

Adaptive Sampling-based View Planning under Time Constraints

Lars Kunze¹ and Mohan Sridharan² and Christos Dimitrakakis³ and Jeremy Wyatt⁴

Abstract—Planning for object search requires the generation and sequencing of views in a continuous space. These plans need to consider the effect of overlapping views and a limit imposed on the time taken to compute and execute the plans. We formulate the problem of view planning in the presence of overlapping views and time constraints as an *Orienteering Problem* with history-dependent rewards. We consider two variants of this problem—in *variant (I)* only the plan execution time is constrained, whereas in *variant (II)* both planning and execution time are constrained. We abstract away the unreliability of perception, and present a sampling-based view planner that simultaneously selects a set of views and a route through them, and incorporates a prior over object locations. We show that our approach outperforms the state of the art methods for the orienteering problem by evaluating all algorithms in four environments that vary in size and complexity.

I. INTRODUCTION

Planning of visual search for objects in large continuous spaces is an open problem with many aspects that make existing solutions inadequate. A set of views must be selected from an infinite number of possible views. To provide any preference ordering over this infinite set, it is necessary to have prior information about where objects might be found, and for that prior to be non-uniform. This formulation of the task of finding the best set of views is a sub-modular problem, and greedy solutions can be shown to have bounded sub-optimality. However, it does not take into account the costs of sequencing those views—the best set of views may, for instance, be time consuming to visit, and there may be another set of views that is slightly worse in terms of the information provided but vastly quicker for a robot to traverse.

The view planning problem is further complicated by four issues. First, some views will overlap and the value of a view, at any point in a sequence of views, will depend on the sequence of views taken so far. Since the values of views are history dependent, the problem ceases to be sub-modular. Second, the sensing process should ideally be modeled as being unreliable, transforming the problem from one of planning in a physical space, to one of planning in belief space. Third, in many practical applications, the time available to execute the planned trajectory of views is constrained. Fourth, existing view planning approaches have not considered the length of time used to plan, which

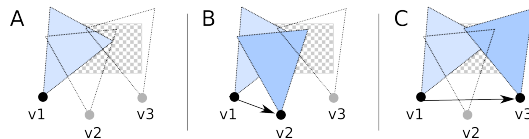


Fig. 1: View planning with overlapping views and time limit. Imagine three possible views onto a table, of which a robot can only observe two in the time available. If view v_1 is taken first (A), then the robot can next take v_2 or v_3 (B or C). The reward for each of these two views depends on its overlap with v_1 . The robot must account for both history-dependent rewards and navigation costs.

will determine how much of the plan (i.e., solution) can be executed in the time available.

We propose a solution to the first, third, and fourth of the issues identified above related to the challenging problem of view planning with overlapping views and time constraints. We do not consider unreliable perception, which transforms view planning into a yet worse class of problems—we leave this for future consideration. In this paper, we refer to the problem of bounded execution time as *variant (I)*, and to the problem of bounded planning and execution time as *variant (II)*. We assume the ability to use sensor inputs to generate 2D and 3D maps of the environment, for navigation and object recognition; and assume prior knowledge about the locations of objects, which can be related to the 3D map. We make the following key contributions:

- We show that the view selection and view sequencing problem can be posed as an orienteering problem (OP) with redundant views and history-dependent rewards.
- We present a Gibbs sampling-based view planning algorithm (GVP) that produces approximate solutions, but will provably converge in the limit to an optimal sequence given a finite set of views.
- We extend our algorithm (GVP) to learn, from a batch of example problems, how to divide the available time between thinking and acting.
- We evaluate our approaches on a range of environments under different time constraints and compare them to the state of the art.

To locate any given object, our algorithm (GVP) first generates a much larger number of candidate views than can possibly be searched in the available time, orders them by the probability of providing a view of the object, and selects the m best views for subsequent analysis. A sampler incrementally selects a sequence of views that simultaneously maximizes the likelihood of finding the object, considering the field of view overlap and the viewing history, and minimizes the time taken

¹Oxford Robotics Institute, Dept. of Engineering Science, University of Oxford, United Kingdom, lars@robots.ox.ac.uk

²Department of Electrical and Computer Engineering, The University of Auckland, New Zealand, m.sridharan@auckland.ac.nz

³Computer Science and Engineering, Chalmers University of Technology, Göteborg, Sweden, chrdimi@chalmers.se

⁴Intelligent Robotics Lab, School of Computer Science, University of Birmingham, United Kingdom, jlw@cs.bham.ac.uk

to do so—see Fig. 1. Many such sampling processes are run, each of which is terminated when the time limit is reached, and the robot chooses the best sequence of views.

