APPEARED IN
IEEE/RSJ Int. Conf. on Intelligent Robots & Systems, 2016

Act to See and See to Act: POMDP Planning for Objects Search in Clutter

Jue Kun Li David Hsu Wee Sun Lee

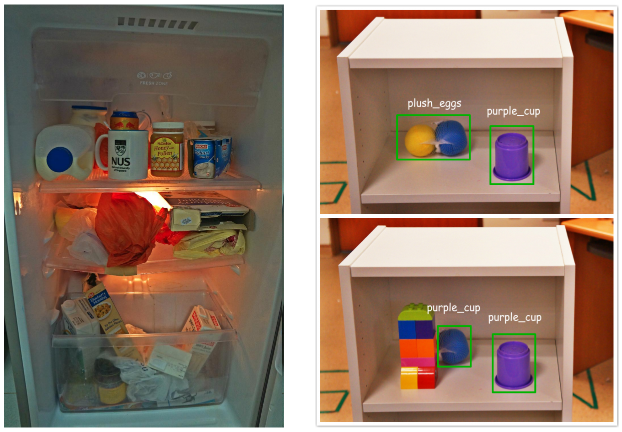
***Abstract*— We study the problem of objects search in clutter. In cluttered environments, partial occlusion among objects prevents vision systems from correctly recognizing objects. Hence, the agent needs to move objects around to gather information, which helps reduce uncertainty in perception. At the same time, the agent needs to minimize the efforts of moving objects to reduce the time required to complete the task. We model the problem as a Partially Observable Markov Decision Process (POMDP), formulating it as a problem of optimal decision making under uncertainty. By exploiting spatial constraints, we are able to adapt online POMDP planners to handle objects search problems with large state space and action space. Experiments show that the POMDP solution outperforms greedy approaches, especially in cases where multi-step manipulation is required.**

I. Introduction

Searching for a specific object is a common practice that humans carry out almost everyday and everywhere. As workers, we speedily search for items on the jam-packed shelves in warehouses. As cooks, we attentively search for the right ingredient on the cupboard to prepare a delicious meal. As researchers, we eagerly search for the Red Bull in a cluttered refrigerator in order to survive the paper deadline. Let us examine the last example in more details. Upon opening the refrigerator, we are greeted with a cluttered scene (shown in Fig. 1a) containing jars, bottles, boxes, etc. We find that right behind the white cup with the NUS logo lies a partially occluded object, whose visible part resembles that of a Red Bull can. However, we are not completely confident about its true type. We naturally adopt the strategy of *Act to See and See to Act*. We decide to move away the cup in front to reduce occlusion (*Act to See*), which in turn reveals more information of the scene for effective object manipulation (*See to Act*).

In this paper, we deploy this search strategy in robotic systems, taking a further step towards fully autonomous and intelligent service robots. Here, we consider the problem of objects search in clutter as shown in Fig. 1b. The robot has a fixed camera view of the shelf. The goal is to find and retrieve the target object that might be partially occluded. The prevalence of uncertainty in cluttered environment makes search tasks insurmountable for the cutting-edge robotics systems. In such cases, robots have incomplete knowledge about the environment due to noisy sensors and occlusion. For example, if the plush eggs are fully visible, the object detector is able to correctly report its true type. However, once it is occluded, it will be mistakenly classified as the

The authors are with School of Computing, National University of Singapore, Singapore. {juekun,dyhsu,leews}@comp.nus.edu.sg



(a) A cluttered refrigerator. (b) Uncertainty due to occlusion.

Fig. 1: Objects search in clutter.

purple cup. Following the idea of *Act to See and See to Act*, the robot needs to plan a sequence of actions to rearrange objects in order to reduce perception uncertainty and find the target object in the most economical manner. More specifically, at each step, the robot needs to decide which objects to move to gather information and where to place that object such that the expected reward over the entire sequence of actions is maximized.

