

# View Planning with Time Constraints — An Adaptive Sampling Approach

Lars Kunze<sup>1</sup> and Mohan Sridharan<sup>2</sup> and Christos Dimitrakakis<sup>3</sup> and Jeremy Wyatt<sup>4</sup>

**Abstract**—Planning for object search requires the generation and sequencing of views in a continuous space. These generated plans need to consider the effect of overlapping views, and there is typically a limit imposed on the time taken to compute and execute the plans. We formulate this challenging problem of view planning in the presence of overlapping views and time constraints as an *Orienteering Problem* with history-dependent rewards. In this paper, we focus on the aspect of the problem in which the plan execution time is constrained, but not the planning time. 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. All algorithms are evaluated in four environments that vary in size and complexity, using a robot simulator that includes a realistic model of robot dynamics. We also demonstrate the robustness of our approach through long-term robot deployment in a real-world environment.

## I. INTRODUCTION

Planning of visual search for objects in large continuous spaces is an open problem in robotics. The problem has many aspects that together 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 thus greedy solutions can be shown to have bounded sub-optimality. However, it does not take into account the costs of sequencing those views—it is possible that the best set of views is time consuming to visit, and there exists another set of views that is only slightly worse but is 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, e.g., for a security robot,

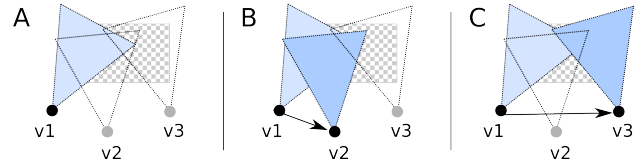


Fig. 1: View planning with overlapping views and time limit. Imagine three possible views onto a table, of which a robot can only observe two in the time available. If view  $v1$  is taken first (A), then the robot can next take  $v2$  or  $v3$  (B or C). The reward for each of these two views depends on its overlap with  $v1$ . The robot must account for both history-dependent rewards and navigation costs.

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

In the context of the challenging problem of view planning with overlapping views and time constraints, we propose a solution to the first and third of the issues identified above. We do not consider unreliable perception, which transforms view planning into a yet worse class of problems, or constraints on planning time—we leave these for future consideration. We assume the ability to use sensor inputs to generate 2D and 3D occupancy grid 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 joint view selection and view sequencing problem can be posed as an orienteering problem (OP) on a redundant set of views, where the rewards are history dependent.
- 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 evaluate our sampling-based view planning algorithm on a range of environments under different time constraints and compare it to the state of the art.
- We demonstrate that our view planning approach is applicable to real-world environments.

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

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

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

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

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

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 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.

We compare GVP to a randomized OP solver, and to two solvers for the traveling salesperson problem (TSP)—a fast but sub-optimal greedy TSP solver (TSP-G), and an optimal but slow TSP solver based on dynamic programming (TSP-DP). We show that GVP typically outperforms other algorithms on 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.

The remainder of the paper is organized as follows. We discuss related work in Sec. II, and formulate the problem of view planning for object search in Sec. III. Sec. IV describes our sampling-based view planning algorithm and the corresponding implementation details. Sec. V discusses experimental results in simulation and on a physical robot in an office environment, and we conclude in Sec. VI.

## II. RELATED WORK

The problem of searching for objects in human environments has been addressed many times. Early work discussed the intractability of object search in a continuous space, even under some simplifying assumptions [1], and devised fast approximate methods [2]. Subsequent approaches have employed different strategies to ameliorate the complexity. Examples include visual saliency [3]; search planning at the level of rooms and views, coupled with room-object associations [4], [5], [6] or place-object associations [7]; indirect search using qualitative spatial relations [8]; and reward maximization for a given task [9]. Some approaches assume reliable observations [2], whereas some others reason about observation failure in a probabilistic framework [10]. Many approaches address the difficulty of reasoning under partial observability by employing a mix of qualitative and probabilistic reasoning, e.g., the combination of non-monotonic reasoning for estimating target location at the level of rooms [11], and the use of decision theoretic and logic-based planning [4], [12], [13]. Almost all of these approaches exploit a combination of probabilistic and relational knowledge to make search more efficient.