For variant (I), we compare GVP to a randomized OP solver, and to two solvers for the traveling salesperson problem (TSP) (1) a fast but sub-optimal greedy TSP solver (TSP-G); and (2) an optimal but slow TSP solver based on dynamic programming (TSP-DP). We show that GVP typically outperforms other algorithms for problems of different sizes and different bounds on execution time. On small problems, TSP-DP produces better solutions, but it is challenging to terminate in reasonable time for larger problems—GVP is thus an effective, if sub-optimal, solution. For variant (II), we evaluate the performance of our algorithm in simulated environments, for object search tasks. We show that our adaptive algorithm outperforms fixed policies.

The remainder of the paper is organized as follows. We discuss related work in Sec. II, and the problem formulation in Sec. III. Sec. IV describes our sampling-based view planning algorithm and implementation details. Sec. V discusses experimental results, and we conclude in Sec. VI.

II. RELATED WORK

Early work on object search discussed its intractability in a continuous space, even under some simplifying assumptions [1]. Subsequent approaches have employed different strategies to address the complexity, e.g., visual saliency [2]; planning at the level of rooms and views with associative knowledge [3], [4]; and search with qualitative spatial relations [5]. Some approaches assume reliable observations, whereas others reason under partial observability using a mix of qualitative and probabilistic reasoning for efficient search, e.g., for estimating target location at the level of rooms [6] or locations in rooms [7], [8].

The fundamental problem of planning a sequence of views to find an object can be viewed as a generalization of the *art gallery problem* and the *watchman problem*. The art gallery problem is NP-hard, and is a generalization of the guarding problem where sensors have infinite range, bounded only by obstacles [9]. A randomized algorithm developed for this problem provides a high probability of bounded error between its coverage and the optimal solution [10]. Approximate algorithms have been proposed for art gallery problem sequenced by a traveling salesman problem (TSP) [11]. With unreliable sensing, the joint art-gallery and watchman problem is a continuous space, continuous action POMDP, but even discrete state, discrete action POMDPs can become computationally intractable. Researchers have partially addressed this intractability through hierarchical decompositions [4].

Our view planning problem is related to planning in belief space, e.g., for task and motion planning [12], although we do not plan in belief space. Also, unlike probabilistic roadmaps [13], we do not restrict the solution space too much, since we consider time constraints. There is also some relation to work in temporal logic planning [14], although

existing approaches do not address the issues of interest and do not scale like our approach.

Our view planning problem is most related to the orienteering problem (OP) [15]. In an OP, a rewarding sequence of locations must be visited, where each location can have (in general) a different associated reward. We use a sampling-based algorithm for the OP as one of the baselines for comparison [16]. OPs, are however, typically stationary reward problems, whereas we must solve a varying-reward OP, also called general OP [17].

We assume reliable perception and work with a continuous state, continuous action MDP with a history-dependent reward function. Our novel contribution is an algorithm for the joint problem of selecting views (art-gallery) and planning a route in continuous space. We present a randomized algorithm that interleaves the selection of views with the selection of the route. Unlike previous work, including those that used stochastic branch and bound [18] or Monte-Carlo tree search [19], our method has very low space complexity.

III. PROBLEM FORMULATION

We decompose the problem of view planning for object search with time constraints in a continuous environment into two parts: (i) transforming the continuous problem into a discrete problem; and (ii) solving the discrete problem using a sampling-based approach. We consider the discrete problem as an Orienteering Problem (OP) with history-dependent rewards, i.e., given a fully connected graph of locations $s \in S$, a cost function $C(s, s')$, and a time limit T , maximize the expected reward $R(s_0, s_1, \dots)$ obtained from a sequence of visited locations—the reward of a location in the sequence depends on the locations that have been visited before. We investigate two variants of this problem formulation. Variant (I) considers only the execution time (T_E) when solving the view planning problem within a time limit, i.e., $T_E \leq T$. Variant (II) considers both planning time (T_P) and execution time (T_E), i.e., $(T_P + T_E) \leq T$.

