robot at time k . $x_k^{(i)}$ is the i -th robot's original position at time k . $\zeta_k^{i,j}$ is a zero-mean, unit-variance random vector with dimensions equal to the dimensions of x^i and α_k is a positive real sequence, which decays depending on the number of iterations. \mathcal{E} is the possible space for robot allocations, X' is the X matrix of samples and x_k^{opt} is the optimal placement of samples. The robots keep repeating three stages until the terminal condition. The final robots positions are assign as subgoals positions. According

to the empirical experiments, the robots will cover as much of the space as possible based on the learned coverage function after hundreds of iterations. The major computation occurs at the learning stage so the computational complexity of CAO is $O(P^3)$ if Eq. 8 is solved by the least square method.

A key issue in linear regression is how to choose suitable approximators. Since coverage function is related to each dimension datum (e.g. x, y, θ), the approximator should include whole dimensional data. Polynomial approximators, radial basis functions, and kernel-based approximators are suitable for the coverage application. In [Renzaglia et al. \(2012\)](#), the authors proposed that random 3rd-order polynomial approximators are sufficient for 2D and 3D coverage problems. The approach to choosing polynomial approximators is the following:

- (1) Choose $L = L_2 + L_3 + 1; (L_2 = L_3)$
- (2) The first term of the approximator is constant;
- (3) Select randomly L_2 terms of ϕ to be any 2nd-order terms of the form $x_a^i \cdot x_b^j$ with $a, b \in \{1, \dots, \dim(x^i)\}, i, j \in \{1, \dots, M\}$ randomly-selected positive integers;
- (4) Select randomly L_3 terms of ϕ to be any 3rd-order terms of the form $x_a^i \cdot x_b^j \cdot x_c^k$ with $a, b, c \in \{1, \dots, \dim(x^i)\}$ and $i, j, k \in \{1, \dots, M\}$ randomly-selected positive integers.

The disadvantage of polynomial approximators is that the number of approximators is increasing dependent on the number of robots. According to [Renzaglia et al. \(2012\)](#), if $N \geq 2 \cdot M \cdot \dim(x^i)$ and $L = 2 \cdot M \cdot \dim(x^i) + 1$, the CAO algorithm is able to find the maximal coverage. However, since the polynomial terms are randomly generated, the chosen terms may not be very correlated to coverage function. Hence, this approach could lead to overfitting. Moreover, the number of needs subgoals are unknown for search applications.

4.1.2 Sparse cognitive-based adaptive optimization (SCAO)

Although the CAO algorithm is able to find the solution of maximal coverage, assigning the minimal number of robots is NP-hard. To solve this problem, the concept of sparse regression is adopted ([Tibshirani 1996](#)). Due to increasing “Big Data” applications, scientists need to analyze millions or billions of data points via machine learning techniques. If the data has the sparse property, it can be solved efficiently. L1-norm (LASSO) is proposed to solve such problems ([Tibshirani 1996](#)) (Eq. 11). After sparse regression, the lower weighting values are discarded.

$$\hat{W}_{L1} = \arg \min_W \left\{ \|Y - XW\|_2^2 + \lambda_1 \|W\|_1 \right\} \tag{11}$$

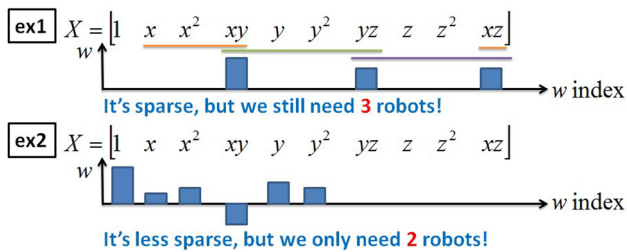


Fig. 4 Illustration of two different sparse weighting vectors. The x axis and y axis represent the weighting index and the magnitude of weighting vector respectively. The orange, green, and purple lines represent the terms including x , y , and z variables respectively (Color figure online)

To apply sparse regression to this coverage problem, let us assume that enough robots are dropped and that the coverage areas of some robots are small. If these robots are discarded, M_{eff} can be found through sparse regression. Although LASSO generates sparse weighting vectors, it does not directly generate a smaller number of the robots. Figure 4 shows that the weighting vector length is 10. x , y , z are the positions of robot₁, robot₂, robot₃. Each weighting value is given with respect to an individual polynomial term. For example, w_4 is associated with xy . There are two examples of sparse regression. The sparsity of the two examples is 30 and 60 % respectively. The weighting vector of the first example is sparser than the second one. However, if checking the related polynomial terms, three robots are needed in the first example but only two robots are needed in the second example because of the interacting terms of x , y , z . This example shows that a sparser weighting vector does not guarantee fewer robots. The sparse regression therefore should be more structured.

Such problems, where some of variables are highly correlated, are called group LASSO (Yuan and Lin 2006). To solve this problem, an elastic net has been proposed for a more structured sparse regression (Zou and Hastie 2005). The concept of an elastic net is the tradeoff between L1-norm and L2-norm (Eq. 12). L1-norm generates a sparse weighting vector while L2-norm generates a group weighting vector. Different values of λ_1 and λ_2 will generate different sparse regressions. However, it is difficult to control the sparsity of groups. To consider more flexible structure, a pairwise elastic net (PEN) is proposed (Lorbert et al. 2010) (Eq. 13). The P matrix is the penalty matrix, which satisfies the condition of positive-semidefinite (PSD). If $P = I$, the solution of PEN is equal to L2-norm. If $P = 11^T$, the solution of PEN is equal to L1-norm. Moreover, the P matrix is able to handle the overlapping group LASSO. In such cases, some groups are overlapping (see Fig. 4). In Lorbert and Ramadge (2013), the P matrix is designed based on Laplacian matrix. PEN is efficiently solved by coordinate descent. The computational complexity of coordinate descent is $O(mP^2)$, where m is the number of iterations.

$$\hat{W}_{EN} = \arg \min_W \left\{ \|Y - XW\|_2^2 + \lambda_1 \|W\|_1 + \lambda_2 \|W\|_2^2 \right\} \tag{12}$$

$$\hat{W}_{PEN} = \arg \min_W \left\{ \|Y - XW\|_2^2 + \lambda_{PEN} |W|^T P |W| \right\} \tag{13}$$

To solve the maximal coverage problem with fewer robots, PEN and CAO are adopted. The proposed algorithm is called Sparse Cognitive-based Adaptive Optimization (SCAO). The major difference between CAO and SCAO is the learning stage. In CAO, the learning stage is formulated as a least square problem. In SCAO, the learning stage is formulated as an overlapping group LASSO problem. The SCAO algorithm is given in Algorithm 1. The advantages of the SCAO algorithm are as follows:

- *Fewer robots required* The SCAO algorithm requires fewer robots to cover the environment than the CAO approach.
- *No overfitting* The SCAO algorithm avoids overfitting because it automatically chooses meaningful polynomial terms. Hence, the predicted coverage is more accurate.
- *Sparsity* Due to the sparsity of the weighting vector, the memorized data points for regression are fewer than for CAO. For example, if L is 61 and the number of non-zero terms is 30, the robot only needs 30 data points for learning. However, for the CAO algorithm, the robot needs at least 61 data points for learning.

```

Input:  $X_M, z_R, z_\theta, z_k^{1:M}$  and map;
Output:  $X_{eff}$ ;
Initialization: Generate random polynomial terms of
approximators  $\phi_{1:L}(\cdot)$ ;
while iteration <  $I_{th}$  do
    % Sensing stage:
    Compute the coverage  $y_k$  of  $X_k$  from  $z_k$ 
    % Learning stage:
    Save the sensing and coverage data as a batch

     $\hat{W} = \arg \min_W \{ \|Y - XW\|_2^2 + \lambda_{PEN} |W|^T P |W| \}$ 

    % Decision-making stage:

     $x_k^{i,j} = x_k^{(i)} + \alpha_k \zeta_k^{i,j}$ 

     $x_k^{opt} = \arg \max_{x_k^{i,j} \in \mathcal{E}} \{ X' \hat{W} \}$ 
end
    
```

Algorithm 1: The SCAO algorithm.

4.2 Probabilistic search

Probabilistic search is divided into two components, perception and decision-making. Using Bayes filter solves

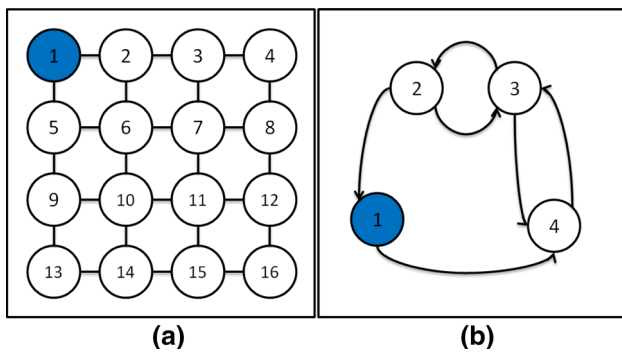


Fig. 5 Grid and graphical models. The white circles and blue circles represent the search space and the cell the robot is located at respectively. The black lines and arrows represent the bidirectional and directional path constraints respectively. For example, the robot in **a** can move to node 2 or node 5. The robot in **b** can only move to node 4 (Color figure online)

perception (see “Bayes filter” section in Appendix). Maximizing cumulative PD solves decision-making. However, such an approach has some constraints and needs additional tuning. To improve it, maximizing cumulative PDR is proposed in Sect. 4.2.2.

4.2.1 Maximal cumulative PD using branch and bound

To compute the optimal search path with uncertainty is NP-hard. Hence, a simplified propagation model is proposed (Washburn 1998). The assumptions used in the propagation model are as follows. First, there is no detection of the target along the path. Second, the robot only moves to the nearest cells or nodes (see Fig. 5). Third, the robot only senses in the node where it is located and each sensing coverage does not overlap with the others. Based on these assumptions, the goal is to maximize the cumulative probability of detection (PD) within T time steps, which is an NP-Complete problem (Trummel and Weisinger 1986). To find an approximation solution, BNB approach is adopted (Washburn 1998).

$$P_{path} = \arg \max_{\pi} F_P(\pi) = \sum_{t=1}^T P(\pi(t), t) \cdot g(\pi(t), t)$$

s.t.

$$\pi(t + 1) \in \Psi(\pi(t))$$

$$F_P(\pi(t)) \geq B(t), t \in 0, \dots, T - 1 \tag{14}$$

where π is the search policy, g is the glimpse function, $0 \leq g \leq 1$, Ψ is the path constraints from a node to another node, and B is the threshold of BNB. For example, $\pi = \{s_1, s_2, s_1\}$ means the robot visits subgoal 1, 2, and 1 sequentially. $P(s_1) = 0.6$, $P(s_2) = 0.4$ and $g = 0.8$. $F_P = 0.6 \times 0.8 + 0.4 \times 0.8 + 0.12 \times 0.8 = 0.896$.

```

Input:  $P_{1:M}, g(\cdot, t), T$ ;
Output: optimal search path  $P_{path}$ ;
while not arrived at  $T$ -th step do
  expand branches of the search path  $\pi(t' + 1)$ 
  compute each cumulative  $PDR(\pi(t' + 1))$ 
  if  $PDR(\pi(t')) \leq B(t')$  then
    prune this path;
  else
     $\pi_{t'} = \arg \max_{\pi} PDR(\pi)$ ;
  end
end
 $P_{path} = \pi(T)$ 
    
```

Algorithm 2: The algorithm for optimal search path.