The fundamental problem of planning a sequence of views to find an object can be formulated as a generalization of two other problems, 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 [14]. A randomized algorithm developed for this problem provides a high probability of bounded error between its coverage and the optimal solution [15]. Approximate algorithms have been proposed to solve the art gallery problem sequenced by a traveling salesman problem (TSP) [16]. If sensing becomes unreliable, the joint art-gallery and watchman problem is

a continuous space, continuous action POMDP, but even discrete state, discrete action POMDPs can become intractable for complex domains. Researchers have partially addressed this intractability by employing hierarchical decompositions, e.g., for visual processing [6].

Our view planning problem is related to planning in belief space, e.g., for task and motion planning [17], although we do not plan in belief space in this work. Also, unlike probabilistic roadmaps [18], we do not restrict the solution space too much, since we consider time constraints. Further, there is some relation to work in temporal logic planning [19], although existing approaches do not address all the issues of interest, and will not scale as well as our approach.

Our view planning problem is most related to the orienteering problem (OP) [20]. 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 OP [21] as one of the baselines for comparison. OPs, are however, typically stationary reward problems, whereas we, due to overlapping views, must solve a varying-reward OP, also called general OP [22].

In this paper, we assume reliable perception and thus address 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. In addition, unlike previous work, which used sparse sampling [23], stochastic branch and bound [24] or Monte-Carlo tree search [25], our sampling 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. Although our proposed algorithm includes both parts, we primarily focus on (ii). We consider the discrete problem as an Orienteering Problem (OP) with history-dependent rewards—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 a sequence depends on the locations that have been visited before. As stated before, we only consider the execution time ( $T_E$ ) when solving the view planning problem within a given time limit, i.e.,  $T_E \leq T$ . The following section describes how we construct the OP and generate a solution using a sampling-based approach.

## IV. SAMPLING-BASED VIEW PLANNING

This section first provides an overview of our view planning algorithm, followed by a detailed description of the steps of the algorithm (Section IV-A). We then describe the implementation (Section IV-B).

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, which is computed based on  $M_{3D}$ ; and (4) function  $C(s, s')$  that provides the temporal cost of moving between locations  $s$  and  $s'$ . Section IV-B describes how  $P$  and  $C$  are computed.

To search for objects within time limit  $T$ , the robot generates a trajectory  $\mathbf{s} = s_0, s_1, \dots, s_{t'}$ , composed of a sequence of locations  $s_t$  ( $t = 0, 1, \dots, t'$ ). Note that  $t$  merely indexes waypoints in the trajectory—the time taken 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  $T$ . At each point  $s$  in the trajectory, there is a chance that the target object will be found. We use  $\phi : s \rightarrow \{0, 1\}$  to denote whether or not we can observe the object at location  $s$ . Then  $P(\phi_t = 1 \mid \phi_{1:t-1}, \mathbf{s}_{1:t})$ , with  $\phi_t = \phi(s_t)$ , is the probability of a positive observation at the next time step given the trajectory history. These probabilities may be determined in an *ad-hoc* manner or through Bayesian inference—Section IV-B describes how they are determined in our implementation.

Given  $T$ ,  $P$  and  $C$ , the objective is to find a trajectory  $\mathbf{s}$  to find an object, which maximizes the expected reward  $R$ :

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

under the constraint that the total cost of the trajectory does not exceed the time limit  $T$ :

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

Instead of exhaustively exploring all possible trajectories  $\mathbf{s}$ , 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), which transforms the continuous problem into an OP, samples a set of possible locations  $S$  independently until the search area is covered to a certain degree (based on  $P$ ) (Lines 3-8). Locations with zero/low probability are filtered out such that the coverage constraint holds (Line 9). Further, 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). Then it 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.,  $\mathbf{s}^k = (s_0^k, s_1^k, \dots, s_{t^k}^k)$ , within time limit  $T$  (Lines 8-20). All trajectories start from  $s_0$ , the current pose of the robot, 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):

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

---

#### 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      $s \leftarrow sampleLocation(M_{2D})$ 
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   $S \leftarrow filterRedundantLocations(S)$ 
11  /* Calculate the cost  $C(s, s')$  for all location pairs */
12  for  $s \in S, s' \in S$  do
13     $C(s, s') \leftarrow computeCost(M_{2D}, s, s')$ 
14  return  $S, C$ 

```

---



---

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

---