IV. SAMPLING-BASED VIEW PLANNING

This section provides an overview of our algorithm, followed by a detailed description (Sec. IV-A), an adaptive extension (Sec. IV-B), and the implementation (Sec. IV-C). We assume that we are given a: (1) 2D environment map (M_{2D}); (2) 3D environment map (M_{3D}); (3) probability distribution P of a robot at location s observing an object of a certain type; and (4) function $C(s, s')$ that provides the temporal cost of moving between locations s and s' . Sec. IV-C describes how P and C are computed.

To search for objects within time limit T , the robot generates a trajectory $s = s_0, s_1, \dots, s_{t'}$ composed of a sequence of locations s_t ($t = 0, \dots, t'$). Here, t indexes waypoints in the trajectory—the time from the t -th to the $t+1$ -th location is not fixed, and t' denotes the index of the last feasible waypoint given time limit T . We use $\phi : s \rightarrow \{0, 1\}$ to denote whether or not we can observe the object at location s . Then $P(\phi_t = 1 \mid \phi_{1:t-1}, s_{1:t})$, with $\phi_t = \phi(s_t)$, is the probability of a positive observation at the next time step

Algorithm 1: View planning (phase one): OP construction

```
1 Function GVP-OP-CONST ( $M_{2D}, P, tc$ )
  Input : 2D map  $M_{2D}$ ; prob. dist. of perceiving an object  $P$  (based
    on  $M_{3D}$ ), target coverage  $tc$  ( $0 < tc \leq 1$ )
  Output : Set of locations  $S$ ; cost function  $C$ 
2 begin
3    $S \leftarrow \emptyset$ 
4    $coverage \leftarrow 0$ 
5   while  $coverage < tc$  do
6      $s \leftarrow \text{sampleLocation}(M_{2D})$ 
7      $S \leftarrow S \cup s$ 
8      $coverage \leftarrow coverage + P(s)$ 
9   /* Filter redundant locations  $s \in S$  with zero/low probability ( $P$ )
    such that the coverage constraint holds*/
10   $S \leftarrow \text{filterRedundantLocations}(S)$ 
11  /* Calculate the cost  $C(s, s')$  for all location pairs */
12  for  $s \in S, s' \in S$  do
13     $C(s, s') \leftarrow \text{computeCost}(M_{2D}, s, s')$ 
14  return  $S, C$ 
```

given the trajectory history—Sec. IV-C describes how we compute these probabilities. Given T, P and C , the objective is to find a trajectory s that maximizes expected reward R :

$$R_T(s) = R_T(s_{1:t'}) = \sum_{t=1}^{t'} P(\phi_t = 1 \wedge \phi_k = 0 \forall k < t); \quad (1)$$

with total cost not exceeding time limit T :

$$C(s) = C(s_{1:t'}) = \sum_{t=1}^{t'} C(s_{t-1}, s_t) \leq T. \quad (2)$$

Instead of exhaustively exploring all possible trajectories s , we generate them from a distribution, which prefers trajectories that look the best myopically, and puts a non-zero probability on every path.

A. The Sampling-based Algorithm

The core idea of our algorithm is a two-phase anytime sampling-based optimization of the reward function. The first phase (Algorithm 1) transforms the continuous problem into an OP, and samples a set of possible locations S until the search area is covered to a certain degree (based on P) (Lines 3-8). Locations with zero or low probability are filtered out to satisfy coverage constraint (Line 9), and the cost $C(s, s')$ for all location pairs is calculated (Lines 12-13).

The second phase (Algorithm 2), which solves the OP, first selects the m best locations from S (Line 4), and updates and normalizes P by taking view dependencies into account (Line 6). Next, it generates a series of trajectories defined as an ordered sequence of locations, i.e., $s^k = (s_0^k, s_1^k, \dots, s_{t_k}^k)$, within time limit T (Lines 8-20). All trajectories start from the current robot pose, i.e., $\forall k, s_0^k = s_0$. The t th location of the k th trajectory (s_t^k) is sampled from the following distribution *without replacement* (Line 14):

$$s_t^k \sim P(\phi_t^k = 1 \mid \phi_{1:t-1}^k, s_{1:t}^k) \frac{e^{-\varrho C(s_{t-1}^k, s_t^k)}}{Z} \quad (3)$$

where $t = 1, \dots, t^k$ and $\phi_{1:t-1}^k = 0$ if we have not found the object yet. The exponential expression is the transition distribution of choosing the next location based on costs, and Z is a normalizing constant. The sampling procedure only