4.2.2 Maximal cumulative PDR using branch and bound

The assumptions of maximal cumulative PD are not realistic for searching in indoor environments. First, the sensing coverage at different positions could be overlapping (see Fig. 1d). Second, the robot is able to freely move to any nodes (see Fig. 1d). Third, the PD approach does not consider the robot motion constraints. Although the non-uniform travel times are considered in Lau et al. (2008) and Lau et al. (2006) (see Fig. 5b), the user needs to predefine the travel times and those travel times do not consider robot motion constraints. Although the approach in Waharte et al. (2010) can deal with overlapping and partial observations, this approach cannot give a theoretical guarantee of search performance.

For searching in minimum time, the PDR is introduced. Once SCAO generates subgoals positions, the robot knows where it can visit. Once CTG maps of subgoals are computed, the robot knows the travel times from any position to the subgoals. Hence, it is able to plan an optimal path considering time-rate of PD. The formulation of the maximal cumulative PDR problem is as follows:

$$P_{path} = \arg \max_{\pi} F_{PDR}(\pi) = \sum_{t=1}^T \frac{P(\pi(t), t)}{\tau(\pi(t))} \cdot g(\pi(t), t)$$

s.t.

$$F_{PDR}(\pi(t)) \geq B(t) \tag{15}$$

where τ is the travel time from a node to another node. The algorithm is given in Algorithm 2.

To select a suitable B is a key issue. When B is close to 0, BNB approach behaves as an exhaustive search. When B is close to 1, BNB approach prunes too many possible solutions. The proposed algorithm is shown in Algorithm 3. The computational complexity is $O(M^2)$ and the generated B gives $(1 - 1/e)$ -approximation in Algorithm 2. When $\beta = 1$, the BNB approach behaves as a greedy approach. The details of the proof are derived in the Sect. 4.4.

The advantages of the PDR algorithm are as follows:

```

Input:  $P_{1:M}, g(\cdot, t), \beta, T$ ;
Output:  $B$ ;
while not arrived at  $T$ -th step do
   $B_{greedy,t} = \max_t PDR(t)$ ;
end
 $B = \beta \cdot B_{greedy}$ , where  $\beta \leq 1$ 
    
```

Algorithm 3: The algorithm for finding B .

- *Sensing overlap* The PD approach assumes that the sensing areas at each node are not overlapping with each other. The PDR can handle the sensing overlapping cases.
- *No path constraints* The PD has path constraints. The robot only moves to neighbor cells or predefined graphical nodes. The PDR does not have any path constraint. The robot can pick up any node as its next subgoal.
- *Consideration of motion cost* The PD approach does not consider the motion cost from one node to the other node. Since the CTG map gives the cost from any position to any subgoal, the cost can be used for the PDR approach.
- *(1 – 1/e)-approximation* Based on the submodularity, the threshold of BNB is computed easily using Algorithm 3. It gives (1 – 1/e)-approximation with high probability.

4.3 Minimum-time trajectory planning

Once the PDR algorithm gives the current subgoal, the robot plans a minimum-time trajectory as follows. First, as Fig. 6 shows, the subgoal is too far for the robot to reach within H horizons. Hence, the robot finds the AWP through the CTGC map, which is the minimum cost cell within distance D . Second, the robot computes the velocity commands to reach AWP within error $(\epsilon_p, \epsilon_\theta)$ using MINLP. Third, the robot repeats first and second steps until reaching the subgoal. The following sections describe how to compute the CTG map and receding horizon control.

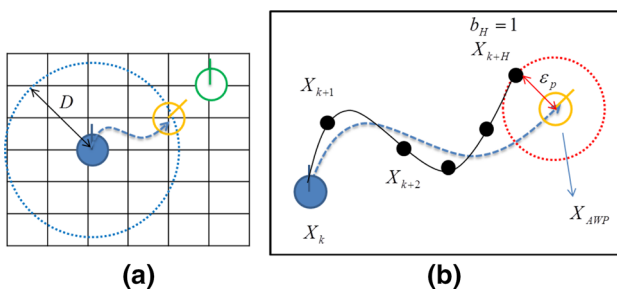


Fig. 6 Illustration of minimum-time trajectory planning. **a** The blue, orange, green circles, blue curve and blue dash circle represent the robot, AWP, subgoal positions, optimal path and distance within D respectively. **b** The blue, black curves, black point and red circle represent optimal, true path, the robot position at h horizon and error bound respectively (Color figure online)

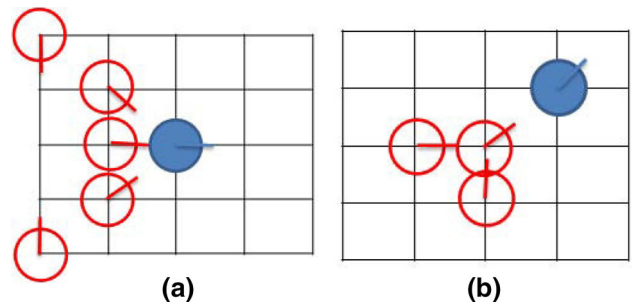


Fig. 7 Motion primitives of different heading angles. The red circles and blue solid circles represent potential actions and the robot end position respectively (Color figure online)

4.3.1 Cost-to-go (CTG) and cost-to-come (CTC) functions

The CTG and CTC functions are computed based on simple motion primitives (see Fig. 7). The heading resolution of the robot is 45° . The 0° motion primitive is similar to 90° , 180° , and 270° . The 45° motion primitive is similar to 135° , 225° , and 315° . The CTG function consists of the cost from any cell to the subgoal. It is efficiently computed by the Dijkstra’s algorithm. The CTC function consists of the cost from the current robot pose to the reachable cells within distance D . The AWP is computed by minimizing the composite cost of CTG and CTC. The algorithm is given in Algorithm 4. More details can be found in Mettler et al. (2010).

4.3.2 Receding horizon control for a mobile robot

Since the motion model of miniature UAVs is linear, receding horizon optimization is formulated as a MILP in Mettler et al. (2010). However, the motion model of a differential wheeled robot is nonlinear (Eq. 22). Hence, the receding horizon trajectory optimization for a mobile robot is formulated as a mixed integer nonlinear programming (MINLP). The position and heading of the active waypoint $(X_{AWP} = [x_{AWP}, y_{AWP}, \theta_{AWP}])$ and the current robot position (X_k) are given. The goal is to find the velocity (v_i) and angular velocity (w_i) that results in the mobile robot’s minimum-time trajectory within a finite horizon (H) , where $i = 1, \dots, H$.

$$\min \sum_{i=1}^H \{M_c(H + 1 - i)(1 - b_i)\} \tag{16}$$

$$s.t. (x_i - x_{AWP})^2 + (y_i - y_{AWP})^2 \leq M_c(1 - b_i) + \epsilon_p \tag{17}$$

$$|\theta_i - \theta_{AWP}| \leq M_c(1 - b_i) + \epsilon_\theta \tag{18}$$

$$\sum_{i=1}^H b_i = 1, \tag{19}$$

$$|v| \leq v_{max}, |w| \leq w_{max} \tag{20}$$

$$|\Delta v| \leq a_{max}, |\Delta w| \leq \alpha_{max} \quad (21)$$

where M_c is a large number, H is the number of samples in the horizon, b_i is a binary decision variable, $[x_i, y_i, \theta_i]$ are the position and heading of the robot at i -th horizon, ϵ_p is the error between the desired and current position, ϵ_θ is the error between the desired and current heading, and v_{max} , w_{max} , a_{max} and α_{max} are the maximum of velocity, angular velocity, acceleration and angular acceleration respectively.

Reaching the AWP in minimum time is enforced by penalizing the cost at the beginning of the planning horizon ($i = 1$) more than at the end of the horizon ($i = H$). Once the robot reaches the AWP, the corresponding binary variable b_i becomes one. Equations (17) and (18) are the constraints of position error and heading error respectively. Equations (19), (20) and (21) are the binary, velocity and acceleration constraints respectively. The unknown variables are $[v_i, w_i]$ but the constraints of (17) and (18) are functions of $[x_i, y_i, \theta_i]$. Hence, a motion model to convert $[v_i, w_i]$ to $[x_i, y_i, \theta_i]$ is needed. The state space model of a mobile robot is the following:

$$\begin{aligned} x_{k+1} &= x_k + v_k \cos(\theta_k) dt \\ y_{k+1} &= y_k + v_k \sin(\theta_k) dt \\ \theta_{k+1} &= \theta_k + w_k dt \end{aligned} \quad (22)$$

where $[x_k, y_k, \theta_k]$ are the position and heading of the robot at time k , $[v_k, w_k]$ are the velocity and angular velocity commands at time k and dt is the sampling time. The position of the robot can be computed through the state space model (22).

```

Input: map, subgoals, velocity constraints;
Output: velocity commands of the robot at each time step;
Initialization: compute CTG maps of all subgoals;
while not arrived at the final subgoal do
  if subgoal is within sensing range then
    Receding-Horizon(robot-position, subgoal);
    read next CTG map;
  else
    compute active way point( $X_{AWP}$ ) from CTG and CTC
    map within distance  $D$ ;
    Receding-Horizon( $X_k, X_{AWP}$ );
  end
end

```

Algorithm 4: The algorithm for minimum-time trajectory planning.

4.4 Optimality of SCAO, PD and PDR

Although maximum coverage, probabilistic search, and minimum-time trajectory planning problems are NP-hard,

near-optimal solutions can be computed efficiently if the objective functions are submodular. The proof steps are as follows: First, the Lemma 1 introduces submodularity. Lemma 2 proves whether three objective functions are submodular or not. Second, Theorems 1 and 2 introduce that the greedy approach generates $(1 - 1/e)$ -approximation for submodular functions. Theorem 3 proves $(1 - 1/e)$ -approximation guarantee of SCAO. Third, Theorems 5 and 6 prove $(1 - 1/e)$ -approximation guarantee of PD and PDR.

Lemma 1 (Submodularity) *Given a finite set $S = \{1, 2, \dots, n\}$, a submodular function is a set function: $F : 2^S \rightarrow \mathbb{R}_+$ 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.*

Lemma 2 (Submodularity of three objective functions)

- Coverage function is submodular.
- Probabilistic search is submodular if there are no false detections and no detections along the search path.
- Minimum-time trajectory planning is not submodular.

Proof Assume F_C, F_{PS}, F_M are the objective functions of coverage, probabilistic search and minimum-time trajectory planning. S is the ground set of subgoals. S_A, S_B are the two sets of subgoals and $S_A \subseteq S_B \subseteq S$. s is an additional subgoal. \square

(a) Coverage function

Since coverage is one of the submodular functions, $F_C(S_A \cup s) - F_C(S_A) \geq F_C(S_B \cup s) - F_C(S_B)$ holds. The details of this proof are found in Krause et al. (2008).