```

1 Function GVP-OP-SOLVE ( $S, P, C, T, n_s, m, \rho$ )
  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  $\rho$ 
  Output : Sequence of locations  $\mathbf{s}$ 
2 begin
3   /*Select the  $m$  best locations form  $S$  according to  $P$  */
4    $S' \leftarrow selectMBestLocations(S, m, P)$ 
5   /* Update and normalize  $P$  according to  $S'$  */
6    $P \leftarrow updateAndNormalize(P, S')$ 
7   /* Generate a set of trajectories  $\mathcal{S}$  (of size  $n_s$ ) */
8    $\mathcal{S} \leftarrow \emptyset$ 
9   for  $k \leftarrow 1$  to  $n_s$  do
10    /* Initialize sequence with current robot location*/
11     $\mathbf{s}^k \leftarrow (s_0^k)$ 
12     $t \leftarrow 1$ 
13    while  $C(\mathbf{s}^k) < T$  do
14       $s_t^k \leftarrow sampleNextLocation(P, C, \mathbf{s}^k, \rho)$ 
15       $\mathbf{s}^k \leftarrow append(\mathbf{s}^k, s_t^k)$ 
16       $S' \leftarrow S' \setminus s_t^k$ 
17      /* Update and normalize  $P$  according to new  $S'$  */
18       $P \leftarrow updateAndNormalize(P, S')$ 
19       $t \leftarrow t + 1$ 
20     $\mathcal{S} \leftarrow \mathcal{S} \cup \mathbf{s}_{0:t-2}^k$ 
21   $\mathbf{s}^* \leftarrow \arg \max_{\mathbf{s}^k \in \mathcal{S}} R_T(\mathbf{s}^k)$ 
22  return  $\mathbf{s}^*$ 

```

---

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 dependent on costs, and  $Z = \sum_{s_i^k \notin s_{1:t-1}^k} e^{-\rho C(s_{t-1}^k, s_i^k)}$  is a normalizing constant. The sampling procedure only stops when time limit  $T$  is exceeded, and discards the last location, i.e., all sampled trajectories are designed to account for the time constraint in Equation 2. However, the trajectories can be of different length, although the last node has an index  $t^k$ . Finally,  $\rho \geq 0$

is a parameter used to adjust the influence of the cost function. If  $\rho = 0$ ,  $e^{-\rho C(s_{i-1}^k, s_i^k)} / Z$  is a uniform distribution, and reward will only be based on location and not consider costs—higher the value of  $\rho$ , 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 found so far is chosen for execution (Equation 1) (Lines 21-22). If there is a tie between trajectories, 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 them is beyond the scope of this paper.

## B. Implementation

This section describes the integration of our algorithm with other components on a robot, followed by a description of 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<sup>1</sup> equipped with a 2D laser range finder, a depth camera, and a *semantic* camera. The range finder is used to create a 2D map of the environment ( $M_{2D}$ ), and to localize the robot during navigation. The cameras are mounted on a pan-tilt unit (PTU) and have the same field of view. The depth camera is used to generate a 3D occupancy grid map ( $M_{3D}$ ) [26]. 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. The semantic camera’s field of view is realistic and similar to a Kinect, but we assume perfect perception as we are primarily interested in evaluating the planning algorithm—a more realistic sensor model can be included. We also use the motion planning and navigation routines from the Robot Operating System (ROS)<sup>2</sup>. The robot can thus be controlled by specifying a target pose, and the cameras can be controlled by specifying the PTU’s angles. Next, consider the implementation of our view planning algorithm.

*b) Sampling of locations ( $S$ ):* Step 1 of Algorithm 1 samples locations  $s \in S$  until a predefined area of  $M_{3D}$  is covered. 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 navigate to 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 robot pose  $x \in X$ , the robot thus takes several views  $v \in V$ . We repeat this process until the predefined space is covered. Fig. 2 shows the robot poses and view poses (black arrows) generated in a mapped office environment.

*c) Probability distribution  $P(\phi_t = 1 | s_{1:t})$ :* The generated poses are used to calculate the probability of observing an object. The probability distribution  $P$  is based on the assumption that objects rest on surfaces in the environment.

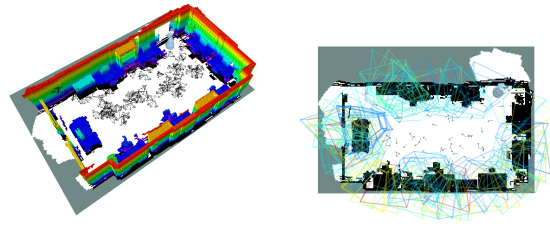


Fig. 2: Left: our algorithm generates a set of robot poses, and a set of views (black arrows) at each pose to search for an object. Right: views are evaluated by counting the number of 3D voxels (of supporting surfaces) that lie within a view frustum. Colors indicate the probability of observing the object in a given view (red: high, blue: low).

