

Combining Search and Action for Mobile Robots

Geoffrey Hollinger, Dave Ferguson, Siddhartha Srinivasa, and Sanjiv Singh

Abstract—We explore the interconnection between search and action in the context of mobile robotics. The task of searching for an object and then performing some action with that object is important in many applications. Of particular interest to us is the idea of a robot assistant capable of performing worthwhile tasks around the home and office (e.g., fetching coffee, washing dirty dishes, etc.). We prove that some tasks allow for search and action to be completely decoupled and solved separately, while other tasks require the problems to be analyzed together. We complement our theoretical results with the design of a combined search/action approximation algorithm that draws on prior work in search. We show the effectiveness of our algorithm by comparing it to state-of-the-art solvers, and we give empirical evidence showing that search and action can be decoupled for some useful tasks. Finally, we demonstrate our algorithm on an autonomous mobile robot performing object search and delivery in an office environment.

I. INTRODUCTION

Think back to the last time you lost your car keys. While looking for them, a few considerations likely crossed your mind. There were certain places in the house where the keys were more likely to be, and these places were certain distances from each other. While planning your search for your keys, you might have thought to minimize distance traveled while ensuring that you hit all the likely places that the keys might be. What may not have crossed your mind was that after finding the keys, you needed to use them to start the car. Would it be advantageous for you to search the car first because then you could immediately drive away, or would it be advantageous to search the car last to avoid backtracking to the house? Consider a similar situation where you need to find a friend at a carnival just before leaving. Might it be advantageous to search near the entrance to allow a quick exit if the friend is found? Does the fact that the friend might be moving change the answer to these questions? What if there are multiple sets of keys or friends? We examine such questions in the context of mobile robotics.

State-of-the-art mobile robots are now capable of complex localization, manipulation, and planning. These capabilities are enabling robot assistants to perform useful tasks, such as cleaning, getting coffee, and setting tables [1]. Prior work in this area has considered the *action* aspect of these tasks. In other words, the robot is assumed to have at least an approximate estimate of the position of all objects of interest. However, assistance robots must typically *search* for the



Fig. 1. Mobile manipulator: Segway RMP200 base with Barrett WAM arm and hand. The robot uses a wrist camera to recognize objects and a SICK laser rangefinder to localize itself in the environment

objects (or targets) of interest. A concrete example is a robot that refreshes coffee in an office. This robot must find coffee mugs, wash them, refill them, and return them to office workers. It is unreasonable to assume that such a robot could maintain an accurate estimate of the locations of all mugs and people in the environment without needing to search.

Similarly, there is growing interest in developing urban search and rescue robots capable of finding survivors (and lost first responders) in disaster scenarios [2]. Work in this area typically deals with the *search* aspect of the task, but rarely is the actual rescue considered. What is ignored is that, in some cases, the specifics of the rescue (or *action*) might affect the search task. For instance, if certain areas of the building are particularly easy extraction points, it may be beneficial to search these areas first.¹ Such considerations have not seen a principled analysis in the literature.

In this paper, we analyze the combined *search/action* task in the context of mobile robotics. We begin with a survey of the work in probabilistic robotics, pursuit/evasion, and graph search (Section II), which highlights the lack of a unified treatment of search and action. We then formally define the search/action problem (Section III) and move to a theoretical analysis of its properties (Section IV). Based on our theoretical analysis, we present an approximation algorithm for search/action (Section V). We demonstrate our algorithm both in simulation and on a mobile robotic platform (Section VI). To the best of our knowledge, this paper is the first to explore the interconnection between search and action in mobile robotics.

¹A broad range of additional applications of our search/action formulation include autonomous Chemical Biological Radiological Nuclear (CBRN) threat identification and cleanup, mine inspection and repair, aged care, and wilderness rescue.