Algorithm 2: View planning (phase two): OP solving

```
1 Function GVP-OP-SOLVE ( $S, P, C, T, n_s, m, \varrho$ )
  Input : Set of locations  $S$ ; prob. dist. of perceiving an object  $P$ ; cost
    function  $C$ ; time limit  $T$ ; number of trajectories  $n_s$ ; number
    of locations to be considered  $m$  ( $0 \leq m \leq |S|$ );
    regularization parameter  $\varrho$ 
  Output : Sequence of locations  $s$ 
2 begin
3   /*Select the  $m$  best locations from  $S$  according to  $P$  */
4    $S' \leftarrow \text{selectMBestLocations}(S, m, P)$ 
5   /* Update and normalize  $P$  according to  $S'$  */
6    $P \leftarrow \text{updateAndNormalize}(P, S')$ 
7   /* Generate a set of trajectories  $S$  (of size  $n_s$ ) */
8    $S \leftarrow \emptyset$ 
9   for  $k \leftarrow 1$  to  $n_s$  do
10    /* Initialize sequence with current robot location*/
11     $s^k \leftarrow (s_0^k)$ 
12     $t \leftarrow 1$ 
13    while  $C(s^k) < T$  do
14       $s_t^k \leftarrow \text{sampleNextLocation}(P, C, s^k, \varrho)$ 
15       $s^k \leftarrow \text{append}(s^k, s_t^k)$ 
16       $S' \leftarrow S' \setminus s_t^k$ 
17      /* Update and normalize  $P$  according to new  $S'$  */
18       $P \leftarrow \text{updateAndNormalize}(P, S')$ 
19       $t \leftarrow t + 1$ 
20     $S \leftarrow S \cup s_{0:t-2}^k$ 
21   $s^* \leftarrow \arg \max_{s^k \in S} R_T(s^k)$ 
22  return  $s^*$ 
```

stops when time limit T is exceeded, and discards the last location, i.e., all sampled trajectories account for the time constraint in Equation 2. However, the trajectories can be of different length. Finally, $\varrho \geq 0$ is a parameter used to adjust the influence of the cost function. If $\varrho = 0$, $e^{-\varrho C(s_{t-1}^k, s_t^k)}/Z$ is a uniform distribution, and reward is only based on location and not costs—higher the value of ϱ , greater the preference for locations with lower costs, leading to more cost-effective trajectories. When a decision must be made, the trajectory with the highest reward (so far) is chosen for execution (Equation 1) (Lines 21-22). If there is a tie, the trajectory with the smallest expected cost is chosen. We experimentally analyzed the effect of the parameters in both algorithms. However, finding an optimal setting for the parameters is beyond the scope of this paper.

B. Adaptive View Planning

The algorithm(s) for Variant (I) do not include planning time within the time constraint¹. To account for the planning time T_P in variant (II), we revised the constraint as $T_P + T_E \leq T$, resulting in a trade-off between planning time and execution time. To maximize the expected reward R_T acquired during execution, the robot is required to minimize its planning time. As an algorithm's planning time is influenced by parameters \hat{T} , n_s , and m , their optimal values need to be determined. Since they depend on T and on the problem size $|S|$, we determine them experimentally.

Let us assume a robot is given a new search problem with time limit T' and size $|S'|$. For an optimal solution, the robot

¹We assume a complete plan should be produced before execution. This assumption can be relaxed by allowing concurrent planning and execution.

need to determine values of the parameters that maximize the reward function R_T . Since the planning time P_T is included in the time limit, the robot cannot search the parameter space exhaustively. We propose to approximate the surface of the reward function from known problems. We assume that the robot has computed the reward functions over the parameter space (\hat{T} , n_s , and m) for some (but not all) search problems, and uses this information to interpolate (or extrapolate) the reward function for new search problems. We index pre-computed reward functions by T and $|S|$, i.e. $r(T, |S|)$. To determine the parameters (\hat{T} , n_s , m) that maximize the reward for a new problem $\langle T', |S'| \rangle$, adaptive extension sampling performs the following steps:

- 1) Select two known problems $\langle T_1, |S_1| \rangle$ and $\langle T_2, |S_2| \rangle$
- 2) Interpolate (or extrapolate) $r(T', |S'|)$ from the pre-computed functions $r(T_1, |S_1|)$ and $r(T_2, |S_2|)$.
- 3) Find the maximum in $r(T', |S'|)$ and return the corresponding \hat{T} , n_s , and m . If this m is lower than the expected plan length for a given T' (determined from known problems), we set m to be the expected plan length plus three times its standard deviation to ensure that the available time T' is used by the robot.