(b) Probabilistic search

Since probabilistic search involves detection uncertainty, the target's PDF is updated by the outcome of detections. $\phi = \{1, -1\}$ denotes whether the target is at the detected area or not and $Z^D = \{1, -1\}$ denotes whether the robot detected the target or not. For example, $P(\phi = 1 | Z^D = -1)$ means the conditional probability that the robot does not detect the target while the target is at the detected area. Hence, the outcome of detections is $Z^D = 1$ or $Z^D = -1$. The goal is to find the optimal policy $\pi_{opt,k}$ for picking k subgoals so that the cumulative PD is maximal (see Fig. 8a). Since adding a visited subgoal could result in a detection or no detection of a target, the updated probability consists of two cases:

$$f(\pi(t), t) = \begin{cases} P(\pi(t), t) \cdot g, & \text{if no detection} \\ P(\pi(t), t) \cdot g', & \text{if there is a detection} \end{cases}$$

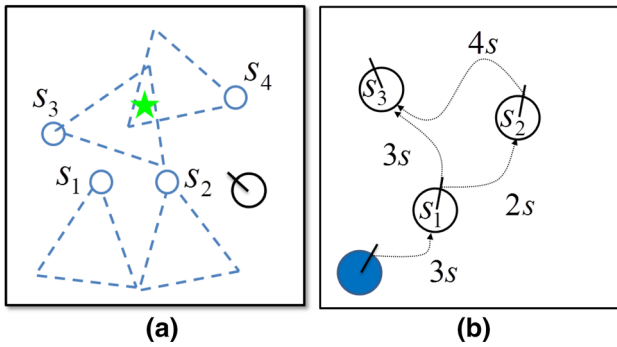


Fig. 8 Illustration of the two problems. **a** Blue circles, triangles, Green star and the black circle represent the subgoal positions, corresponding detection area, target position and the robot position respectively. **b** The blue solid circle, circles, arrows and numbers represent the robot position, subgoals, paths, travel times respectively (Color figure online)

where g, g' are glimpse functions with a detection and no detection respectively. The objective function is in the following:

$$\pi_{opt,k} = \arg \max_{\pi, |\pi| \leq k} F_{PS}(\pi) = \arg \max_{\pi, |\pi| \leq k} \sum_{t=1}^T f(\pi(t), t),$$

where T is the number of the selected set, $0 \leq g \leq 1$ and $g' < 0$. More details about the range of g and g' can be found in “Probability of detection (PD) model” section in Appendix. The probability could be increasing or decreasing depending on the detection. Hence, $F_{PS}(S_A \cup s) - F_{PS}(S_A) \geq F_{PS}(S_B \cup s) - F_{PS}(S_B)$ doesn't hold.

If there is no detection ($Z^D = -1$) of a target along the path, the problem is simplified as PD model (see “Probability of detection (PD) model” section in Appendix), where $f(\pi(t), t) = P(\pi(t), t) \cdot g$ and $F_P = \sum_{t=1}^T f(\pi(t), t)$. Under such assumptions, the probability of scanning the area is always decreasing. The diminishing returns of S_A and S_B are as follows:

$$\begin{aligned} &F_P(S_A \cup s) - F_P(S_A) \\ &= \sum_{t=1}^{|A|} P(\pi(t), t) \cdot g + P(\pi(|A| + 1), |A| + 1) \cdot g \\ &\quad - \sum_{t=1}^{|A|} P(\pi(t), t) \cdot g \\ &F_P(S_B \cup s) - F_P(S_B) \\ &= \sum_{t=1}^{|B|} P(\pi(t), t) \cdot g + P(\pi(|B| + 1), |B| + 1) \cdot g \\ &\quad - \sum_{t=1}^{|B|} P(\pi(t), t) \cdot g \end{aligned}$$

$\because S_A \subseteq S_B \therefore |A| \leq |B|$. To compare the magnitude of $P(\pi(|A| + 1), |A| + 1)$ and $P(\pi(|B| + 1), |B| + 1)$, there are two cases: First, if the scanned area in set B and set s do not overlap, then $P(\pi(|A| + 1), |A| + 1)$ is equal to $P(\pi(|B| + 1), |B| + 1)$ (e.g. $S_A = \{S_1\}$, $S_B = \{S_1, S_2\}$, and $s = \{S_3\}$ shown in Fig. 8a). Second, if the observed areas in set B and S overlap, the probability of overlapping area is discounted (e.g. $S_A = \{S_1\}$, $S_B = \{S_1, S_4\}$ and $s = \{S_3\}$ shown in Fig. 8a). So, $P(\pi(|A| + 1), |A| + 1) > P(\pi(|B| + 1), |B| + 1)$. Based on the two cases, $P(\pi(|A| + 1), |A| + 1) \geq P(\pi(|B| + 1), |B| + 1)$. Therefore, $F_P(S_A \cup s) - F_P(S_A) \geq F_P(S_B \cup s) - F_P(S_B)$ holds.

(c) *Minimum-time trajectory planning*

If the robot adds a subgoal s in trajectory planning problem, the total travel times will not decrease. As Fig. 8b shows, $A = \{s_1\}$, $B = \{s_1, s_2\}$ $s = \{s_3\}$.

$$\begin{aligned} &F_M(S_A \cup s) - F_M(S_A) = (3 + 3) - 3 = 3 \\ &F_M(S_B \cup s) - F_M(S_B) = (3 + 2 + 4) - (3 + 2) = 4 \\ &F_M(S_A \cup s) - F_M(S_A) < F_M(S_B \cup s) - F_M(S_B) \end{aligned}$$

Therefore, $F_M(S_A \cup s) - F_M(S_A) \geq F_M(S_B \cup s) - F_M(S_B)$ doesn't hold.

Theorem 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 $OPT = \max_{A \subseteq S, |A|=k} F(A)$. The lower bound of the greedy algorithm is $F(A_G) \geq (1 - (\frac{k-1}{k})^k) OPT \geq (1 - 1/e) OPT$.*

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

Theorem 3 (Submodularity of SCAO) *Given a finite set $S = \{1, 2, \dots, n\}$ and $S_{s,i} \subseteq S$ is sampling set at i -th iteration, let $OPT_i = \max_{A \subseteq S_{s,i}, |A|=k} F_C(A)$. $F_C(A_G) \geq (1 - 1/e) OPT_i$ holds.*

Proof Assume F_C and \hat{F}_C are the objective and approximation functions of the coverage problem. After certain iterations, $\hat{F}_C \approx F_C$. The approximated function behaves as a submodular function. According to Theorem 2, if the robot picks up the sampling set greedily at i th iteration, the solution has $(1 - 1/e) OPT_i$ lower bound.¹ Theorem 3 proves the optimality of Algorithm 1.

¹ The coverage function of SCAO is an approximation function instead of a true function. Hence, the near-optimal solution of SCAO is based on the \hat{F}_C and sampling set.

The major difference between greedy approach for the K -max coverage and SCAO is how to choose a set sequentially. Greedy approach chooses a set from the whole set S sequentially. SCAO picks up the set from a small sampling set $S_{s,i}$ and keep changing the sampling set such that the coverage is maximal. The way to change the sampling set is according to online learned approximation function \hat{F}_C . The difference between CAO and SCAO is that sparse regression will reduce the predicted error such that the \hat{F}_C is closer to F_C . \square

Theorem 4 (Optimality of maximizing cumulative PD using BNB) F_P is the objective function for cumulative probability of detection. Given suitable bound B , the following equation holds: $F_P(\pi_{BNB}) \geq F_P(\pi_g) \geq (1 - 1/e)F_P(\pi_{opt,k})$.

Proof Lemma 2 proves F_P is submodular. The next step is to prove $F_P(\pi_{BNB})$ is greater than or equal to $F_P(\pi_g)$. According to Theorem 4, F_P is submodular, so the greedy policy gives $(1 - 1/e)$ -approximation. Assume the greedy approach chooses set $\pi_g = \{s_1, s_2, \dots, s_k\}$. The cumulative bound can be computed and the greedy bound is $B_{greedy} = \{b_1, b_2, \dots, b_k\}$. Assign $B^* = B_{greedy}$ as the bound of cumulative PD problem. The BNB algorithm behaves as a greedy algorithm. $F_P(\pi_g) \geq (1 - 1/e)F_P(\pi_{opt})$ holds. To get better performance, users can assign a bound B , where $b_i < b_i^*$. Then $F_P(\pi_{BNB}) \geq F_P(\pi_g)$ holds. This conclusion doesn't violate Theorem 2 since BNB is not a polynomial time approach. \square

Theorem 4 gives an interesting result. PD model with BNB (Eq. 14) is adopted for most of the probabilistic search applications. However, to decide the parameters of the bound is a trade-off between optimality and computation. Lower bound values generate more branches while higher bound values prune more branches. Furthermore, no research indicates the performance guarantee of a bound. Theorem 4 shows that if the path constraints (Ψ) are removed, the greedy approach generates near-optimal solution for the PD model. Choosing any bounds lower than the greedy approach will have an equal or better performance than the greedy approach.

To illustrate the difference between the original PD model and Theorem 4, as Fig. 5a shows, there are 4 directions for each cell. If the robot is at cell 1 and tries to find 10 steps such that PD is maximal using BNB, the worst case computation could be 4^{10} . If the robot adopts Theorem 4 and the path constraints are skipped, the computation is 10×16 . Moreover, this approach gives 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.

Motion cost is also an important factor for search. Theorem 4 assumes that choosing any subgoal has uniform cost. However, it is oversimplified. For example, s_4 is nearer than

s_3 for the robot in Fig. 8a. Choosing s_4 as the next subgoal has lower motion cost. The following Theorems 5 and 6 prove that the PDR algorithm generates $(1 - 1/e)$ -approximation with high probability.

Theorem 5 (Submodularity with Non-uniform Item Costs (Khuller et al. 1999)) Given a finite set $S = \{1, 2, \dots, n\}$, the item cost of element $s \in S$ is $\tau(s) \in \mathbb{R}_+$ and a submodular function $F : 2^S \rightarrow \mathbb{R}_+$. The objective function is $\max_{S_g \subseteq S} f(S_g)$ s.t. $\tau(S_g) \leq \tau_{th}$, where $\tau(S_g)$ denotes the motion cost to visit each subgoal of S_g . The inequality $f(S_g) \leq (1 - e^{-\frac{\tau(S_g)}{\tau_{th}}})OPT$ holds. Based on this inequality, the greedy approach gives $(1 - 1/\sqrt{e})OPT$ guarantee. The greedy approach with partial enumeration gives $(1 - 1/e)OPT$ guarantee.