Planning under perception uncertainty can be modeled as a Partially Observable Markov Decision Process (POMDP). POMDPs provide a principled and general planning and decision-making framework for acting optimally in partially observable domains. Although POMDP is a powerful modeling tool, the curse of dimensionality and the curse of history have prevented it from being widely applied in robotic tasks. Robotic tasks often involve a large number of states. Since the states are partially observable, robots must reason over beliefs, which are probability distribution over the states. With the increase in planning horizon, the number of action-observation histories, which should be taken into account during planning, grows exponentially. Fortunately, state-of-the-art POMDP solvers can now scale to large state spaces and reasonable planning horizons. In this work, we use an approximate online POMDP algorithm, DESPOT [1], that searches a sampled search tree at each time step. While DESPOT is able to handle large state spaces, the inherent large action space of the problem still hinders the efficient tree search. The key insight in this paper is that the number of feasible actions can be largely reduced by taking account of spatial constraints, which makes efficient search in DESPOT possible. For example, collision among objects should be avoided upon taking an action.

We implemented the proposed POMDP model and tested it in eight non-trivial scenarios in simulation to demonstrate the advantages of online POMDP planning over both greedy algorithms using heuristic and intuitive human strategy. Experiments show that the POMDP planner is able to search for objects effectively in a cluttered and confined environment. POMDP planning has been applied to similar robotic manipulation tasks before [2], but our work is able to handle complex problems of much larger scale in state space and action space.

This paper makes two main contributions. First, we propose a POMDP model for objects search problem in cluttered environment with visual occlusion and perception uncertainty; second, by exploiting spatial constraints of the problem, we design a general and effective heuristic to facilitate the search process.

II. Related Work

Manipulation planning among movable objects has been studied extensively in [3], [4], [5]. The problem assumes that the poses of all objects are known. In reality, the poses and the attributes (e.g., colors, shapes, etc.) of objects cannot exactly be known initially due to occlusion and noisy sensory feedback. Hence, the strategy of *Act to See and See to Act* becomes crucial to gather information of objects to reduce uncertainty, which in turn helps accomplish the task.

With the recent advancement in sophisticated robotic manipulators, manipulation-aided objects search has gained its popularity in the community. Manipulation-aided objects search allows the agent to explore unknown regions of the space, which are not directly accessible to its sensors. In [6], the problem of finding a fully occluded object in a container is considered. In their work, they exploit co-occurrence relationships among similar objects and spatial constraints of the environment to facilitates the search process. However, they do not try to find the target object using the minimum amount of effort (time spent, number of objects to be moved). For optimal search, [7] proposes a connected components algorithm for objects search in clutter and show that it is optimal under all conditions. However, the worst case complexity of the algorithm is $\mathcal{O}(n^2 2^n)$. They also assume that the target is the only hidden object in the scene, which might not be true in general. To remove this assumption, [8] proposes an adaptive lookahead exploration algorithm that guarantees complete exploration of the environment. At each step, the Adaptive Horizon Exploration algorithm chooses an action that maximizes the information gain of the search with a given planning horizon.

All aforementioned pieces of work consider the problems of objects search when the target is fully occluded. An overlooked fact is that more often than not we can roughly recognize the target object based on its partial view. However, additional information about the objects does not make the problem any easier algorithmically because partial information naturally introduces another source of uncertainty. As the example in Section I shows, the recognition system might misclassify the object based on the partial view. Joni et al.[2] model manipulation of multiple objects in clutter as a POMDP, taking into account observation uncertainty due to partial occlusion and action uncertainty. They use the example of cleaning dirty cups on the table to demonstrate the clear advantage of multi-step POMDP planning over simple greedy approaches. However, it is not obvious that their algorithm can handle complex problems with large state and action spaces. In the example of searching for objects in a refrigerator, it is preferable to move objects inside the confined space rather than directly moving them out when comparing their respective cost (time to move, consequences of dropping objects due to unstable grasps, etc.). A natural question that comes next is where exactly to place the object inside the space. The problem of placement requires the model to explicitly represent object poses in certain coordinate system, which easily blows up the state space and the action space.

In this work, we try to close the gaps by designing a POMDP model for objects search with large state space and action space. Specifically, the state space is at the scale of 10^{32}, with more than 1000 number of possible actions. In addition, objects placement is traditionally considered as an independent research question [9], [10]. This work makes the attempt to couple objects placement problem with objects search. By exploiting spatial constraints of the problem, we are able to demonstrate the superiority of POMDP planning over greedy approaches.