In this work we have used a linear interpolation method.

C. Implementation

This section describes the integration of our algorithm with other components, and our algorithm’s implementation.

a) Integration on a robot: We integrated our view planner with the perception and action components of a simulated SCITOS A5 robot (<http://metralabs.com>) equipped with a 2D laser range finder, a depth camera, and a *semantic* camera. The range finder is used to create M_{2D} and for robot localization. The cameras are mounted on a pan-tilt unit (PTU) and have the same field of view. The depth camera is used to generate M_{3D} . The semantic camera is used for object recognition—it returns *true* (*false*) when an object of a given type is (is not) in the field of view. Recall that we assume perfect perception as we are primarily interested in evaluating the planning algorithm—a more realistic sensor model can be included if needed. We also use the motion planning and navigation routines from the Robot Operating System (ROS). The robot can thus be controlled by specifying a target pose, and the cameras can be controlled by specifying the PTU’s angles.

b) Sampling of locations (S): Step 1 of Algorithm 1 samples locations $s \in S$ until a predefined area of M_{3D} is covered (based on P). Each location s is composed of robot pose ($x \in X$) and view pose $v \in V$ variables, i.e., $s = (x, v)$. We first sample a robot pose x from M_{2D} and verify that the robot can reach it. We then generate a number (n_v) of random pan and tilt angles for the PTU (can use fixed angles too), and use the SCITOS robot model’s forward kinematics to compute the poses of the cameras on the PTU. At each pose $x \in X$, the robot takes several views $v \in V$. We repeat this process until the predefined space is covered.

c) Probability distribution $P(\phi_t = 1|s_{1:t})$: The probability distribution P is based on the assumption that objects rest on surfaces. From a given M_{3D} , we first extract the supporting surfaces based on the estimated normal of each voxel. Then, we identify the supporting surfaces’ voxels that would be observed at each generated pose, by counting the number of voxels that lie within a frustum projected to the pose s . This provides the initial distribution over views, i.e., $P(\phi_0 = 1)$. Since we sample views without replacement, we remove any selected views, update the probabilities of dependent views, and normalize the distribution. We treat overlapping views as mutually dependent; once a view is chosen, we update the probabilities of all dependent views. We do this until we reach a plan length t^k .

d) Cost function $C(s, s')$: The cost, represented as time in seconds, of moving between locations s and s' is the maximum of two sub-costs (1) navigation cost C_{nav} ; and (2) pan-tilt unit cost C_{ptu} —we assume the robot can navigate and operate its PTU concurrently:

$$\begin{aligned} C(s, s') &= C((x, v), (x', v')) \\ &= \max(C_{nav}(x, x'), C_{ptu}(v, v')). \end{aligned} \quad (4)$$

To compute the navigation cost for a pair of robot poses (x, x') , we call the motion planner in ROS, retrieve a trajectory of waypoints, calculate the linear and angular distances between all consecutive waypoints, multiply them by their respective velocities, take the maximum of the linear and angular durations, and sum them up. The PTU cost is calculated by multiplying the differences between the current and the target pan-tilt angles by their respective velocities, and computing the maximum of these values.

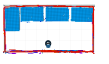


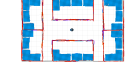
V. EXPERIMENTAL EVALUATION

We evaluated our view planning algorithm in simulation. For variant (I), we compared our algorithm with baseline algorithms in four simulated office environments that vary in size and complexity (Tab. I). The hypothesis was that our algorithm would scale better than the baseline algorithms, especially in larger environments. We used the expected reward as the performance measure. For variant (II), we demonstrate the selection of our algorithm’s parameters as a function of problem size ($|S|$) and time limit (T), thus supporting scaling by adapting to spatio-temporal constraints. The hypothesis was that our adaptive approach would outperform fixed strategies in novel environments. We measured the (a) expected rewards from the generated plans; and (b) performance when the plans were executed in a simulated environment—for the latter, performance was measured by the number of objects found.

A. Experimental Results: Simulation

Simulation trials were conducted in the robot simulator MORSE [20]. For each environment in Tab. I, we generated 2D and 3D maps, and sampled locations such that 95% of the space was covered. In all environments, the robot had a predefined starting location.