If all item costs are uniform (e.g., $\tau(s) = 1$ and $B = k$), the lower bound of $f(S_g)$ is $(1 - 1/e)OPT$, which is the case for Lemma 1. Theorem 5 shows when the $\tau(S_g)/\tau_{th}$ is close to 1, the lower bound of $f(S_g)$ is close to $(1 - 1/e)OPT$ [Khuller et al. (1999)]. Since the $\tau(s)$ depends on the given set S_A or S_B , a probability bound is necessary. For example, as Fig. 5a 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. $\tau(s|A)$ and $\tau(s|B)$ are $T_{1,3}$ and $T_{2,3}$ respectively. Hence, $\tau(s|A) \neq \tau(s|B)$. To prove the submodularity with motion costs, a probabilistic bound is introduced as follows:

Lemma 3 (Chebyshev's inequality) Let $X : \Omega \rightarrow \mathbb{R}$ be any random variable, and let $\epsilon > 0$ be any positive real number. Then, $P(|X - \mathbb{E}(X)| \geq \epsilon) \leq \frac{Var(X)}{\epsilon^2}$.

Theorem 6 (Optimality of Maximizing Cumulative PDR with Motion Costs using BNB) F_{PDR} is the objective function for cumulative probability of detection rate. Given a bound τ_{th} , the following equation holds. $F_{PDR}(\pi_{BNB}) \geq (1 - 1/e)F_{PDR}(\pi_{opt,k})$ with $P[|r - \mathbb{E}(r)| \geq \epsilon] \leq \frac{Var(r)}{\epsilon^2}$ where $r = \frac{\tau(S_g)}{\tau_{th}}$.

Proof Since $F_{PDR}(s) \leq 1$ and $\tau(s) > 1$, the $\tau(s)$ dominates the greedy selections. The greedy approach chooses the lower item cost with high probability so the $\tau(S_g)/\tau_{th}$ is very close to 1. In other words, the variance of $\tau(S_g)/\tau_{th}$ is very small. Applying Lemma 3, the probability that $F_{PDR}(S_g) \leq (1 - 1/e)OPT$ is bounded by $P[|r - \mathbb{E}(r)| \geq \epsilon] \leq \frac{Var(r)}{\epsilon^2}$. For example, if $\mathbb{E}(r) = 0.95$, $Var(r) = 0.05^2$ and $\epsilon = 0.2$, the probability that the near-optimal guarantee does not hold is lower than 6.2%. According to Theorem 4 and 5, maximizing cumulative PDR using BNB (Algorithm 2) gives $(1 - 1/e)$ -approximation with high probability.

4.5 Summary of the proofs

First, Theorem 3 proves that SCAO (Algorithm 1) generates near-optimal solutions based on the sampling. Second, Theorem 4 proves that if the path constraints are removed, the greedy solutions of PD model (Eq. 14) give near-optimal guarantee. Finally, Theorem 6 proves that PDR approach (Algorithm 2) gives near-optimal guarantee with high probability.

5 Experiments

To verify the proposed algorithms, the coverage performance and search time are evaluated through a series of coverage and search experiments. In Sect. 5.2, the coverage experiments focus on coverage performance of the greedy, CAO and SCAO algorithms. The results of greedy and SCAO approach are adopted as the subgoals of the following search experiments. Section 5.3 compares the search time using six combinations of algorithms including greedy, nearest neighbor (NN), PD, and PDR. In Sect. 5.4, the optimality and time complexity of SCAO and PDR are investigated. Finally, in Sect. 5.5, a sensitivity analysis is presented to illustrate the trade-off between coverage and motion cost.

5.1 Experimental setup

The experiments are conducted at the Interactive Guidance and Control Lab (IGCL). The dimensions of the search task space are $5 \times 5m^2$. The mobile robot is a Pioneer 3DX equipped with a Microsoft Kinect. The robot gets the RGBD image and odometer data from the Kinect and P3DX encoders respectively. The detector is trained using Adaboost and Haar-like features to detect the target (a soccer ball) (Viola and Jones 2001). The conditional probability table of detection for a Bayes filter is based on Adaboost training data. The grid map of IGCL is built using FastSLAM (Montemerlo et al. 2013). The update rate of the sensors and velocity commands are at 4Hz and 1Hz respectively.

The setup of receding horizon control is as follows: The resolution of CTG map and heading resolution are $10 \times 10cm^2$ and 45° respectively. Once the robot knows the current subgoal, it will load the CTG map of this subgoal. The robot computes the 3D CTGC map within D distance and chooses the minimal cost as AWP. The robot computes the optimal velocity commands within H horizons using MINLP. The MINLP problem is formulated in AMPL and is solved by a BNB algorithm provided by the BONMIN software (Bonami et al. 2008). Once the robot reaches the current subgoal, the robot loads the CTG map of next subgoal and moves toward it.

The setup of the coverage experiments is as follows: For the greedy approach, the map is discretized using $10 \times 10 cm^2$ cells with a heading resolution of 45° . The number of available cells is around 6000. The greedy algorithm chooses the maximal coverage sequentially. For the (S)CAO approaches, the robots position is continuous. At the initial stage, there are 8 robots, with initial positions at $[0^\circ, 0^\circ, 90^\circ]$. After 500 iterations, the final robots positions are assigned as the subgoals² of (S)CAO.

The setup of the search experiments is as follows: The robot searches for the target using different approaches. The termination condition is when the total probability of the H-area exceeds 90% and then the robot decides the target is in the H-area. The H-area is defined as the $20 \times 20 cm^2$ area around the highest probability cell. If the target is in the H-area, it is a positive decision. If the target is not in the H-area or the search time is over 240 s, it is a negative decision. $E[TTD+]$ denotes the expected time till positive decisions (Chung and Burdick 2012).

The human machine interface (HMI) of the search system is as follows: Figure 9d and e show the RGB and depth images used for object detection and measuring the target 3D position respectively. Once the target is detected, the HMI indicates its position. Figure 9c shows that if there is a detection, the probability of that area will increase. Otherwise, the probability of the scanned area will decrease. Figure 9b shows that there are six subgoals and corresponding probability. The optimal path is 3–1–4. Figure 9a shows that the robot moves toward the subgoal 3 using receding horizon control.

The selections of experimental parameters (see Table 1) are as follows:

SCAO: If the number of subgoals (M) is equal to 8, the CAO has high probability to cover around 90%. Since the robot works in 2D space, the dimension ($dim(x)$) is 3. The number of polynomial approximators (L) and samples (N) for the least square method are $2 \times M \times 3 + 1 = 49$ and $2 \times M \times 3 = 48$ respectively. The penalty parameters λ_1 , λ_2 and λ_{PEN} are tuned according to the trade-off between accuracy and sparsity/grouping. For example, if λ_1 is higher, the weights are sparser.

PDR: β decides the number of expanded branches. $\beta = 1$ means the greedy approach is adopted. Glimpse function value (g) is decided by $P(\phi = 1 | z^D = -1)$, which is available after Adaboost training. The depths of branches (T) is

² The relationship between the coverage of subgoals and maximal coverage for searching is the following. The coverage of subgoals is based on discrete subgoal positions. However, the robot will move continuously to K subgoals. Hence, the computed coverage is the lower bound of the real coverage.

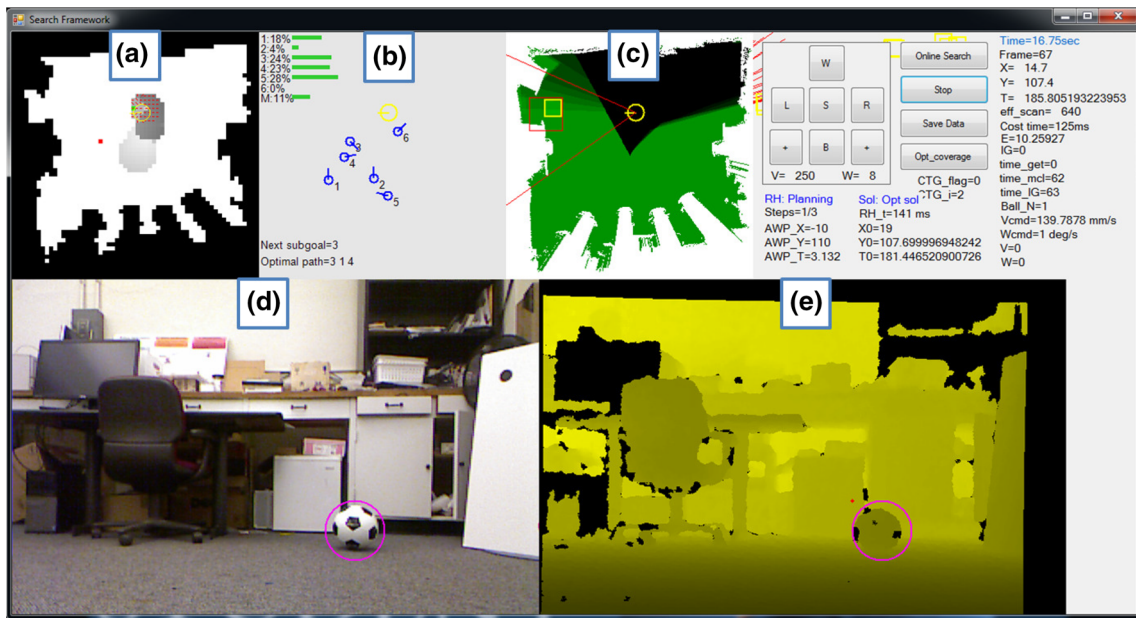


Fig. 9 Snapshot of the human machine interface. **a** The RH CTG window shows receding horizon control and CTGC map. The red point and green point represent the subgoal position and AWP point respectively. The red lines and yellow circle represent the optimal direction of CTGC map and the robot position respectively. The grey scale represents the CTGC cost within 50 cm of the robot position. **b** The PDR BNB window shows the target distribution for each subgoal and optimal path. The green bars, blue circles, and yellow circle represent the probability of individual subgoal, subgoal positions, and the robot position respectively. **c** The Bayes filter window shows the target distribution in the

grid map. The bright green and black represent the higher and lower probability distribution respectively. The red rectangle, yellow rectangle, red line, and yellow circle represent the H-area, detected area, field of view, and robot position respectively. **d** The RGB window shows the real-time image. Pink circle represents the detection of the target. **e** The depth window shows the real-time depth image. The 3D (x - y - z axis) data is projected to the 2D (x - y axis) data. Once the target is detected, the 3D (x - y - z axis) target position is projected to the 2D (x - y axis) position. The 2D map and detection data is used to update the target distribution using the Bayes respectively (Color figure online)

3, which is computational efficiency for the PD(R) approach. z_R and z_θ are the parameters of Kinect sensors.

MINLP: The horizons of MINLP (H) are decided by the run time and uprate of the system. When $H = 8$, the run time of MINLP is around 120 ms. The distance D for computing CTGC is 50 cm, which can be reached with $H = 8$. The velocity and acceleration parameters (V_{max} , ω_{max} , a_{max} and α_{max}) are determined according to requirements. Higher speeds lead to image blur, which decreases the detection rate of Adaboost. Hence, those parameters are set based on the trade-off between the robot speed and detection rate of Adaboost.

5.2 Coverage experiments

The goal of the experiment is to compare the performance of the greedy approach, CAO with least square (LS), CAO with L2-norm, CAO with L1-norm, and CAO with PEN (SCAO). Figures 10 and 11 show the results obtained after 500 iterations.