G. Hollinger and S. Singh are with The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA, gholling@ri.cmu.edu, ssingh@ri.cmu.edu

D. Ferguson and S. Srinivasa are with Intel Research Pittsburgh, Pittsburgh, PA 15213, USA, dave.ferguson@intel.com, siddhartha.srinivasa@intel.com

II. RELATED WORK

The problem of searching for an object has been heavily studied in both 2D and 3D environments. Ye and Tsotsos proved that the 3D search problem is NP-complete even for a single stationary object [3]. This complexity result motivates approximation algorithms in this domain. Sarmiento et al. presented an approximation algorithm for the 2D search problem [4], and Saidi et al. demonstrated a heuristic solution in 3D with a humanoid robot [5]. Neither of these algorithms consider the possibility of an action performed with the object after it has been found.

Parsons was one of the first to study the connection between search and the “pursuit/evasion” problem in which a team of searchers seeks to locate a moving, potentially adversarial target (or evader) [6]. The searchers’ aim is to move in such a way that the evader cannot escape. We refer to this as *guaranteed search* because the searchers move so as to guarantee finding the target. Guibas et al. examined the guaranteed search problem in a mobile robot workspace (rather than abstract graphs) [7]. They propose a visibility-based formulation that guarantees finding a target with a single searcher if such a path exists in a given environment.

The work mentioned above makes a worst-case assumption on the target’s behavior (i.e., that it moves arbitrarily fast and actively avoids being found). An alternative formulation is to assume that the target is non-adversarial. We refer to this as *efficient search* because the searchers move in such a way as to efficiently (in terms of time) find the target. This formulation is particularly appropriate for robotic assistance scenarios in which the objects and people in the environment are not actively avoiding the robot. The efficient search problem can be formulated as a Partially Observable Markov Decision Process (POMDP) for both stationary and moving targets [8]. The simpler MDP framework is insufficient because the state of the target is partially known. POMDPs maintain a belief estimate of the current state, and solving the POMDP yields a policy mapping from belief state to action that maximizes reward.

Large POMDPs are notoriously difficult to solve, but near-optimal algorithms have been proposed capable of solving POMDPs with thousands of states. Two state-of-the-art POMDP solvers are Heuristic Search Value Iteration (HSVI2) [9] and Successive Approximations of the Reachable Space under Optimal Policies (SARSOP) [10]. Both algorithms use point-based value iteration to progressively improve the policy. Unfortunately, the size of search/action POMDPs can grow well beyond the limitations of near-optimal solvers for representations of complex environments. This is particularly true with multiple searchers and/or multiple targets.

Research in assistive robotics typically does not consider searching for an object before performing an action on it [1], [11], [12]. These systems utilize computer vision to find objects, but they are given a coarse estimate of their position (e.g., the table on which it rests). Thus, they do not search on the scale of an office building or home. One notable

exception is the work of Roy et al. in which robots search for patients in a nursing home [8]. This work, along with work in urban search rescue [2], are examples of search for a moving target. However, this research falls in the previous category of pure search: they do not consider performing any action with the target after it is found.

In our prior work we proposed a bounded approximation algorithm for solving the efficient search problem [13]. Our method uses finite-horizon planning and implicit coordination between searchers to remain scalable in large environments with many searchers. We have shown that our approximation algorithm is competitive with near-optimal POMDP solvers for both moving and stationary targets. In the following sections, we extend these results to tasks requiring both search and action.

III. PROBLEM SETUP

In this section we formulate the combined search/action problem and show how it can be expressed as a POMDP. We start with a single searcher and single stationary target on a graph and then discuss extensions to multiple moving searchers and physical environments. Assume we are given a graph $G(N, E)$ with $|N|$ vertices and $|E|$ edges. A searcher exists at any time t at a vertex $s(t) = u \in N$, and the searcher can move to any vertex $s(t+1) = v \in N$ if an edge exists in E between u and v . Similarly, a target exists at some vertex $e(t) = u \in N$. A searcher at a given vertex can detect any target at the same vertex with some fixed probability $P(C|s(t) = e(t))$, where C is a detection (or capture) event. The searcher’s goal is to locate the target and perform some subsequent action with the target. We refer to the subsequent action as the *action task*, which is distinct from actions performed during search. The cost of the action task is dependent on the vertex at which the target was found. We can now define the searcher’s objective function $J(U)$ as in Equation 1. Note that the objective function depends on the term τ , which is the time to complete the combined search/action task (as defined below).