From a given  $M_{3D}$  (Fig. 2, left), 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  $s$  (Fig. 2, right). This provides the initial distribution over views, i.e.,  $P(\phi_0 = 1)$ . Since we sample views without replacement, we remove any selected view from the distribution, update the probabilities of dependent views, and normalize the distribution. We treat overlapping views as mutually dependent, i.e., once a view is chosen, we update the probabilities of all dependent views. We do this until we reach a plan length  $t^k$ . The expected reward  $R_T$  is calculated based on this probability distribution.

*d) Cost function  $C(s, s')$ :* we represent cost as time (in seconds). The cost of moving between locations  $s$  and  $s'$  is the maximum of two sub-costs (1) the navigation cost  $C_{nav}$ ; and (2) the 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 and tilt angles by their respective velocities, and computing the maximum of the pan and tilt costs.

## V. EXPERIMENTAL EVALUATION

We evaluated the proposed sampling-based view planning algorithm in simulation and on a physical robot. The simulation trials compared our algorithm with baseline algorithms in four different simulated office environments that vary in size and complexity (Section V-A). The hypothesis was that our algorithm would scale better than the baseline algorithms, especially in larger environments. In these trials, we used the expected reward of a plan as the performance measure. We also evaluated our algorithm on a mobile robot in the context

<sup>1</sup><http://metralabs.com>

<sup>2</sup><http://wiki.ros.org/navigation>

of object search in an office environment (Section V-B). In these trials, we used the fraction of planned views that were successfully executed, as the performance measure.

### A. Experimental Results: Simulation

Simulation trials were conducted in the robot simulator MORSE [27]. The size and complexity of the four environments used in these trials are provided in Tab. I. For each environment, 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.

First, we compared our algorithm (GVP) with other methods for the OP problem, in four environments, for different time limits  $T$ . As baselines, we considered a stochastic algorithm for the OP [21] (OP-S), and two sequential approaches that greedily select the  $m$  best views and sequence them using a TSP solver—one sequences the views greedily (TSP-G), whereas the other solves the TSP using dynamic programming (TSP-DP). We expect our approach to scale better in larger environments as it can choose locations freely from the initial distribution, unlike the TSP-based approaches that have to 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 and  $T = 120$ . Although OP-S achieves a comparable performance in smaller environments, its performance is much worse in larger environments. Similarly, TSP-G performs reasonably well for the small environments, but the results for the larger environments are poor, as hypothesized. TSP-DP provided good results in several trials, but it could not compute a solution in environments E1-E3 for longer intervals<sup>3</sup>. 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 over a range of environments and time limits.

### B. View Planning in the Real World

We also evaluated our approach on a mobile robot searching for target objects in a real-world office environment (Fig. 3). This evaluation was part of the STRANDS project that explore long-term autonomy of mobile robots in dynamic domains<sup>4</sup>. In this context, our algorithm has been successfully used within a long-term deployment (>60 days) in an office environment. Overall the robot performed 130 object search tasks (>2 per day). In each search task, the robot was required to search a small region of interest including surfaces of office desks, cabinet tops, meeting tables, and kitchen counters. To this end, it had to plan views and observe the

<sup>3</sup>Although the planning time is not considered within these experiments, the approach could not compute a solution on a standard laptop in under 10 minutes. The maximum number of locations the TSP-DP implementation was able to sequence was 21.

<sup>4</sup><http://www.strands-project.eu/>

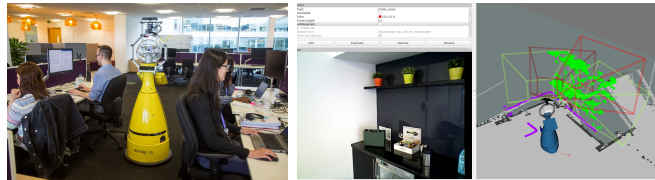


Fig. 3: View planning in a real-world office environment.

respective surfaces. On average, each search task included 3.65 views. The total number of views was 474. Tab. III describes the performance outcome of these views. More than 95% of the views (451 out of 474) were *successful* in that they were executed successfully by the robot, regardless of whether an object was perceived in the view or not. The remaining 23 (< 5% of 474) views were *unsuccessful* in that they could not be taken due to navigation failures. These results demonstrate that our view planning approach is applicable to robot platforms performing object search in realistic environments.