TABLE I: Experimental Environments

Environment	E1	E2	E3	E4
Size (m^2)	Small ($32m^2$)	Medium ($96m^2$)	Large ($168m^2$)	Huge ($240m^2$)
Coverage (%)	95%	95%	95%	95%
Locations ($ S $) (unfiltered)	91 (322)	139 (402)	319 (1462)	399 (1533)
3D grid size (#voxels)	2362	4519	8775	12950
3D occupancy grid maps				

1) *Variant (I)*: For different time limits T , we compared our algorithm (GVP) with a stochastic algorithm for the OP [16] (OP-S), and two sequential approaches that greedily select m best views and sequence them using a TSP solver—one sequences greedily (TSP-G) and the other uses dynamic programming (TSP-DP). We expect our approach to scale better in larger environments as it chooses locations freely from the initial distribution—TSP-based methods compute a solution for a fixed set. Tab. II summarizes the results.

Since the sampled locations cover 95% of the space, 0.95 is the maximum reward possible. All approaches degrade in performance as the environments grow larger. Among the two sampling-based approaches, GVP is superior to OP-S in all environments, except in E2 with $T = 120$. Although OP-S achieves a comparable performance in smaller environments, performance is much worse in larger environments. Similarly, TSP-G performs well for the small environments, but the results for the larger environments are poor, as hypothesized. TSP-DP provided good results in multiple trials, but it could not compute a solution in environments E1-E3 for longer intervals, e.g., although planning time is not considered, a solution could not be computed on a standard laptop in ≤ 10 minutes—TSP-DP was able to sequence no more than 21 locations. The performance of the TSP-based approaches depends on the spatial distribution of the best m views. Hence, their performance cannot be predicted easily. Overall, our algorithm provides effective, if sub-optimal, solutions for a range of environments and time limits.

2) *Variant (II)*: The planning time of the algorithms would have had a considerable effect on the results reported above if it were included in the time limits, e.g., T_P (in seconds) of algorithms in Tab. II were: OP-S (5-7); TSP-G (4-6); TSP-DP (4-584); and GVP (102-321). We performed an additional set of over $55k$ trials that evaluated the expected reward by considering both T_P and T_E (for E1 and E3). In Fig. 2, we illustrate the expected reward in environment E3 corresponding to two conditions: (a) $T_E \leq T$, and (b) $T_P + T_E \leq T$. The expected reward in (b) was calculated based on partial trajectories—only locations visited within T contributed to the reward. The reward function in (a) shows a rapid increase with respect to m ; most of the configurations in

TABLE II: Comparison with baseline algorithms. Configuration for OP-S and GVP: $n_s = 250$; $m = 80$; $\varrho = 1.0$.

T	OP-S		TSP-G		TSP-DP		GVP	
	R_T	T_E	R_T	T_E	R_T	T_E	R_T	T_E
Environment E1								
120	0.81	119	0.59	67	0.95	116	0.89	113
240	0.93	234	0.92	131	0.95	136	0.95	237
360	0.94	359	0.95	216	-	-	0.95	358
480	0.94	471	0.95	235	-	-	0.95	429
600	0.95	597	0.95	272	-	-	0.95	479
Environment E2								
120	0.51	119	0.35	116	0.25	45	0.36	110
240	0.54	239	0.70	233	0.90	239	0.57	239
360	0.64	341	0.83	328	0.92	265	0.68	358
480	0.91	479	0.89	450	-	-	0.95	477
600	0.91	597	0.91	587	-	-	0.95	530
Environment E3								
120	0.17	118	0.17	135	0.08	64	0.28	118
240	0.28	226	0.30	220	0.36	207	0.39	237
360	0.35	334	0.46	332	0.61	358	0.57	324
480	0.50	477	0.57	458	0.87	459	0.60	478
600	0.56	597	0.66	552	-	-	0.76	596
Environment E4								
120	0.08	118	0.10	96	0.05	59	0.17	117
240	0.12	233	0.20	217	0.15	230	0.30	239
360	0.26	356	0.29	337	0.24	317	0.42	353
480	0.32	470	0.37	437	0.45	479	0.54	479
600	0.36	597	0.52	583	0.72	575	0.67	597