$$J(U) = \sum_{t=0}^T \gamma^t P(C|U(t) = e(t))P(U(t) = e(t)), \quad (1)$$

where $U = [U(1), \dots, U(T)]$ are the deterministic moves of the searcher specifying its location (i.e., $s(t) = U(t)$). The search/action time $\tau = t + \alpha(U(t))$, where $\alpha(U(t))$ is the expected time to complete the action task if the target is found at vertex $U(t)$. The discount factor $\gamma \in [0, 1]$ is a measure of the importance of finding the target quickly. The searcher’s goal is to choose $U(1), \dots, U(T)$ so as to maximize $J(U(1), \dots, U(T))$. The searcher must then complete the action task after finding the target.

For the search/action POMDP, the states are defined by the cross product of the searcher and target locations, the actions are the searchers’ movements on the graph, and the observation probabilities are defined by $P(C|s(t) = e(t))$. The initial belief is a potentially multi-modal estimate of the location of the target at the start of the search. To complete

the formulation, we add a set of states in which the searcher has control of the target. These states are reached once the searcher has captured the target, and their associated actions represent an arbitrary task. Reward is received after the action task is completed, and a negative reward can be added for each searcher step. The use of negative rewards allows a POMDP formulation with or without a discount factor.

To extend to a moving target, we can simply modify the transition probabilities to account for the target's motion model.² To extend to multiple searchers and/or targets, we can place K searchers on vertices $s_k(t)$ and M targets on vertices $e_m(t)$. The state and action space is now the cross product of the searchers' states and actions. Note that the state space increases exponentially with the number of searchers and targets.

Applying the search/action POMDP to a physical workspace requires representing the workspace as a discrete graph. This can be done using a regular grid or using rooms and hallways in indoor environments (see Figure 2) [13]. This formulation implicitly assumes that the searchers have a sensor capable of locating a target in the same cell and that the discretized cells (corresponding to vertices) be convex and sufficiently small.

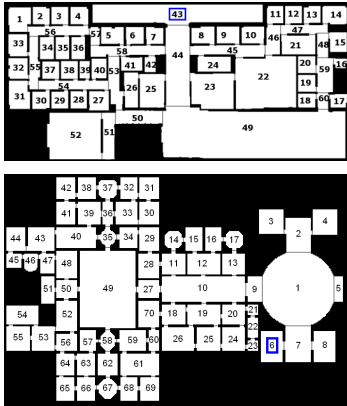


Fig. 2. Floorplans of environments used for search/action trials: office (top), museum (bottom). The searcher must find a target in one of the convex cells and then return the target to its starting cell. Starting cells are denoted with a blue box.

IV. THEORETICAL ANALYSIS

In this section we prove that the search tasks and action tasks can be decoupled for a single stationary target (e.g., a coffee mug) if an undiscounted reward metric is used. Conversely, we show that this is not the case for multiple targets, moving targets, and/or when using discounted reward. Moving targets of interest include patients in a nursing home and survivors in a search and rescue scenario.

A. Single Stationary Target

For a single searcher and stationary target located in a workspace, the searcher must find the target and then perform some subsequent action on the target. The cost of

²Note that this requires that the target's motion model be Markovian.

the subsequent action is dependent on the location at which the target was found. The searcher's goal is to find the target in the workspace and then complete the subsequent action. Theorem 1 shows that the searcher can achieve the optimal strategy for the search/action task by solving the search and action tasks separately.

Theorem 1: Adding a subsequent action cost does not change the optimal search strategy w.r.t. expected cost for a single stationary target.