Table 1 Parameters used in the experiments

M	N	L	λ_{PEN}	λ_1	λ_2
8	48	49	0.65	0.001	0.001
β	g	T	z_R	z_θ	
1	0.8	3	4 (m)	57°	
H	D	V_{max}	ω_{max}	a_{max}	α_{max}
8	50 cm	20 (cm/s)	22 (°/s)	5 (cm/s ²)	11 (°/s ²)

Figure 10d shows that the number of robots with LS and L2 is always 8. In contrast, the number of subgoals with L1 and PEN is around 4–8. Due to the structured group, PEN requires fewer robots than L1. In addition, Fig. 11a–e show that the maximal coverage for all approaches is over 85%. In terms of the algorithms, Fig. 11a–c show that the maximal coverage of LS, L2, L1 is around 90% and the number of robots is 8. While Fig. 11d–e show that the maximal coverage rate of PEN is 85.6%, the number of subgoals is only 6; the 2nd and 8th robots are discarded. Note also that the greedy approach only needs 6 subgoals to cover over 90%.

There are three advantages of the SCAO algorithm. First, the weighting vector is sparse. Sparsity is the number of

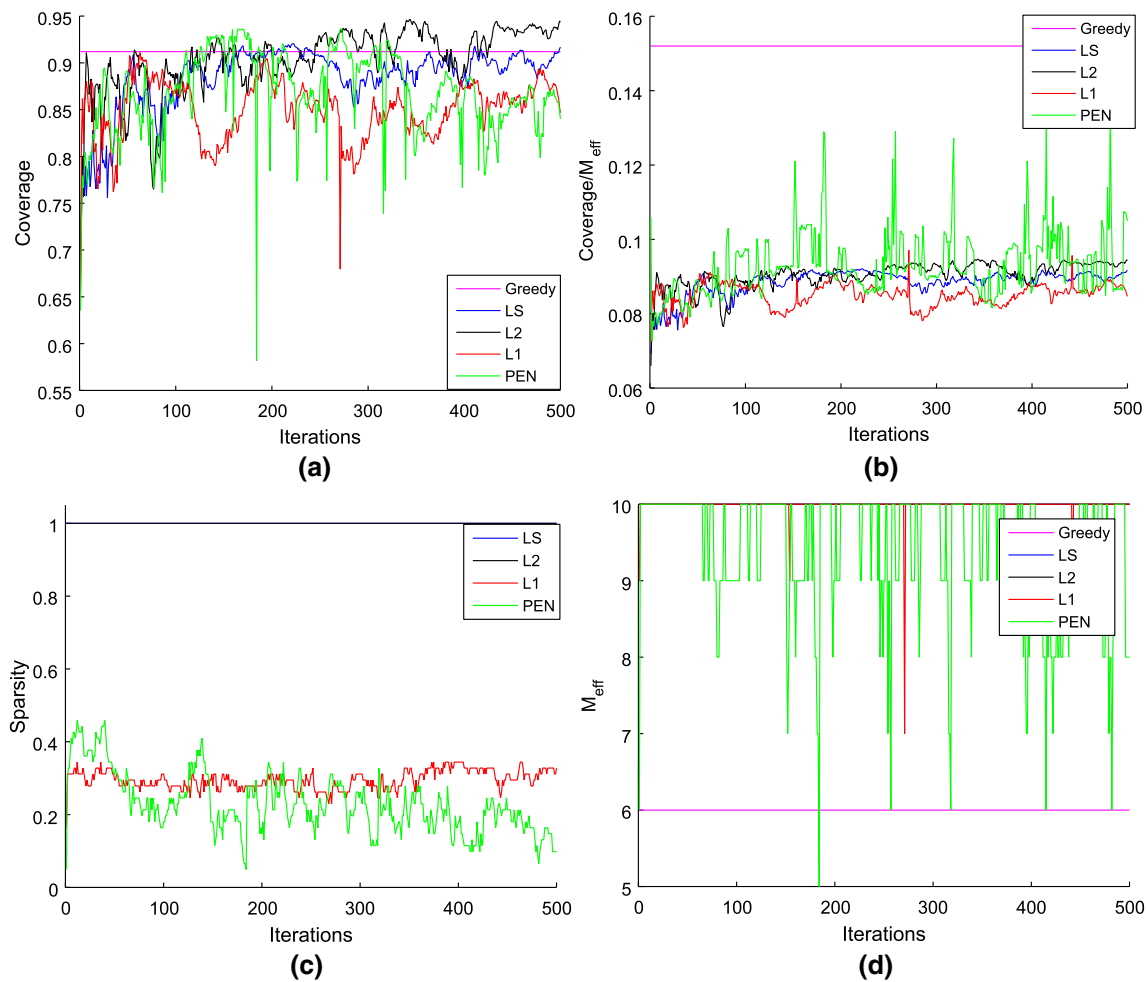


Fig. 10 Results of greedy, LS, L2, L1, PEN algorithms over the iterations (Color figure online)

non-zero weights divided by the number of total weights. Figure 10c shows that the sparsity obtained with LS and L2 is 1 because neither approach makes the weighting sparse. In contrast, L1 and PEN discard around 60% of the weights.

Second, fewer subgoals are required. This result is illustrated in Fig. 11. SCAO only requires 6 subgoals while CAO needs 8 subgoals. Although L1 approach also generates sparse weights, the sparse constraints are not structured. Hence, even if the weights of L1 approach are sparse, L1 approach cannot generate fewer subgoals.

Third, the regression employing regularization prevents overfitting. The predicted error is defined as $c_k - \hat{c}_k$. It provides an index to judge whether the learned coverage function is accurate. Table 2 shows that the predicted error of LS approach is bigger than that of the other approaches. Because the polynomial terms are randomly generated, some terms are not very correlated; therefore, the LS approach leads to overfitting. In contrast, the other approaches employ regularization; therefore, the uncorrelated terms are discarded or

reduced. As a result, the L2, L1 and PEN predicted errors are less than that of LS approach.

The Greedy and SCAO generate fewer subgoals and achieve maximal coverage (see Fig. 10b). Therefore, the following section adopts the subgoal generated by these two approaches to evaluate the search time.

5.3 Search experiments

The goal of the search experiment is to evaluate the performance in terms of coverage, probabilistic search, and trajectory planning. The algorithms are shown in Table 3. To compare the $E[TTD+]$ for these six approaches, the target is placed in the unoccupied cells of the map. For each approach, the robot searches for the target 50 times (Table 4). The 6 subgoals are determined by the Greedy or SCAO algorithms as shown in Fig. 11d and e. The $E[TTD+]$ attained by the six approaches are summarized in Table 5. The experimental results are as follows.

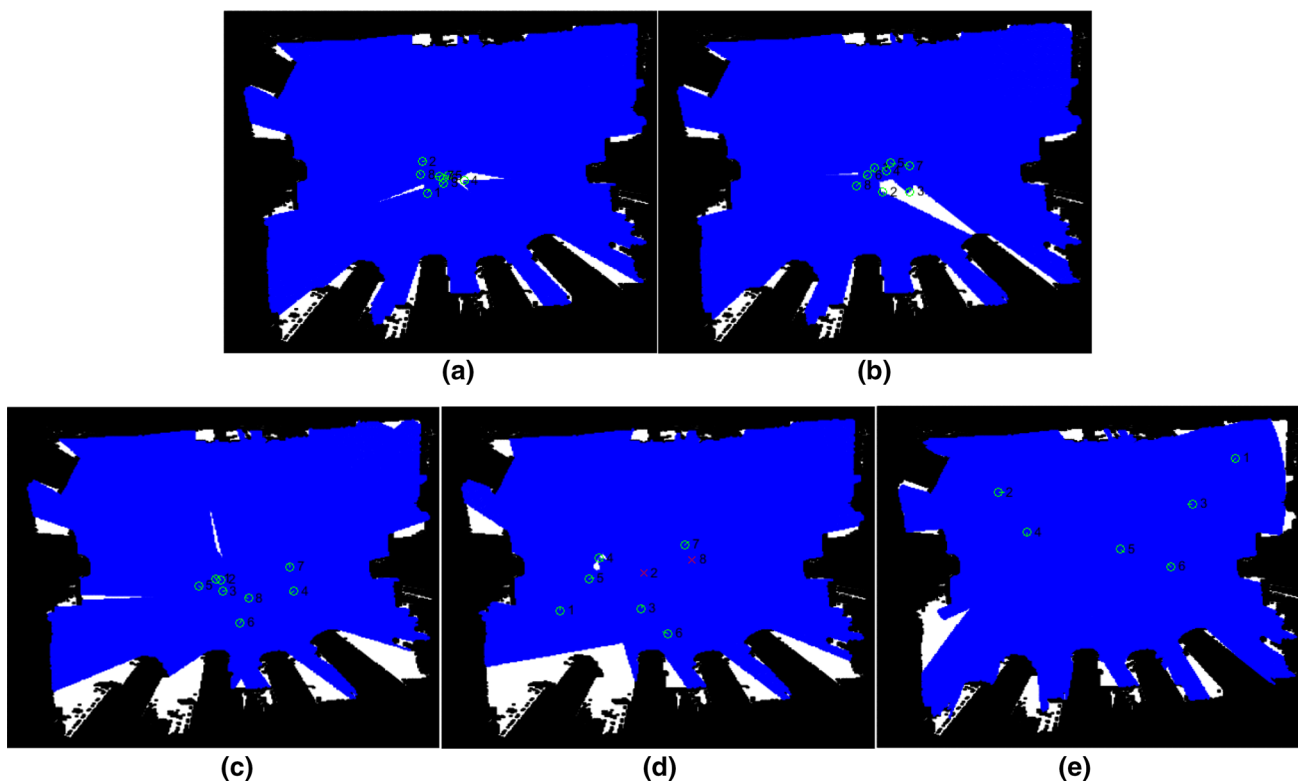


Fig. 11 Maximal coverage of LS, L2, L1, PEN and Greedy algorithms. The black, white, and blue areas are obstacles, uncovered, and covered areas respectively. The green circles and red cross are the robots and the discarded robots respectively (Color figure online)

First, the approaches considering motion cost are faster than the others. Although the greedy approach has higher coverage than SCAO, the greedy subgoals are farther apart than the SCAO subgoals. Hence, the former have higher travel times than the latter. Furthermore, the Greedy + NN has higher search time than SCAO + NN because the former’s motion cost is higher. Greedy + PD is slower than Greedy + NN because the robot moves to the subgoal with higher probability instead of the closest subgoal. For example, the 1st and 2nd chosen subgoals are subgoal 1 and 4, which have high travel times (see Fig. 12). The SCAO + NN and SCAO + PD exhibit similar phenomena.

Second, the PDR approach considers both the PD and motion cost; hence the robot can sweep the environment in the shortest time, which reduces the search time. Furthermore, Greedy + PDR is faster than Greedy + NN because the robot moves to the subgoal with the highest PDR. SCAO + PDR and SCAO + PD exhibit similar behavior. The SCAO + PDR is faster than Greedy + PDR because SCAO subgoals have lower motion costs (see Fig. 13).