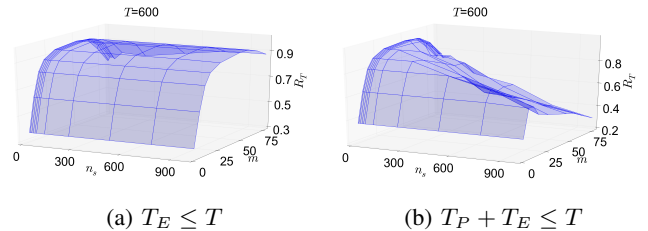


Fig. 2: Reward function for E3 ($T = 600$) considering (a) only execution time, and (b) both planning time and execution time. The range of parameter values explored $n_s \in \{5, \dots, 1000\}$, $m \in \{5, \dots, |S|\}$, $\hat{T} \in \{30, \dots, 600\}$ (only maximal reward with respect to \hat{T} is shown).

(a) lead to a high reward (mainly dependent on m). However, in (b), configurations with large n_s and m lead to low or even zero rewards as planning with an increasing number of views and trajectories becomes more expensive. The selection of appropriate values for parameters is thus critical.

Our adaptive view planning algorithm determines values of parameters n_s and m for a given time limit T and problem size $|S|$ based on planning results of known environments (cf. Sec. IV-B). For instance, consider environments E1 and E3 to have been explored—Fig. 2 shows some of the reward functions for these environments. Based on these reward functions we derive parameters for a novel, medium-sized environment (E2) through interpolation, and a large-sized environment (E4) through extrapolation. We hypothesize that our adaptive approach, can outperform *fixed* strategies in novel environments because it can better adjust to the problem size.

Fig. 3 shows the average performance of our adaptive approach and two fixed strategies ($f1$ and $f2$) in 3900 simulations of object search tasks in all environments (E1-E4).

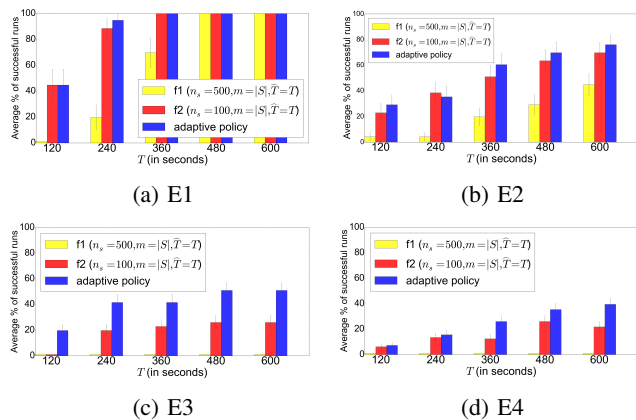


Fig. 3: **Average performance per time T .** Average fraction of successful trials for single-object search tasks in E1-E4, considering T_P and T_E . For each time limit T , each strategy was executed multiple times, proportional to the surface area of an environment (E1: 20x, E2: 40x, E3: 80x, E4: 120x; with a total of 3900 searches).

Each trial is successful if a single object (whose location is unknown) is perceived. Performance is measured as the percentage of successful trials. The *adaptive policy* performs at least as well as the fixed strategies in all cases but one. The performance improvement is statistically significant in E2, E3, E4 when our adaptive approach is compared with fixed strategy $f1$. Similarly, the results are statistically significant in E3 and E4 when our approach is compared with $f2$. The fixed strategies use the full set of views ($|S|$), whereas the adaptive strategy only uses a subset of views according to the problem size and time limit, thus reducing planning time and leaving more time for execution. The performance improvement is particularly pronounced in the larger environments (E3, E4) where the fixed strategies perform poorly.

Results in Fig. 3 indicate that adaptive parameter selection generalizes to unknown environments (E2 and E4), and that it outperforms the best fixed strategy ($n_s = 100, m = |S|, \hat{T} = T$). This is not entirely surprising, as any fixed strategy will eventually fail as problem size and/or time limit are changed. The adaptive strategy performs worse than one of the fixed strategies in E2 ($T = 240$). Although our adaptive strategy chose the optimal set of plan parameters based on the reward function, the actual performance was sub-optimal. If robot encounters a significant mismatch between the expected reward and its actual performance, this could be an indication that its prior knowledge (here P) is incorrect. In such situations, a robot can optimize its parameters from its actual performance, e.g., through reinforcement learning. These results encourage us to further explore adaptive planning.