Proof: Let $|N| = 2$, and label these nodes n_1, n_2 . The probability of the target being at each node is p_1, p_2 . A fixed (or expected) action cost is associated with each node, a_1, a_2 . A cost to reach each node from the start is denoted as d_1, d_2 , and a cost between them is denoted as d_{12} . The searcher can choose to visit n_1 first or to visit n_2 first. We examine the expected cost for these two cases.

For search:

$$\text{Go to } n_1 \text{ first: } C_1 = p_1 d_1 + p_2 (d_1 + d_{12})$$

$$\text{Go to } n_2 \text{ first: } C_2 = p_2 d_2 + p_1 (d_2 + d_{12})$$

For search/action:

$$\text{Go to } n_1 \text{ first: } C_{a1} = p_1 (d_1 + a_1) + p_2 (d_1 + d_{12} + a_2)$$

$$\text{Go to } n_2 \text{ first: } C_{a2} = p_2 (d_2 + a_2) + p_1 (d_2 + d_{12} + a_1)$$

Subtracting we get:

$$C_{a1} - C_1 = p_1 a_1 + p_2 a_2$$

We also see that:

$$C_{a2} - C_2 = p_2 a_2 + p_1 a_1$$

Therefore:

$$C_{a2} - C_2 = C_{a1} - C_1$$

Generalizing to more than two possible locations, we see that any search strategy S and corresponding search/action strategy A visiting N locations in the sequence $\{1, \dots, N\}$ will take the following form:

$$C_S = p_1 d_1 + \dots + p_N (d_1 + d_{12} + \dots + d_{N-1, N})$$

$$C_A = p_1 (d_1 + a_1) + \dots + p_N (d_1 + d_{12} + \dots + d_{N-1, N} + a_N)$$

$$C_A - C_S = \sum_{i=1}^N p_i a_i \quad (2)$$

The expected cost of completing the subsequent action is given by Equation 2 and is independent of the search strategy. ■

Theorem 1 applies to the *undiscounted* POMDP formulation by using the shortest path distances between nodes u and v as the d_{uv} . Thus, we can solve the search POMDP and action task (PO)MDP sequentially and still find the optimal policy. This is a potentially large gain because it avoids a large expansion of the state space. Additionally, this theorem demonstrates that the subsequent action cost has no effect on the optimal search strategy. This means that knowledge of the subsequent action is unnecessary during search.

One important caveat, however, is that Theorem 1 is only true if the POMDP reward is not discounted. Adding a discount factor modifies the reward in such a way that the expected subsequent action cost is dependent on the search strategy. The intuition behind this is that the discount factor allows the search strategy to modify the subsequent action cost of a location by adjusting the time at which it is visited.

This difference will become clearer when examining the multiple target and moving target cases.

B. Multiple Stationary Targets

Now let there be multiple stationary targets that a searcher must find before completing the subsequent action task. An example of this would be if a service robot needed to collect several mugs and then take them to the dishwasher. Also assume that the searcher can carry more than one target, so it may continue to search after locating target. In this case Theorem 1 does not hold. To see why, examine the expected reward for the search/action task with two targets. Let p_{ij} be the probability of finding target i in location j .

For search:

$$C_1 = p_{11}p_{21}d_1 + (p_{12}p_{22} + p_{11}p_{22} + p_{21}p_{12})(d_1 + d_{12})$$

$$C_2 = p_{12}p_{22}d_2 + (p_{11}p_{21} + p_{12}p_{21} + p_{22}p_{11})(d_2 + d_{12})$$

For search/action:

$$C_{a1} = p_{11}p_{21}(d_1 + a_1) + (p_{12}p_{22} + p_{11}p_{22} + p_{21}p_{12})(d_1 + d_{12} + a_2)$$

$$C_{a2} = p_{12}p_{22}(d_2 + a_2) + (p_{11}p_{21} + p_{12}p_{21} + p_{22}p_{11})(d_2 + d_{12} + a_1)$$

Subtracting, we see that the ratio $C_{a2} - C_2 \neq C_{a1} - C_1$ for the multiple target case. Thus, the subsequent action cost can modify the optimal search strategy. Intuitively, the reason for this is that in the multiple target case, the search strategy can affect the likelihood of finishing the search at a given location. The search is most likely to end at the last point visited because targets have been gathered from all other points. If a location has a very low subsequent action cost, leaving it for last may be a desirable search/action strategy even if it is not a desirable search strategy.