In conclusion, the $E[TTD+]$ results demonstrate that the PDR approach is faster than the PD and NN approaches. For example, Greedy and PD are state of the art for coverage and probabilistic search respectively. In experiments, the

Table 2 Predicted coverage error

	LS	L2	L1	PEN
Error	1.136	0.024	0.025	0.040

$E[TTD+]$ of SCAO + PDR is 1.7 times faster than that of Greedy + PD.

5.4 Optimality and time complexity analysis

To evaluate the optimality and time complexity of SCAO and greedy approaches, the solutions of the two approaches are compared. As Fig. 14a shows, the SCAO solution is always better than $(1 - 1/e)OPTi$. Although the greedy solution is better than SCAO, the computation of the greedy approach is higher than SCAO.

The SCAO computations include sensing, learning, and decision making while the primary computation of the greedy algorithm is sensing. The primary computation of the SCAO algorithm is learning, which can be solved by coordinate descent. Hence, the computational complexity is $O(mN^2)$, where m and N are the number of iterations and approximators respectively. The greedy algorithm computes the

Table 3 Overview of the algorithm tested in the experiments

Greedy	The subgoal positions are computed using the greedy algorithm
SCAO	The subgoal positions are computed using SCAO
NN	The robot chooses next subgoal based on nearest Euclidean distance
PD	The robot plans the optimal path based on maximizing cumulative PD. The path constraints are three nodes with the lowest motion cost and the number of time steps (T) is 3
PDR	The robot plans the optimal path based on maximizing cumulative PDR. There is no path constraint and T is 3
Greedy/SCAO+NN	Considering coverage only
Greedy/SCAO+PD	Considering coverage and probability
Greedy/SCAO+PDR	Considering coverage, probability and motion cost

Table 4 Map parameters

CTG map size	$60 \times 60 \times 8$
Grid map size	300×300
CTG map resolution	$10 \times 10 \text{ cm}^2 \times 45^\circ$
Grid map resolution	$2 \times 2 \text{ cm}^2$

coverage according to the selected position and map data. The computational complexity of the greedy algorithm for k -max coverage is $O(k|G|)$, where $|G|$ is the number of available grid points. The major difference in computation between greedy and SCAO algorithms is as follows. The greedy algorithm has to sense the environment when placing each subgoal position. SCAO only senses the environment one time per iteration. For example, with $|G|$ of 6000 and k of 6, the greedy algorithm needs to sense the environment 36000 times. In contrast, SCAO runs 500 iterations. As a result, it only needs to sense the environment 500 times. In the experiments, the greedy approach takes 4 hours while SCAO takes 5 minutes to compute the subgoal positions. As a result, the SCAO's computational time is 50 times faster than the greedy approach's.

To evaluate the optimality and time complexity of PDR and OPT, their PDR are compared. The data is collected from the 50 SCAO + PDR experiments. The OPT computes all possible solutions while PDR computes the greedy solutions. As Fig. 14b shows, the 99% solution of PDR is better than $(1 - 1/e)OPT$. In other words, it shows the high probability guarantee of Theorem 7. PDR average approximation rate is 85%.

The complexity of finding PDR solutions is as follows: The OPT approach needs to compute each branch and finds the branch with maximal PDR. If $\beta = 1$, the PDR BNB approach is equal to the greedy approach. The time complexity of OPT and PDR are $O(|S|^T)$ and $O(T|S|)$ respectively, where $|S|$ is the number of subgoals and T is the planning steps. In the search experiments, with $|S|$ of 6 and T of

Table 5 $E[TTD]$ obtained from the six methods

	$E[TTD+]$ (s)	Std (s)	Successful rate (%)
Greedy+NN	35.8	32.4	82
Greedy+PD	42.4	33.0	82
Greedy+PDR	33.0	19.4	86
SCAO+NN	34.9	22.8	90
SCAO+PD	36.5	18.7	92
SCAO+PDR	24.7	14.3	94

3, optimal solutions require computing 216 solutions while PDR only requires computing 18 solutions. Even if PDR's computational complexity is lower, its performance is near-optimal with high probability. The Table 6 summarizes the complexity and optimality of four approaches.

5.5 Sensitivity analysis

Since the objective functions of coverage and minimum-time trajectory planning problems are coupled, the sensitivity analysis of coverage and motion cost is shown using two experiments. First, SCAO generates 1 to 7 subgoals, 10 times for each. Once the subgoal positions are given, their coverage (C) and motion cost (T) are computed using measurements and CTG map respectively. The time gap (T_g) is defined as $T_g = \frac{T_{max} - T}{T_{max}}$, where T_{max} is the maximal motion cost of all SCAO subgoals. Figure 15a shows that when the number of subgoals increases, the coverage increases while the time gap decreases. In other words, the larger number of subgoals generates higher coverage and higher motion cost.

Second, SCAO generates 5 subgoals, 20 times for each and then 6 subgoals, 20 more times each. Once the subgoal positions are given, their coverage (C) and motion cost (T) are computed using measurements and CTG map respectively. C and T_g data based on subgoals are used for fitting lines.

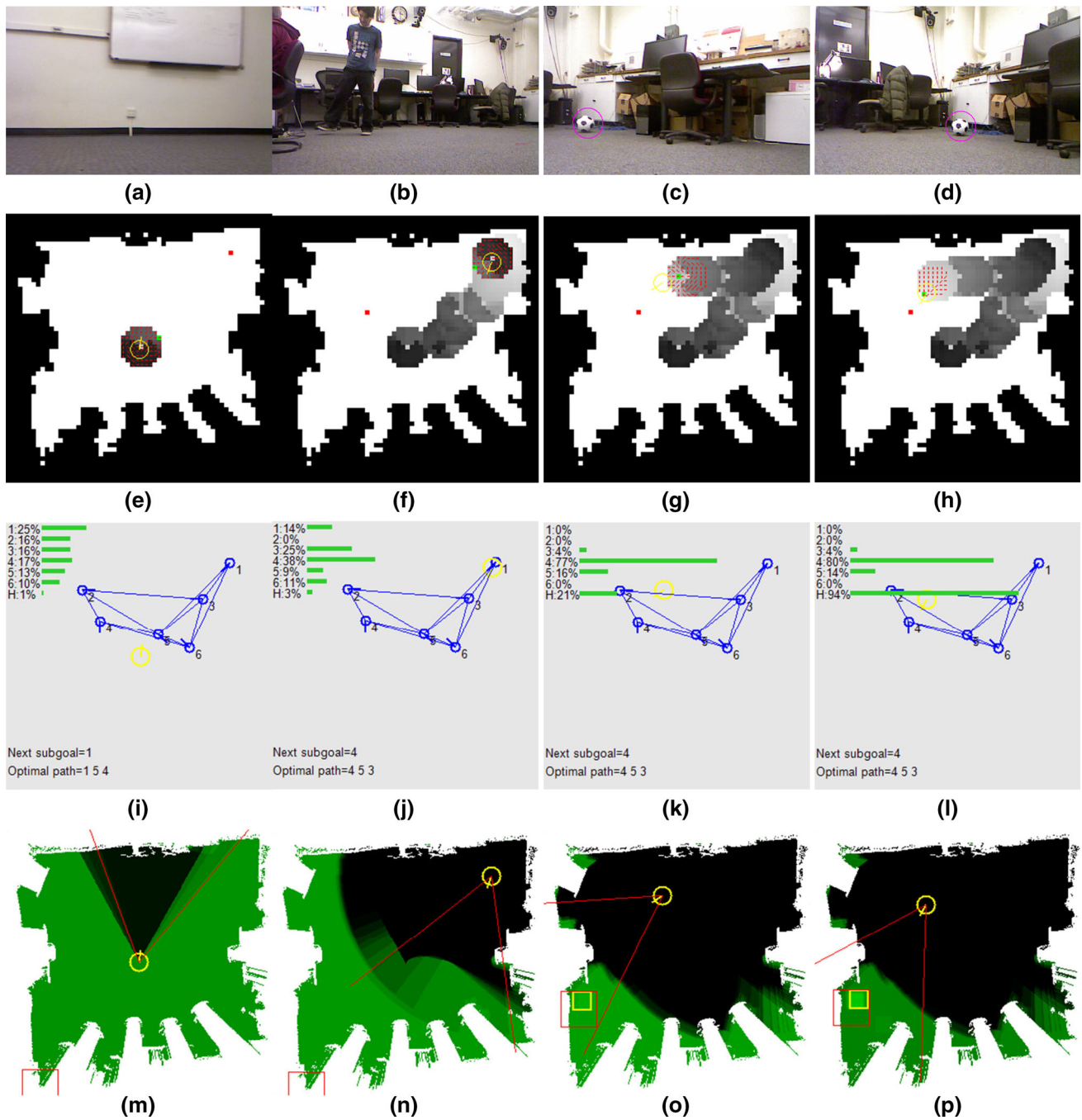


Fig. 12 Search using PD. **a–d** RGB image. **e–h** CTGC HMI. **i–l** PD_BNB HMI. The graph represents the links and nodes for 6 subgoals. The *green bar* and *numbers* represent the probability of each subarea. **m–p** BF HMI. $t = 1.5$: The PD_BNB generates $P_{path} = \{1, 5, 4\}$ and

the robot moves toward subgoal 1. $t = 40$: The PD_BNB generates $P_{path} = \{4, 5, 3\}$ and the robot moves toward subgoal 4. $t = 61.75$: The robot detects the ball before arriving at subgoal 4. $t = 63.5$: The probability of H-area is over 90% (Color figure online)

The fitting lines of 5 subgoals and 6 subgoals are shown in Fig. 15b.