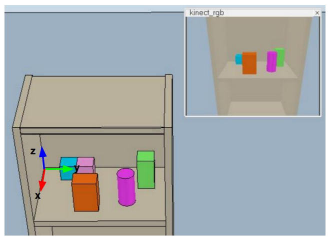
III. POMDP Planning for Objects Search in Clutter

POMDPs provide a principled and general planning and decision-making framework for acting optimally in partially observable stochastic domains. This makes POMDPs a suitable and powerful tool to model real-world robotic tasks where inherent uncertainties are prevalent. The state-of-the-art online POMDP solvers are POMCP [11] and DESPOT [1]. In this work, we opt for DESPOT, because it has a much stronger worst-case performance bound. In [12], DESPOT was successfully applied to real-time autonomous driving among many pedestrians.

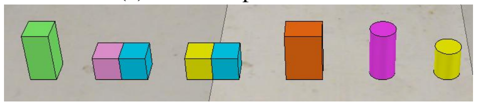
A. An Overview

Searching for objects in a cluttered environment is an essential ability that intelligent robots should possess. Under observation uncertainty, a robot often needs to plan a sequence of actions to move objects inside or outside a confined space to reveal the target that could potentially be occluded. At the same time, certain objectives need to be met: to find the target accurately while minimizing the execution time. Modeling the problem of objects search in clutter as a POMDP helps address the issue of uncertain observation and complex objectives jointly.

Formally, a POMDP is defined by the tuple $< S, A, Z, T, O, R, \gamma >$, where S, A and Z denote the system's state space, action space and observation space respectively. T is the transition function $T(s,a,s') = p(s'|s,a)$ that models



(a) An example scene.



(b) Six types of objects.

Fig. 2: Simulated environment.

the effect of taking an action $a \in A$ in the current state $s \in S$. O is the observation function $O(s', a, z) = p(z|s', a)$ that models noisy sensor observations. An immediate reward $R(s, a)$ is obtained upon taking action a in state s. In partially observable domains, planning is performed on a belief b_t, which denotes the probability distribution over possible states at time t. Bayes' rule is applied to update the belief after taking action a_t and receiving observation z_t. In this work, we represent a belief as a finite set of particles or weighted state instances in order to deal with large state space.

The solution of POMDP planning is presented in the form of a policy π, which maps a belief b to an action a. During online POMDP planning, the goal is to find a policy to maximize the expected total reward at the current belief b:

$$V_\pi = E(\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(b_t))|b_0 = b) \quad (1)$$

where b_0 is the initial belief, and $\gamma \in (0, 1)$ is a discount factor, which places preferences for immediate rewards over future rewards. γ is set to 0.95 in this work.

We consider a general problem of objects search in clutter. In a cluttered and confined environment, there are typically several objects of different attributes. The objective is to find one target object correctly and efficiently (though there could be multiple objects of the same attributes). Fig. 2a shows one example scene. From this IKEA shelf, the goal is to find out the pink-blue object that is partially occluded by the orange object. Our POMDP model captures the information of all objects as the state. The agent may move objects inside or outside the shelf, which constitutes the action space. The Kinect sensor provides RGBD image streams to the agent to perceive the scene, from which the essential information, such as object poses and type, is extracted as the observation. We model the uncertainty in object types due to occlusion in the observation function. Finally, the POMDP model encodes the objectives of the task in the reward function. Next, we begin by formalizing a POMDP for objects search in clutter, and then elaborate the modified DESPOT algorithm for fast online planning.

B. A POMDP Model for Objects Search in Clutter

1) Scene Modeling: A state s is a concise description of the scene composition. Specifically, $s = \{Obj_i | i \in [1, MAX]\}$. Obj_i is the attributes for the ith object inside the shelf, which comprises its position (x, y) whose origin is shown in Fig. 2a, orientation θ, and type t. Due to the limited space inside the shelf, we put a constraint on the maximum number of objects that could be concurrently present in the shelf. For the IKEA shelf of real scale, it is reasonable to set MAX to be 8. Six different types of objects are considered as shown in Fig. 2b. For simplicity, they are distinguishable by color only, but other object attributes (e.g., shape, affordance, etc.) can be easily incorporated into this POMDP model, which is explained in next section. Since all objects sit on shelf surface of the same level, only 2D positions and orientation in z axis are considered. Positions are further discretized into 12 in x direction and 17 in y direction with resolution of $2cm$. Orientations are discretized into 10 levels with resolution of $36°$. At a rough estimate, the state space is at the scale of 10^{32}, which is horrendously large.