C. Moving Targets

For the moving target case, Theorem 1 also does not hold. The probability of a moving target being at a given location is dependent on when that point is visited (i.e., p_i is dependent on t yielding $p_i(t)$). The dependence of the probability of capture on time breaks the equality so that $C_{a2} - C_2 \neq C_{a1} - C_1$. For example, a target may be moving towards a location with a very low subsequent action cost. It may be advantageous for the searcher to move to that location in the search/action case even if the probability of the target being there is low compared to alternative locations.

V. ALGORITHM DESIGN

We can combine our theoretical analysis with prior work in search to design an approximation algorithm for the search/action problem. For a single stationary target with undiscounted reward, we have shown (Theorem 1) that the search and subsequent action can be solved separately. This allows for search approximation algorithms to be used without modification in this case.

In contrast, for discounted reward cases, it may be advantageous to consider the subsequent action cost as in Equation 1. The finite-horizon path enumeration (FHPE) algorithm has been shown to provide high-quality, scalable results for discounted search POMDPs [13]. The algorithm examines

all possible paths to a fixed horizon and then replans with a the receding horizon. FHPE is particularly well-suited to problems where maximizing short-term reward does not conflict with maximizing long-term reward. This is often the case during search tasks because searchers will usually search nearby locations before searching further away.

Algorithm 1 shows an application of FHPE to the discounted search/action problem. Depending on the exact instance, the reward function $J(U)$ can be substituted as appropriate. For both moving and stationary targets, the reward function is the time-truncated version of Equation 1, which includes the cost of the subsequent action.

Algorithm 1 Finite-horizon path enumeration

Input: Single-searcher search/action problem

for All paths U to horizon d **do**

Calculate $J(U)$

end for

$U \leftarrow \arg \max_U J(U)$

while Target not found **do**

Execute U replanning as needed

end while

Perform action task with found target

A. Multiple Searchers

The finite-horizon approximation algorithm can be extended to multiple searchers using implicit coordination. The multi-searcher reward function is shown in Equation 3.

$$J(U) = \sum_{t=0}^T \gamma^t P(C|\exists k : U_k(t) = e(t))P(\exists k : U_k(t) = e(t)), \quad (3)$$

where $U(t) = [U_1(t), \dots, U_K(t)]$ is now a K -dimensional control vector specifying the location of each searcher (i.e., $s_k(t) = U_k(t)$). The search/action time $\tau = t + \alpha(U_1(t), \dots, U_K(t))$, where $\alpha(U_1(t), \dots, U_K(t))$ is the expected time to complete the action task if the target is found on each of vertices $U_1(t), \dots, U_K(t)$ weighted by the probability that the target is at each vertex.

Algorithm 2 gives an implicit coordination algorithm for the search/action problem. The searchers sequentially allocate their paths through the environment, and then simultaneously execute these paths. The algorithms run on the time-unfolded graph N' , which allows searchers to revisit locations and for more than one searcher to be in the same location simultaneously. Sequential allocation provides linear scalability in the number of searchers, and gives a constant factor approximation guarantee for nondecreasing, submodular objective functions [13]. The search/action objective function $J(U)$ is both nondecreasing and submodular (proof is omitted but is a straightforward corollary of prior work [13]), which leads to a bounded approximation algorithm for the multi-searcher search/action problem.