The profit (Z) is defined as $Z = C + \Delta T_g$. If Δ increases, the weight of the motion cost is higher. The goal is to find maximal Z under different Δ values using 5 subgoals or 6 subgoals. As Fig. 15 shows, the optimal values are as follows:

$$\begin{cases} \text{If } \Delta > 0.3, & \text{optimal point is at } A_3 \\ \text{If } 0.17 \leq \Delta \leq 0.3, & \text{optimal point is at } A_2 \\ \text{If } \Delta < 0.17, & \text{optimal point is at } A_1 \end{cases}$$

If $\Delta > 0.3$, the user prefers lower motion cost and should choose 5 subgoals. The optimal value is at A_3 . C and T in

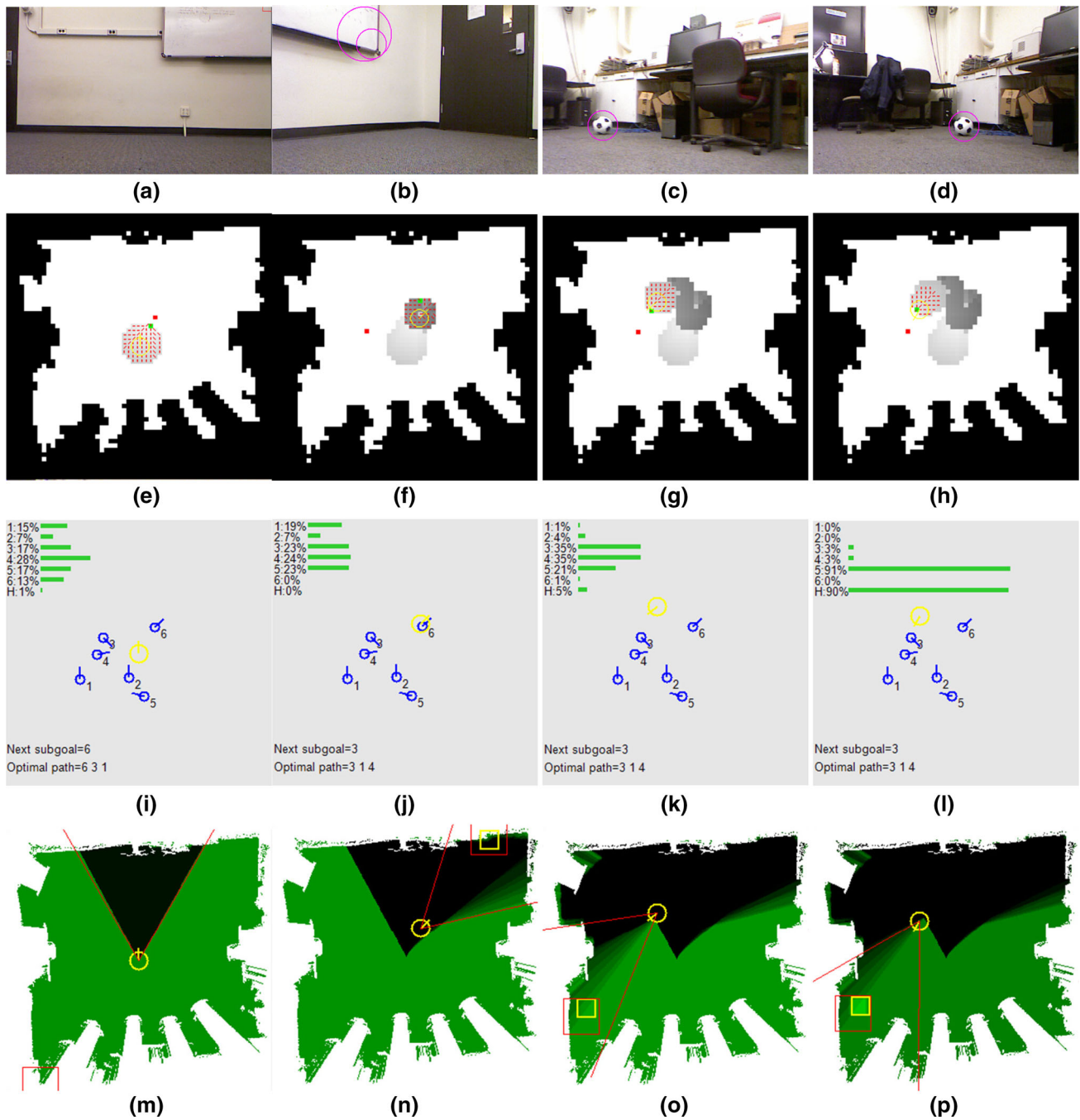


Fig. 13 Search using PDR. $t = 1.5$: The PDR_BNB generates $P_{path} = \{6, 3, 1\}$ and the robot moves toward subgoal 6. $t = 10.5$: The PDR_BNB generates $P_{path} = \{3, 1, 4\}$ and the robot moves toward subgoal 3. $t = 20.75$: The robot detects the ball before arriving at subgoal 3. $t = 22.5$: The probability of H-area is over 90% (Color figure online)

this case are 74% and 21.5 s respectively. If $\Delta < 0.17$, the user prefers higher coverage and should choose 6 subgoals. The optimal value is at A_1 . C and T in this case are 86% and 38.5 s respectively. If $0.17 \leq \Delta \leq 0.34$, the balanced coverage and motion cost are preferred. C and T in this case are 83% and 33.5 s respectively. The user could choose 5 or 6 subgoals to achieve such performance.

5.6 Summary of experiments

The coverage, search, and optimality experiments demonstrate three facts. First, the SCAO can find the subgoals such that coverage is maximized with fewer sensors. Second, the PDR can find the search path such that the PDR along the path is maximal. Third, the optimality of SCAO is near-optimal

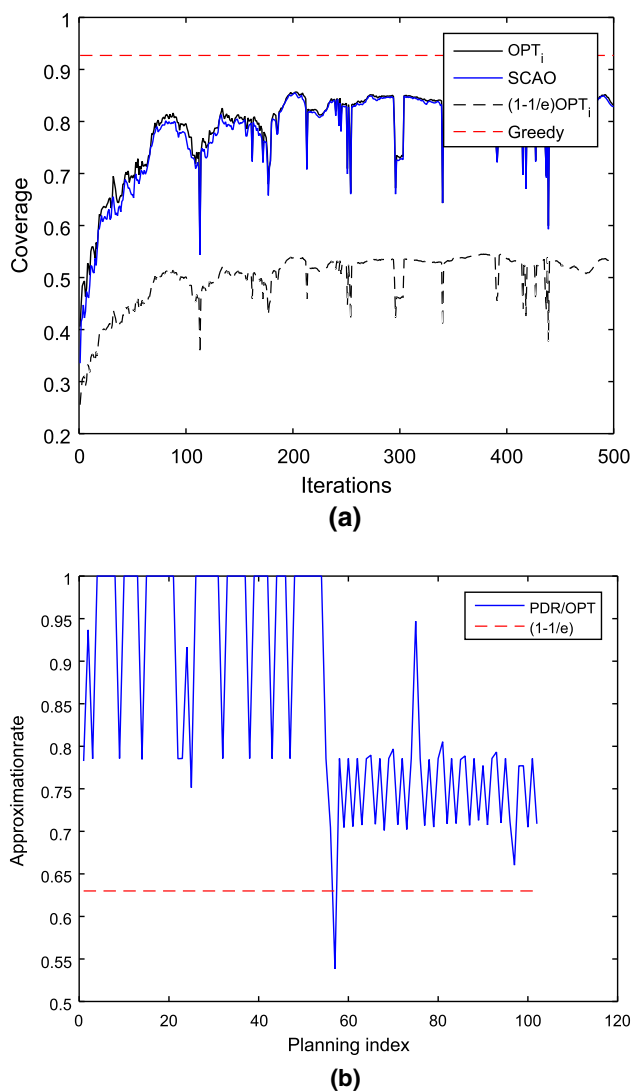


Fig. 14 SCAO and PDR optimality (Color figure online)

and the optimality of PDR is near-optimal with high probability. Finally, the sensitivity analysis shows a way for choosing the desired trade-off between coverage and motion cost.

6 Discussion and future work

In this research, the proposed approach is to enable a mobile robot to search for a static target considering three problems. Searching for a moving target and multiple targets are discussed next.

Table 6 Comparisons of time complexity and optimality

	SCAO	Greedy	PDR	OPT
Complexity	$O(mN^2)$	$O(k G)$	$O(T S)$	$O(S ^T)$
Optimality	$(1 - 1/e)OPT_i$	$(1 - 1/e)OPT$	$(1 - 1/e)OPT$	OPT

First, the proposed formulation can be extended to multiple targets by accounting for multiple likelihood maps and by computing the near-optimal PDR according to the combination of all likelihood maps. For example, searching considers two targets, a soccer ball and a basketball. The robot needs to train one Adaboost classifier for each target. After training, the robot gets two conditional probability functions about each target $P(\phi_s|z_s^D)$ and $P(\phi_b|z_b^D)$, where ϕ_s, ϕ_b, z_s^D and z_b^D denote the soccer ball state, basketball state, detection of soccer ball and detection of basketball respectively.

At the perception stage, the robot creates two likelihood maps first. Then, the robot computes two likelihood maps according to the detections of two targets using a Bayes filter. The space and time complexity are as follows. The likelihood maps in this paper are 300×300 cells and each cell is computed as floats (4 bytes). The storage space of a likelihood map is around 0.3 MB. The computation of each target is around 1 ms. Hence, it is feasible to compute multiple targets' PDF using a Bayes filter.

At the decision-making stage, the robot combines two likelihood maps into a new likelihood map. The new likelihood map displays the combined likelihood maps for the two targets. PDR finds a search path to collect the PDR. The computation of PDR for multiple targets is the same as that for a single target. Once one of the targets is located (H-area is over 90%), the corresponding likelihood map is removed. At this point, search task becomes the single object searching case. The robot keeps searching for another target.

Second, to search for a moving target, perception and decision-making need to be modified. In the perception stage, the difference between a moving and static target is the motion model of Bayes filter (Eq. 23). If the target is moving, Eq. 23 is updated based on the known target motion model $P(X_k^t|X_{k-1}^t)$. An efficient way to implement the motion model is using Gaussian filter (or called Gaussian blur in image processing).

In the decision-making stage, the difference between a moving and static target is that the target's PDF is time-varying. To pursue the target, the robot has to change its subgoals according to the target's PDF. The dynamic coverage algorithm can be implemented by learning a time-varying coverage function (Du et al. 2014).

Another limitation of the proposed approach is that the coverage problem and CTG map are currently computed offline. Therefore, an important future direction is to learn coverage and the CTG functions via reinforcement learn-

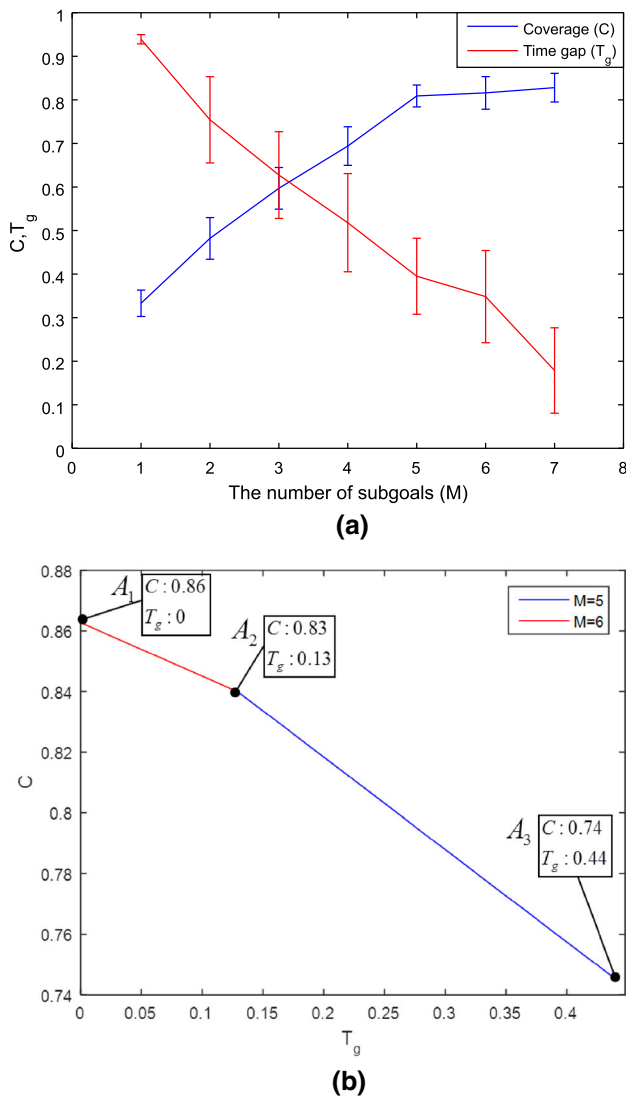


Fig. 15 Sensitivity analysis. **a** The blue and red segments represent coverage and time gap at different the number of subgoals respectively. **b** The blue and red segments represent the fitting lines of 5 subgoals and 6 subgoals respectively. The larger T_g indicates faster motion. The red line ($M = 6$) has higher coverage and slower motion. The blue line ($M = 5$) has lower coverage and faster motion. In this experiment, T_{max} is 38.5 s. $T_g = 0.1$ and $T_g = 0.3$ are equal to 34.65 and 30.8 s respectively (Color figure online)

ing. Second, to search for a moving target with near-optimal guarantee is still a challenge. Learning a time-varying coverage function is a possible direction. Finally, how humans search for targets is still a mystery. The authors are currently performing experiments investigating the gaze of humans in remote control search tasks. The formal search framework can be used to investigate human search strategies, and conversely the results obtained from human experiments could help improve the proposed algorithms.