2) Action Modeling: There are four types of actions for each object present: MOVE_INSIDE(i, x, y): move the ith object to (x, y) on the 12×17 2D plane; MOVE_OUTSIDE(i): move the ith object out of the shelf if there is no space inside the shelf; DECLARE_FIND(i): this action declares the ith object is the target and move it to a designated location; DECLARE_NONE: this action declares there is no instance of the target object type in the shelf. Both declaration actions terminate the planning. Transition function is assumed to be deterministic. The size of the action space is over 1000.

3) Sensor Modeling: An observation, $z = \{Cluster_i | i \in [1, MAX]\}$, is a set containing observed attributes for each cluster. Clusters are obtained by performing segmentation on point clouds from a Kinect sensor. We assume there is a minimum separation of $3cm$ between objects to ensure correct segmentation initially. We also assume that the number of clusters correctly reflects the number of objects on the shelf. $Cluster_i$ is composed of its estimated positions (x', y') and hypothesized object type t'. We collect data in simulation, and learn a decision tree regressor to estimate positions of each cluster, based on which the positions of particles in the belief are updated. We also learn a color-histogram classifier using SVM that outputs a probability distribution over object types, from which we sample a type as the hypothesized type of this cluster. Detailed implementation on how to convert raw Kinect input to an observation is provided in IV-A.

The observation function $O(s', a, z)$ captures the uncertainty of object types due to occlusion. The intuition is that the more an object is occluded, the less likely the object detector will report its type correctly. Upon taking an action a, a new state s' is encountered. s' captures the spatial relationship among objects, from which occlusion ratio $occl$ on $Cluster_i$ can be calculated. Concretely, each type of objects is associated with a 3D bounding box, specifying its length, width and height. Given the poses and the types of all objects in state s', their 3D bounding boxes are first projected onto the back of the shelf (y-z plane), which results in sets of 2D rectangles $R = \{Rect_k\}$. The occlusion ratio $occl_i$ of Obj_i is defined as the ratio between the area of intersection with other rectangles and the area of the projected rectangle of Obj_i. Formally,

$$occl_i = \frac{Area(\bigcup_{Rect_j \in R \setminus \{Rect_i\}} Rect_i \bigcap Rect_j)}{Area(Rect_i)} \quad (2)$$

The occlusion ratio is further discretized into 10 levels. Assuming independent observation among objects, the observation function can be rewritten as,

$$\begin{aligned} O(s',a,z) &= p(z|s',a) \\ &= \prod_{i=1}^{MAX} p(Cluster_i|Obj_i, occl_i) \\ &= \prod_{i=1}^{MAX} p(Cluster_i.t'|Obj_i.t, Obj_i.\theta, occl_i) \end{aligned} \quad (3)$$

In the processing, we assume that all features are estimated accurately except the type which is noisy. We learn the probability distribution function $p(t'|t,\theta,occl)$ from simulations using a color-histogram classifier and store it as a table, which can be looked up during planning. One technical issue in the observation function is to ensure that $Cluster_i$ and Obj_i refer to the same entity. We treat it as an Assignment problem and employ the Kuhn-Munkres algorithm [13] to find the correspondence among clusters from two consecutive observations such that the cost (Euclidean distance) of the assignment is minimized.

4) Reward Modeling: The reward function is designed to align with the objectives of the task: to find out the target object by manipulating objects safely and efficiently.

- For moving an object that either does not exist or is occluded, the agent receives a large penalty of -10000. This ensures safety of objects manipulation.
- MOVE_INSIDE: if moving an object to a new position results in collision with other objects, the agent receives a large penalty of -10000; if the goal position is the same as the current position of an object to be moved, the agent receives a penalty of -20; otherwise, the agent incurs a small movement cost that is proportional to the distance traveled: $abs(y-y') + abs(2*L-x-x')$, where (x,y) and (x',y') are the start and the goal positions, and L is the length of the shelf along x axis. The first part of it is the cost along y direction, while the second part takes into account that the agent moves the object in a U-shape fashion rather than a straight line path in order to avoid collision with other objects during transition.
- MOVE_OUTSIDE: the agent receives a cost of -200. We discourage to move objects outside because it is more time consuming and the chances of dropping or damaging objects due to unstable grasps is higher.
- DECLARE_FIND: if the declared object is not of the target type, the agent receives a large penalty of -10000; otherwise it receives a reward of 10.
- DECLARE_NONE: if there is at least one target object on the shelf, the agent receives a penalty of -250; otherwise it receives a reward of 10.