Algorithm 2 Implicit coordination

```
Input: Multi-searcher search/action problem
%  $V \subseteq N'$  is the set of nodes visited by searchers
% A node in  $N'$  is a time-stamped node of  $N$ 
 $V \leftarrow \emptyset$ 
for all searchers  $k$  do
  %  $U_k \subset N'$  is a feasible path for searcher  $k$ 
  % Finding this  $\arg \max$  solves the search/action for
  % searcher  $k$ 
   $U_k \leftarrow \arg \max_{U_k} J(V \cup U_k)$ 
   $V \leftarrow V \cup U_k$ 
end for
while target not found do
  Execute  $U_k$  for all searchers  $k \in K$  replanning as
  needed
end while
Perform action task with found target
```

VI. EXPERIMENTAL RESULTS

A. Simulated Environments

To test our search/action approximation algorithm, we set up a simulated scenario requiring both a search and subsequent action. One or more searchers move around an indoor environment with omnidirectional line-of-sight sensors. The searchers must locate a stationary or moving target and then take the target back to the starting location. The searchers use a *discounted* reward metric, which means that Theorem 1 does not apply in either the stationary or moving cases.

We ran simulations in the two environments shown in Figure 2. The office environment has two major cycles corresponding to the hallways, and the museum is a highly connected graph with many cycles. The starting location of the target is initialized randomly, and the target's movement (if applicable) is a random walk. Our simulator runs in C++ in Linux on a 3.0 GHz P4 with 2 GB RAM.

We compared our approximation algorithm to two state-of-the-art POMDP solvers: HSVI2 and SARSOP. Since these solvers are optimized with discounted reward in mind, we limit our experiments to this case (with arbitrarily set $\gamma = 0.95$). The POMDP solvers were given two minutes of solving time for each instance. At this point the bounds on solution quality were not improving. Since FHPE runs online with a receding horizon, it did not need to precompute a solution. Table I shows that FHPE provides solutions competitive with the POMDP solvers in these environments. It is important to note that the number of states in these POMDPs (approximately 5000) is at the frontier of what general POMDP solvers can handle. FHPE, on the other hand, is scalable to much larger environments.

Table I also compares two versions of FHPE applied to the search/action problem. The first version decouples search and action by excluding the subsequent action costs in the calculation of the search strategy. The full version of FHPE includes subsequent action costs as in Equation 1. The results

show that, in these environments, taking into account the subsequent action cost yields only a small improvement in the final reward. This is somewhat contrary to our theoretical analysis in Section IV, which shows how search and action *cannot* be provably decoupled with a discounted reward metric. However, the results show that one can “get away” with decoupling search and action in these instances. This is likely due to the relatively small differences between the subsequent action costs at each location in these scenarios.

We also ran trials with multiple searchers using FHPE+SA. The searchers move through the environment until the target is found, and the searcher that locates the target then takes it back to the starting cell. Figure 3 shows the results from the multi-searcher trials. The linear scalability of FHPE+SA easily handles five searchers in these environments. Neither of the general POMDP solvers were able to even fit the two searcher problem (343,000 states) in memory.

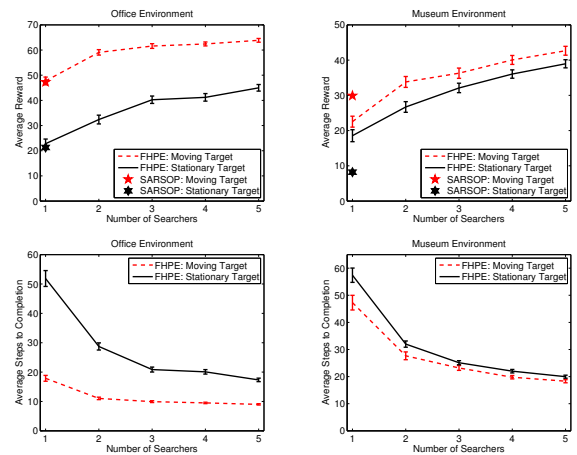


Fig. 3. SEARCH/ACTION results using FHPE and implicit coordination with multiple searchers. Averages are over 200 trials, and error bars are one SEM. SARSOP, a general POMDP solver, was unable to fit the two searcher instance in memory. Top graphs show average reward (higher is better), and bottom graphs show average steps to complete the search/action task (lower is better).