## VI. CONCLUSIONS

This paper has presented a randomized solution to the well known open problem of view planning for object search under time constraints. We have posed this problem as an OP with history-dependent rewards, and imposed a time limit. Experimental results show that our sampling-based view planning algorithm outperforms state of the art methods.

The proposed view planning approach has been tightly integrated with the representations and processing routines of a mobile robot platform. At the same time, the sampling-based algorithm itself does not depend on any kinematic structure, and it can (potentially) also be used to plan views of a camera mounted on a manipulator arm or on a quadcopter. Future work will further explore planning where both are limited planning time and execution time, and also investigate view planning under partial observability.

*Acknowledgments:* This work was supported by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No 600623, STRANDS; and the EPSRC grant EP/M015777/1, ALOOF.

## 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] Y. Ye and J. Tsotsos, "Sensor planning for 3d object search," *Computer Vision and Image Understanding*, vol. 73, no. 2, pp. 145–168, 1999.
- [3] S. Frintrop, *VOCUS: A visual attention system for object detection and goal-directed search*. Springer, 2006, vol. 3899.
- [4] A. Aydemir, A. Pronobis, M. Göbelbecker, and P. Jensfelt, "Active visual object search in unknown environments using uncertain semantics," *IEEE Trans. on Rob.*, vol. 29, no. 4, pp. 986–1002, Aug. 2013.
- [5] 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.
- [6] 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.
- [7] D. Joho, M. Senk, and W. Burgard, "Learning search heuristics for finding objects in structured environments," *Rob. Auton. Syst.*, vol. 59, no. 5, pp. 319–328, May 2011.
- [8] 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.

TABLE I: Experimental Environments

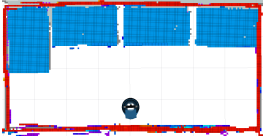
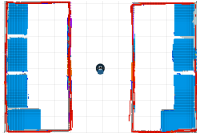
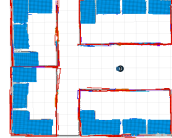
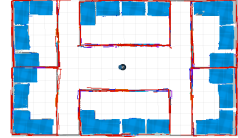
Environment	E1	E2	E3	E4
Size ( $m^2$ )	Small ( $32m^2$ )	Medium ( $96m^2$ )	Large ( $168m^2$ )	Huge ( $240m^2$ )
Coverage (%)	95%	95%	95%	95%
Locations ( $ S $ ) (unfiltered)	91 (322)	139 (402)	319 (1462)	399 (1533)
3D occ. grid size (#voxels) (only surfaces)	2362	4519	8775	12950
3D occupancy grid maps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

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

TABLE III: View planning performance in the real world.

Performance	Explanation	%	#views
Successful	Object(s) observed	37.77	179
	Nothing observed	57.38	272
Unsuccessful	Navigation failure	4.85	23
Total		100.00	474

- [9] J. Nunez-Varela and J. Wyatt, "Models of gaze control for manipulation tasks," *ACM Trans. on Applied Perception*, vol. 10, no. 4, p. 20, 2013.
- [10] J. Velez, G. Hemann, A. Huang, I. Posner, and N. Roy, "Planning to perceive: Exploiting mobility for robust object detection," in *International Conference on Automated Planning and Scheduling*, Freiburg, Germany, June 2011.
- [11] 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.

- [12] M. Hanheide, M. Gobelbecker, G. Horn, A. Pronobis, K. Sjo, 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.
- [13] 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.
- [14] B. Nilsson, "Guarding art galleries: Methods for mobile guards," Ph.D. dissertation, Lund University, 1995.
- [15] 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.
- [16] 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.
- [17] 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.
- [18] 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.
- [19] 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.
- [20] 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.
- [21] T. Tsiligirides, "Heuristic methods applied to orienteering," *The Journal of the Operational Research Society*, vol. 35, no. 9, pp. 797–809, 1984.
- [22] 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.
- [23] T. Wang, D. Lizotte, M. Bowling, and D. Schuurmans, "Bayesian sparse sampling for on-line reward optimization," in *Proceedings of the 22nd International Conference on Machine Learning*. ACM, 2005, pp. 956–963.
- [24] 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.
- [25] 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.
- [26] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013.
- [27] 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.