C. Enhanced DESPOT Search

Online POMDP planning searches a belief tree for the best action to move objects at the current belief b_0. A belief tree has a maximum height H (planning horizon), with each node being a belief, and each edge being an action-observation pair. A new b' is obtained by performing Bayes' update on the current b with an edge (a,z). At each leaf node, a default policy is simulated to obtain a lower bound on its value. Then Bellman's principle of optimality (4) is applied to recursively compute the maximum value of action branches and the average value of observation branches. This results in an approximately optimal policy for the current belief b_0. The complete belief tree grows on the order of $\mathscr{O}(|A|^H|Z|^H)$. In practice, it is not affordable to search the full tree when the action space or the observation space is large. This is indeed the case in this work. To overcome this challenge, we use DESPOT to handle large observation space, and we exploit spatial constraints to reduce number of feasible actions.

$$V(b) = \max_{a \in A}\{\sum_{s \in S} b(s)R(s,a) + \gamma \sum_{z \in Z} p(z|b,a)V(b')\} \quad (4)$$

1) DESPOT: The key idea of DEterminized Sparse Partially Observable Tree (DESPOT) is to search a belief tree under K sampled scenarios only. Not all observation branches need to be examined to identify an approximately optimal policy, because the value of all observation branches is ultimately averaged in the Bellman's equation (4). A sampled subset of observations branches may be sufficient to estimate this average. Under each scenario, a policy traces out a particular sequence of action-observation pairs. The size of the DESPOT tree is reduced to $\mathscr{O}(|A|^H K)$, leading to dramatic improvement in computational efficiency for moderate K values.

DESPOT builds its tree incrementally by performing heuristic search guided by a lower bound and an upper bound on the value of each tree node. The lower bound at a leaf node b_l is the empirical value of a default policy under sampled scenarios. In this work, the default policy for the lower bound is to choose an action that maximizes the weighted total reduction in occlusion ratio and the movement cost of that action, which is of the same form as (5) but is calculated based on a state instance in the belief. The value of this policy under a set of sampled scenario can be easily calculated by simulation. The upper bound is an optimistic estimate of the value at a node. We employ the same strategy to choose actions, but we use underestimated cost of movement to obtain the upper bound value at b_l. Specifically, we assign a small constant cost to all valid MOVE_INSIDE actions. For the internal nodes of the DESPOT tree, we evaluate their lower and upper bounds recursively by (4).

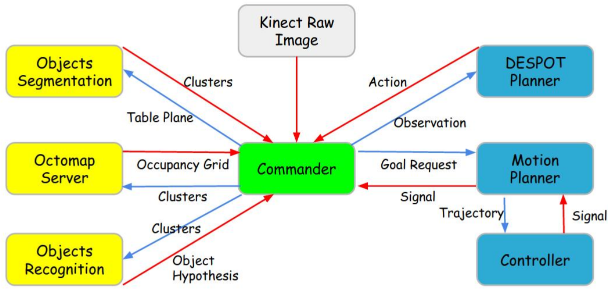


Fig. 3: System architecture.

2) *Enhancement:* DESPOT expands its tree by trying all possible actions. When the action space is large, the search is prohibitive. In our POMDP formulation, the number of possible actions is over 1000. Fortunately, exploiting spatial constraints helps reduce the number of feasible actions tremendously. The insight is that many of the actions are invalid. Specifically, before expanding the tree we apply an action filter to exclude invalid actions based on the current observation z. Invalid actions can be categorized into: 1) an object is not movable due to occlusion; 2) an action results in collision between objects; 3) an object is moved to a location that results in itself being occluded; 4) an action keeps an object where it is; 5) an action only moves an object in x direction. The result is that over 95% of the actions are filtered out. An interesting discovery is that as the number of objects inside the shelf increases, the number of feasible actions decreases. This is understandable that the remaining free space shrinks as more objects are present in a confined space. However, there are still around 80 actions to expand the tree. Another insight that further reduces the number of actions is that not all objects need to be moved in order to get the target object. This drives us to design a heuristic that unveils the utility of an action. Formally, the utility of an action a given the current observation z is defined as,

$$\begin{aligned} U(a,z) =& w_1 * ORT(a,z) + w_2 * ORO(a,z) \\ &+ w_3 * OB(a,z) + w_4 * MC(a) \end{aligned} \tag{5}$$

where w_1,w_2,w_3 and w_4 are constant weights, and

- $ORT(a,z)$: total reduction in occlusion ratio on the target object by taking a given z.
- $ORO(a,z)$: total reduction in occlusion ratio on objects other than the target.
- $OB(a,z)$: a bonus is assigned to a if it tries to move away an object that occludes the target.
- $MC(a)$: move cost of taking a.