7 Conclusions

Robotic search combines coverage, probabilistic search and minimum-time trajectory planning problems, which are NP-hard. The proposed approach solves these problems using: (1) SCAO to determine the minimal number of subgoals to perform the sensor coverage task; (2) PDR to determine the path such that the cumulative probability of detection rate is maximal; and (3) MINLP to determine the velocity commands to guide the robot from the current position to each of subgoals in minimal time.

The contributions of this research are as follows: First, the proposed approach enables a mobile robot to search for a target considering sensing coverage, sensing uncertainty, and motion cost. Second, the SCAO algorithm solves the maximal coverage problem using fewer sensors, avoids overfitting, and reduces computation. Third, PDR solves the maximal cumulative PDR with linear time. The advantages of PDR are the consideration of sensing overlap and motion cost. Finally, SCAO gives near-optimal solutions while PDR gives near-optimal solutions with high probability. The theoretical proofs and experiments demonstrate that the SCAO performance exceeds the CAO performance; the PDR performance exceeds the PD performance; the SCAO+PDR performance exceeds the performance of any combinations of existing approaches (e.g. NN, Greedy, CAO and 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 this research. The authors would also like to thank Alessandro Renzaglia for discussing the details about his PhD thesis, the CAO algorithm.

Appendix

Bayes filter

If the probability distribution of the target’s motion $P(X_k^t|X_{k-1}^t)$ and the probability distribution of the robot’s sensor $P(z_k|X_k^t)$ are known, the probability distribution of the target is computed recursively through Eqs. 23 and 24. The distribution of the target is fused based on the motion and sensor models.

$$P(X_k^t|z_{1:k-1}) = \sum_{X_{k-1}^t} P(X_k^t|X_{k-1}^t)P(X_{k-1}^t|z_{1:k-1}) \tag{23}$$

$$P(X_k^t|z_{1:k}) = \eta P(z_k|X_k^t)P(X_k^t|z_{1:k-1}) \tag{24}$$

Table 7 Conditional probability table

$P(Z^D = 1 \phi = 1)$	q
$P(Z^D = -1 \phi = 1)$	$1 - q$
$P(Z^D = 1 \phi = -1)$	0
$P(Z^D = -1 \phi = -1)$	1

where X_k^t is the target’s position at time k , $z_{1:k}$ is the measurement of the sensors at time 1 to k , and η is the normalized factor.

Probability of detection (PD) model

To plan the optimal path, PD model is proposed under the assumption of no false detections (see Table 7). Assume the probability of sensor covered area is $P(\phi = 1) = p$. The PD is defined as $F_{PD} = P(\phi = 1) - P(\phi = 1|Z^D)$. Since Z^D is 1 or -1 , there are two cases of F_{PD} .

No detection case

$$\begin{aligned}
 P(\phi = 1|Z^D = -1) &= \frac{P(Z^D = -1|\phi = 1)P(\phi = 1)}{P(Z^D = -1|\phi = 1)P(\phi = 1) + P(Z^D = -1|\phi = -1)P(\phi = -1)} \\
 &= \frac{1 - q}{(1 - q)p + (1 - p)}p = \frac{1 - q}{1 - pq}p = \alpha p \\
 (\because 0 \leq p \leq 1, \therefore 0 \leq \alpha \leq 1) \\
 F_{PD} &= P(\phi = 1) - P(\phi = 1|Z^D = -1) \\
 &= p - \alpha p = p(1 - \alpha) = pg, \\
 &\text{where } 0 \leq g \leq 1
 \end{aligned}$$

Detection case

$$\begin{aligned}
 P(\phi = 1|Z^D = 1) &= \frac{P(Z^D = 1|\phi = 1)P(\phi = 1)}{P(Z^D = 1|\phi = 1)P(\phi = 1) + P(Z^D = 1|\phi = -1)P(\phi = -1)} \\
 &= \frac{q}{qp}p = \beta p, \quad \text{where } 1 \leq \beta \\
 F_{PD} &= P(\phi = 1) - P(\phi = 1|Z^D = 1) \\
 &= p - \beta p = p(1 - \beta) = pg', \quad \text{where } g' \leq 0
 \end{aligned}$$

Hence, the PD is updated dependent on the detections. For most applications (Lau et al. 2008, 2006), the detection case ($Z^D = 1$) is skipped for planning more horizons.

$$F_{PD} = \begin{cases} p \cdot g, & \text{if no detection } (Z^D = -1) \\ p \cdot g', & \text{if there is a detection } (Z^D = 1) \end{cases}$$

Computing coverage

Definition 1 (Coverage) As Fig. 16 shows, the robot is at X_R and gets the sensing measurements $z = \{r_i, \theta_i\}$, where $i = 1, \dots, N_z$. Assume there are N_g unoccupied cells in

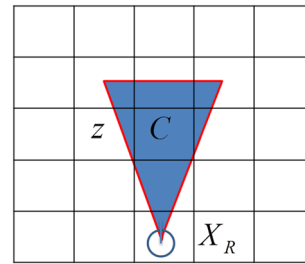


Fig. 16 Illustration of coverage concept. The blue circle and blue area represent the current robot position and corresponding covered area respectively (Color figure online)

a grid map (m). The robot’s sensor covers N_c cells. The coverage (C) is defined as N_c/N_g .

According to X_R and z , the covered area is computed. There are 25 cells and 8 cells are covered by the robot. Hence, the coverage is 32 %.

References

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

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

Binney, J., & Sukhatme, G.S. (2012). Branch and bound for informative path planning. In *IEEE International Conference on Robotics and Automation*.

Bonami, P., Biegler, L. T., Conn, A. R., Cornuejols, G., Grossmann, I. E., Laird, C. D., et al. (2008). An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2), 186–204.

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*.

Chung, T. H., & Burdick, J. W. (2012). Analysis of search decision making using probabilistic search strategies. *IEEE Transactions on Robotics*, 28(1), 132–144.

Chung, T. H., Hollinger, G., & Isler, V. (2011). Search and pursuit-evasion in mobile robotics. *Autonomous Robots*, 31(4), 299–316.

Du, N., Liang, Y., Balcan, M.F., & Song, L. (2014). Learning time-varying coverage functions. *Neural Information Processing Systems*.

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*, 1, 65–100.

Golovin, D., & Krause, A. (2011). Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *The Journal of Artificial Intelligence Research*, 42(1), 427–486.

- Hollinger, G., & Sukhatme, G.S. (2013). Sampling-based motion planning for robotic information gathering. In *Robotics: Science and Systems Conference*.
- Hollinger, G., Kehagias, A., & Singh, S. (2010). Gsst: Anytime guaranteed search. *Autonomous Robots*, 29(1), 99–118.
- Hollinger, G., Choudhuri, C., Mitra, U., & Sukhatme, G.S. (2013). Squared error distortion metrics for motion planning in robotic sensor networks. In *Proceedings International Workshop Wireless Networking for Unmanned Autonomous Vehicles*.
- 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., Mossb, 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.
- Kosmatopoulos, E. B. (2009). An adaptive optimization scheme with satisfactory transient performance. *Automatica*, 45(3), 716–723.
- 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 Intelligent Robots and Systems*.
- 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. Jr. (1999). Randomized kinodynamic planning. In *IEEE International Conference on Robotics and Automation*.
- 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*.
- Lorbert, A., & Ramadge, P.J. (2013). The pairwise elastic net support vector machine for automatic fmri feature selection. In *International Conference on Acoustics, Speech and Signal Processing*.
- Lorbert, A., Eis, D., Kostina, V., Blei, D.M., & Ramadge, P.J. (2010). Exploiting covariate similarity in sparse regression via the pairwise elastic net. In *The 13th International Conference on Artificial Intelligence and Statistics*.
- 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 *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*.
- Nemhauser, G. L., Wolsey, L. A., & Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14, 265–294.
- Nowakowski, R., & Winkler, P. (1983). Vertex-to-vertex pursuit in a graph. *Discrete Mathematics*, 43(2–3), 235–239.
- Renzaglia, A., Doitsidis, L., Martinelli, A., & Kosmatopoulos, E.B. (2011). Adaptive-based distributed cooperative multi-robot coverage. In *American Control Conference*.
- 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.
- Richardson, H. R., & Stone, L. D. (1971). Operations analysis during the underwater search for scorpion. *Naval Research Logistics Quarterly*, 18(2), 141–157.
- 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*.
- Singh, A., Krause, A., & Kaiser, W. (2009). Nonmyopic adaptive informative path planning for multiple robots. In *International Joint Conference on Artificial Intelligence*.
- Stone, L.D. (1975). The theory of optimal search. In *Operations Research Society of America*.
- Stone, L. D., Keller, C. M., Kratzke, T. M., & Strumpfer, J. P. (2014). Search for the wreckage of air france flight af 447. *Statistical Science*, 29(1), 69–80.
- Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58, 267–288.
- Trummel, K. E., & Weisinger, J. R. (1986). The complexity of the optimal searcher path problem. *Operations Research*, 34(2), 324–327.
- Viola, P., & Jones, M.J. (2001). Rapid object detection using a boosted cascade of simple features. In *IEEE Conference Proceedings on Computer Vision and Pattern Recognition*.
- Waharte, S., Symington, A., & Trigoni, N. (2010). Probabilistic search with agile UAVS. In *IEEE International Conference on Robotics and Automation*.
- Washburn, A. R. (1998). Branch and bound methods for search problems. *Naval Research Logistics*, 45(3), 243–257.
- Yuan, M., & Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68, 49–67.
- Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67, 301–320.



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. He is currently a Ph.D. candidate in the Department of Computer Science and Engineering at University of Minnesota. His current research interests include people tracking and searching, reinforcement learning, Bayesian statistics, SLAM, and bio-robotics.



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 MIT's 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 *Identification Modeling and Characteristics of Small-Scale Rotorcraft* (Kluwer, 2003).