VI. CONCLUSIONS

This paper has presented a sampling-based algorithm for the challenging open problem of view planning for object search under time constraints. We posed this problem as an OP with history-dependent rewards, and imposed a time limit. Experimental results indicate that our approach outperforms state of the art methods. Furthermore, our adaptive strategy

generalizes well in comparison with fixed sampling strategies for object search in different (new) environments.

Although the proposed approach has been integrated with the components of a mobile robot, the algorithm itself does not depend on any kinematic structure, e.g., it can be used to plan views of a camera on a manipulator arm or a quadcopter. Future work will further explore view planning with time constraints, under partial observability.

REFERENCES

- [1] J. Tsotsos, "On the relative complexity of active vs. passive visual search," *International Journal of Computer Vision*, vol. 7, no. 2, pp. 127–141, 1992.
- [2] S. Frintrop, *VOCUS: A visual attention system for object detection and goal-directed search*. Springer, 2006, vol. 3899.
- [3] M. Lorbach, S. Höfer, and O. Brock, "Prior-assisted propagation of spatial information for object search," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.
- [4] S. Zhang, M. Sridharan, and C. Washington, "Active visual planning for mobile robot teams using hierarchical POMDPs," *Robotics, IEEE Trans. on*, vol. 29, no. 4, pp. 975–985, 2013.
- [5] L. Kunze, K. Doreswamy, and N. Hawes, "Using qualitative spatial relations for indirect object search," in *IEEE International Conference on Robotics and Automation*, Hong Kong, China, 2014.
- [6] J. Wyatt, A. Aydemir, M. Brenner, M. Hanheide, N. Hawes, P. Jensfelt, M. Kristan, G. Kruijff, P. Lison, A. Pronobis *et al.*, "Self-understanding and self-extension: A systems and representational approach," *Autonomous Mental Development, IEEE Trans. on*, vol. 2, no. 4, pp. 282–303, 2010.
- [7] M. Hanheide, M. Gobelbecker, G. Horn, A. Pronobis, K. Sjojo, P. Jensfelt, C. Gretton, R. Dearden, M. Janicek, H. Zender, G.-J. Kruijff, N. Hawes, and J. Wyatt, "Robot Task Planning and Explanation in Open and Uncertain Worlds," *Artificial Intelligence*, 2015.
- [8] S. Zhang, M. Sridharan, and J. Wyatt, "Mixed Logical Inference and Probabilistic Planning for Robots in Unreliable Worlds," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 699–713, 2015.
- [9] B. Nilsson, "Guarding art galleries: Methods for mobile guards," Ph.D. dissertation, Lund University, 1995.
- [10] H. González-Baños, "A randomized art-gallery algorithm for sensor placement," in *Proceedings of the Seventeenth Annual Symposium on Computational Geometry*. ACM, 2001, pp. 232–240.
- [11] A. Sarmientoy, R. Murrieta-Cid, and S. Hutchinson, "A sample-based convex cover for rapidly finding an object in a 3-d environment," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 3486–3491.
- [12] D. Hadfield-Menell and E. Groshev and R. Chitnis and P. Abbeel, "Modular Task and Motion Planning in Belief Space," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, September 28–October 2, 2015.
- [13] Roland Geraerts and Mark H. Overras, "A Comparative Study of Probabilistic Roadmap Planners," in *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, Nice, France, December 15-17, 2002.
- [14] C. Vasile and C. Belta, "An Automata-Theoretic Approach to the Vehicle Routing Problem," in *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.
- [15] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, "The orienteering problem: A survey," *Eur. J. Oper. Res.*, vol. 209, no. 1, pp. 1–10, 2011.
- [16] T. Tsiligrirides, "Heuristic methods applied to orienteering," *The Journal of the Operational Research Society*, vol. 35, no. 9, pp. 797–809, 1984.
- [17] Q. Wang, X. Sun, B. L. Golden, and J. Jia, "Using artificial neural networks to solve the orienteering problem," *Ann. Oper. Res.*, vol. 61, no. 1, pp. 111–120, 1995.
- [18] C. Dimitrakakis, "Complexity of stochastic branch and bound for belief tree search in Bayesian reinforcement learning," in *2nd International Conference on Agents and Artificial Intelligence*, 2009, pp. 259–264.
- [19] A. Guez, D. Silver, and P. Dayan, "Efficient Bayes-adaptive reinforcement learning using sample-based search," in *Advances in Neural Information Processing Systems*, 2012, pp. 1025–1033.
- [20] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan, "Modular openrobots simulation engine: Morse," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*, 2011.