B. Implementation on Mobile Robot

We ran search/action tests on a mobile manipulator platform consisting of a Barrett anthropomorphic arm and hand mounted on a Segway mobile base. The platform localizes itself using AMCL Monte-Carlo sampling with a laser rangefinder and runs using the OpenRave [14] and Player [15] software. It carries a miniature camera, which it uses to identify coffee mugs for grasping with the arm/hand [1]. The system is shown in Figure 1.

The robot was given three waypoints in the environment that may contain mugs. The optimal path between the waypoints (w.r.t. navigation cost) was computed using an exhaustive search. Since the navigation cost is undiscounted, Theorem 1 applies, and we only need to consider the search costs. The robot then proceeded to search these waypoints by moving to them and scanning them with the camera. Upon finding a mug, the robot would pick it up with the arm/hand and take it back to the sink. The path of the robot through

TABLE I

SEARCH/ACTION REWARD COMPARISON (HIGHER IS BETTER) OF HSVI2, SARSOP, AND FHPE (WITH AND WITHOUT INCLUDING ACTION COST) FOR A SINGLE SEARCHER IN TWO ENVIRONMENTS. AVERAGES ARE OVER 200 TRIALS, AND ERRORS ARE ONE SEM.

Environment (target)	HSVI2 [9]	SARSOP [10]	FHPE (w/o action cost)	FHPE (with action cost)
Office (stationary)	17.3 \pm 3.3	21.3 \pm 3.3	21.3 \pm 1.9	22.8 \pm 1.8
Museum (stationary)	14.2 \pm 2.5	8.2 \pm 3.0	17.5 \pm 1.5	18.5 \pm 1.7
Office (moving)	46.2 \pm 3.8	47.3 \pm 2.7	47.2 \pm 1.6	47.7 \pm 1.5
Museum (moving)	27.5 \pm 3.6	29.9 \pm 3.4	21.6 \pm 1.5	22.5 \pm 1.5

the office in an example trial is given in Figure 4. The video attachment to this paper shows the robot successfully finding a coffee mug and placing it in the sink.

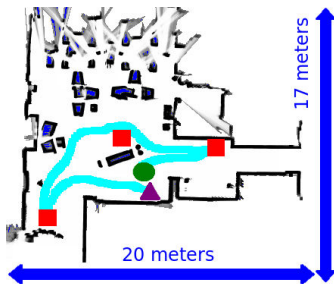


Fig. 4. Map of a kitchen area within the Intel labs used for implementation of search/action on mobile manipulator. Circle shows starting robot location, squares show possible locations of coffee mugs, and triangle shows the location to which the robot must move the mug. An example robot path is shown in cyan (light grey); the mug was found in the bottom left location in this trial.

VII. CONCLUSIONS AND FUTURE WORK

Our results have opened the door to a rich study of the connection between search and action in mobile robotics. We have proved that the undiscounted search/action problem with a single stationary target can be solved by considering the search and action components separately. For discounted reward, multiple targets, and moving targets, on the other hand, it can be beneficial to consider the action component before performing the search. Drawing on this theoretical analysis, we have designed approximation algorithms for the search/action task with both stationary and moving targets.

We have demonstrated the performance of our algorithm with both simulated analysis and on a physical mobile manipulator. Our simulated results show that solving search and action separately has only a small affect on solution quality if the action costs are not highly disparate across the environment. This is often the case in search and retrieval tasks. We have also shown that our approximation algorithm is competitive with general POMDP solvers for moderately sized problem instances. General POMDP solvers quickly grow intractable with multiple agents and in large environments. In contrast, our approximation algorithm uses a receding-horizon technique to remain scalable in large environments, and we utilize implicit coordination to achieve linear scalability in the number of searchers.