The remaining actions after filtering are sorted by their utility in descending order. The top 10 actions are then used to expand the DESPOT tree. This enhancement enables the DESPOT to search for near optimal plan efficiently.

IV. Experiments and Results

A. Setup

We designed eight non-trivial test cases in V-REP[1] simulation environment as shown in Fig. 4. Across all test cases, the goal is to find out a pink-blue object and put it on a table besides the shelf. For each figure in Fig. 4, the top right and the bottom right windows are the RGB and depth images seen by the robot respectively. The left part of the figure is used to show the object configuration only. All test cases involve occlusion of varying degree. In test case (1), both the pink-blue and the yellow-blue objects are occluded, and the robot only sees the blue portion of both objects, which confuses the robot on where the target object is. Test cases (2), (3) and (4) are to test if our planner can find the optimal solution. Test case (4) is a blocking chain problem, in which two objects need to be moved in sequence in order to reach the target. Test case (5) is to verify whether our planner can tell the existence of the target object. In test case (6), the orange object has to be moved outside due to limited space. Test cases (7) and (8) are designed to show clear-cut advantage of POMDP planning over baseline approaches.

The system architecture is illustrated in Fig. 3. Our system is based on the Robot Operating System (ROS) Indigo in Ubuntu 14.04. The Kinect sensor streams RGBD images of the scene to the Commander, which relays the information to various modules for preprocessing. The Objects Segmentation module is adapted from *tabletop_segmentation* package[2]. Given a table plane, it segments out point cloud clusters on the table. The Octomap Server converts each cluster to its occupancy grid representation, which are the input features to our position estimator. It is adapted from *octomap_mapping* package[3]. The Objects Recognition module uses our color-histogram classifier to produce object hypothesis for each cluster. Commander fuses all processed information to form an observation, which is passed to the DESPOT planner. The planner returns an action to Commander, which requests the robot to execute that action. This process repeats until the program terminates.

We compared our POMDP planning with two other reasonable baselines. One employs greedy search strategy, which is the same as our lower bound. The other is an "Out-only" approach, which we humans often adopt to search for objects in clutter. At each step, the agent simply moves outside one object, which potentially occludes the target object most.

B. Results and Discussion

We performed 100 runs for each test case for all three approaches. We compared them based on two criteria: the average discounted reward with its standard deviation and the success rate in 100 runs. The result is shown in Table I. With limited space, we present the sequence of movement for some test cases for illustration as shown in Fig. 5. For more details, see the video online at `https://goo.gl/8MX55h`.

The results for all test cases demonstrate the capability of our approach to handle perception uncertainty under severe occlusion. In terms of the average discounted reward, our approach outperforms the greedy approach for 4

[1]http://www.coppeliarobotics.com/

[2]http://wiki.ros.org/tabletop_object_detector

[3]http://wiki.ros.org/octomap_mapping

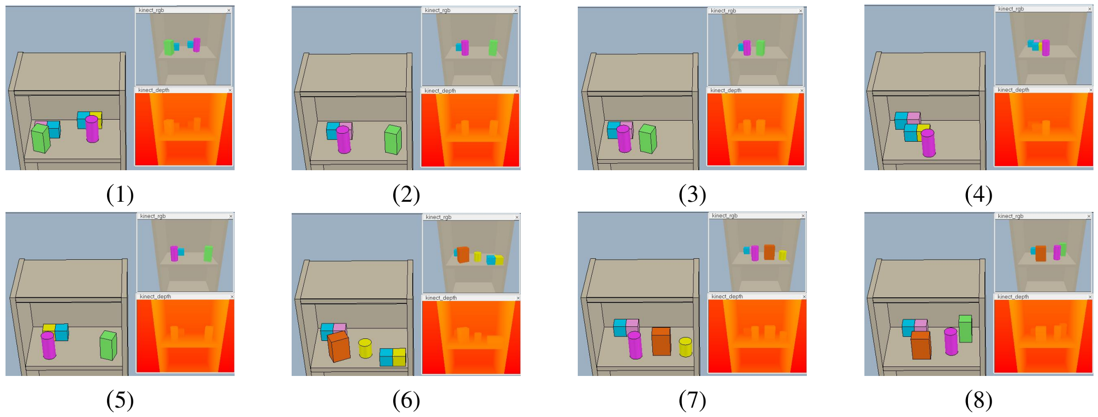


Fig. 4: Initial configurations of eight test cases. The objective is to find out and retrieve the pink-blue object.