Short-term future work includes testing of our approximation algorithm with multiple objects and a more extensive

analysis of environments and tasks in which search and action can be decoupled. In addition, we are interested in developing a framework for ordering and solving multiple queries. We hope to extend the approach presented here to cope with such scheduling and thereby enable our system to provide continuous assistance for daily living.

VIII. ACKNOWLEDGMENTS

We gratefully acknowledge Martin Rufi from ETH Zurich, Casey Helfrich from Intel Research Pittsburgh, and Mike Vande Weghe from CMU for assistance with the mobile robot platform.

REFERENCES

- [1] S. Srinivasa, D. Ferguson, M. Vande Weghe, R. Diankov, D. Berenson, C. Helfrich, and H. Strasdat, "The Robotic Busboy: Steps Towards Developing a Mobile Robotic Home Assistant," in *Int'l Conf. on Intelligent Autonomous Systems*, 2008.
- [2] V. Kumar, D. Rus, and S. Singh, "Robot and Sensor Networks for First Responders," *Pervasive Computing*, pp. 24–33, 2004.
- [3] Y. Ye and J. Tsotsos, "Sensor Planning in 3D Object Search: its formulation and complexity," in *4th Int'l Symp. on Artificial Intelligence and Mathematics*, 1996.
- [4] A. Sarmiento, R. Murrieta-Cid, and S. Hutchinson, "A Multi-Robot Strategy for Rapidly Searching a Polygonal Environment," in *Proc. 9th Ibero-American Conference on Artificial Intelligence*, 2004.
- [5] F. Saidi, O. Stasse, and K. Yokoi, "Active Visual Search by a Humanoid Robot," *Recent Progress in Robotics: Viable Robotic Service to Human*, vol. 370, pp. 171–184, 2007.
- [6] T. Parsons, "Pursuit-Evasion in a Graph," in *Theory and Applications of Graphs*, Y. Alavi and D. Lick, Eds. Springer, 1976, pp. 426–441.
- [7] L. Guibas, J. Latombe, S. LaValle, D. Lin, and R. Motwani, "Visibility-Based Pursuit-Evasion in a Polygonal Environment," *Int'l Journal of Comp. Geometry and Applications*, vol. 9, no. 5, pp. 471–494, 1999.
- [8] N. Roy, G. Gordon, and S. Thrun, "Planning under Uncertainty for Reliable Health Care Robotics," in *Proc. Int'l Conf. on Field and Service Robotics*, 2003.
- [9] T. Smith, "Probabilistic Planning for Robotic Exploration," Ph.D. dissertation, Carnegie Mellon University, 2007.
- [10] H. Kurniawati, D. Hsu, and W. S. Lee, "SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces," in *Proc. Robotics: Science and Systems Conf.*, 2008.
- [11] H. Nguyen, C. Anderson, A. Trevor, A. Jain, Z. Xu, and C. Kempe, "El-e: An Assitive Robot that Fetches Objects from Flat Surfaces," in *Proc. Human Robot Interaction, The Robotics Helpers Workshop*, 2008.
- [12] M. Quigley, E. Berger, and A. Ng, "STAIR: Hardware and Software Architecture," in *Proc. AAAI Robotics Workshop*, 2007.
- [13] G. Hollinger and S. Singh, "Proofs and Experiments in Scalable, Near-Optimal Search with Multiple Robots," in *Proc. Robotics: Science and Systems Conf.*, 2008.
- [14] R. Diankov and J. Kuffner, "OpenRAVE: A Planning Architecture for Autonomous Robotics," Robotics Institute, Carnegie Mellon University, CMU-RI-TR-08-34, Tech. Rep., 2008.
- [15] B. Gerkey, R. Vaughan, and A. Howard, "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems," in *Proc. Int'l Conf. on Advanced Robotics*, 2003, pp. 317–323.