TABLE I: Comparison of POMDP planning and baseline algorithms.

		(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Average Reward	DESPOT	**-11.87 (4.41)**	0.50 (0.00)	**-4.50 (0.00)**	**-16.25 (0.56)**	-5.52 (0.00)	-190.50 (0.00)	-190.50(0.00)	**-10.50 (0.15)**
	Greedy	-12.43 (0.00)	0.50 (0.00)	-8.58 (0.00)	-19.66 (0.0)	-5.52 (0.00)	-190.50 (0.00)	-	-
	"Out-only"	-354.31 (66.09)	-190.50 (0.00)	-190.50 (0.00)	-380.98 (0.00)	-190.50 (0.00)	-190.50 (0.00)	-190.50 (0.00)	-190.50 (0.00)
Success Rate	DESPOT	0.92	0.97	0.98	0.84	1.00	1.00	1.00	0.83
	Greedy	1.00	1.00	1.00	1.00	1.00	1.00	-	-
	"Out-only"	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

((1),(3),(4),(8)) out of the 8 cases and matches the performance in the remaining 4 cases. The "Out-only" strategy incurs much larger cost by only considering moving objects out. We take a closer look at a few test cases, where our approach can find near-optimal solution while the greedy approach cannot. As illustrated in Fig. 5, in test case (4), the greedy approach first moves the pink cylinder slightly to the right. Then it realizes there is no space to put away the yellow-blue object, so it moves the pink cylinder to the right again, which creates enough space to move the yellow-blue to the right. Finally, the target object can be retrieved. In contrast, our approach creates enough space in the first step by moving the pink cylinder further to the right. The reason is that the greedy baseline chooses an action that does not take into account the long-term rewards. This results in a suboptimal solution. Our approach looks ahead several steps to choose an optimal action given the horizon. The greedy approach may even be stuck forever for certain configurations ((7),(8)). In test case (8), the greedy approach falls into an infinite loop by moving the orange block from the left to the right and then from the right to the left. This is because the heuristic value of moving the orange block is higher than that of moving the pink cylinder. On the contrary, POMDP planning employs lookahead search and figures out that the optimal strategy is to move the pink cylinder to the right first, which creates space for the large orange block. Despite the fact that the immediate reward of moving the pink cylinder is less than that of moving the orange block, its long-term reward is much higher. Another observation is that our approach usually takes fewer steps to retrieve the target than the greedy approach. This serves as positve evidence that our approach will be more efficient on real robotic systems, because fewer steps imply less execution time in general.

With our current implementation, the POMDP planning does not achieve 100% success rate. Sometimes our planner moves objects too close to each other, which results in segmentation failure, i.e., two clusters merge into one. This totally messes up the observation in the current system. Hence, we treat such cases as failures. This is a normal behavior of our planner because of its aggressiveness in maximizing the long-term reward. For example, in test case (3), moving the pink cylinder anywhere to the right of the green block can reveal the target. Therefore, moving the pink cylinder closer to the green block is preferred because it minimizes the moving cost. Even though we enforce a safety margin to prevent objects from moving too close to each other, it still happens due to inaccurate position estimates. Nonetheless, one workaround is to track each object [14] in real time so that the system will not be confused when objects get too close to each other.

V. Conclusion

Motivated by the idea of *Act to See and See to Act*, this paper presents a POMDP framework for objects search in clutter with large state space and action space. Exploiting spatial constraints makes it possible for our planner to efficiently search for near-optimal plans under perception uncertainty due to partial occlusion. Experiments show clear-cut advantages of our planner over greedy heuristic search and a natural baseline strategy. One lesson we have learned is that when a scene has complex occlusion relationships, multi-step lookahead POMDP planner is preferred since it takes the long-term effects into account. In the future, we plan to connect the current system with our real robotic manipulator to complete the story. Future experiments will contain randomly generated scenes, which may help reveal more about the problem structure of object search in confined

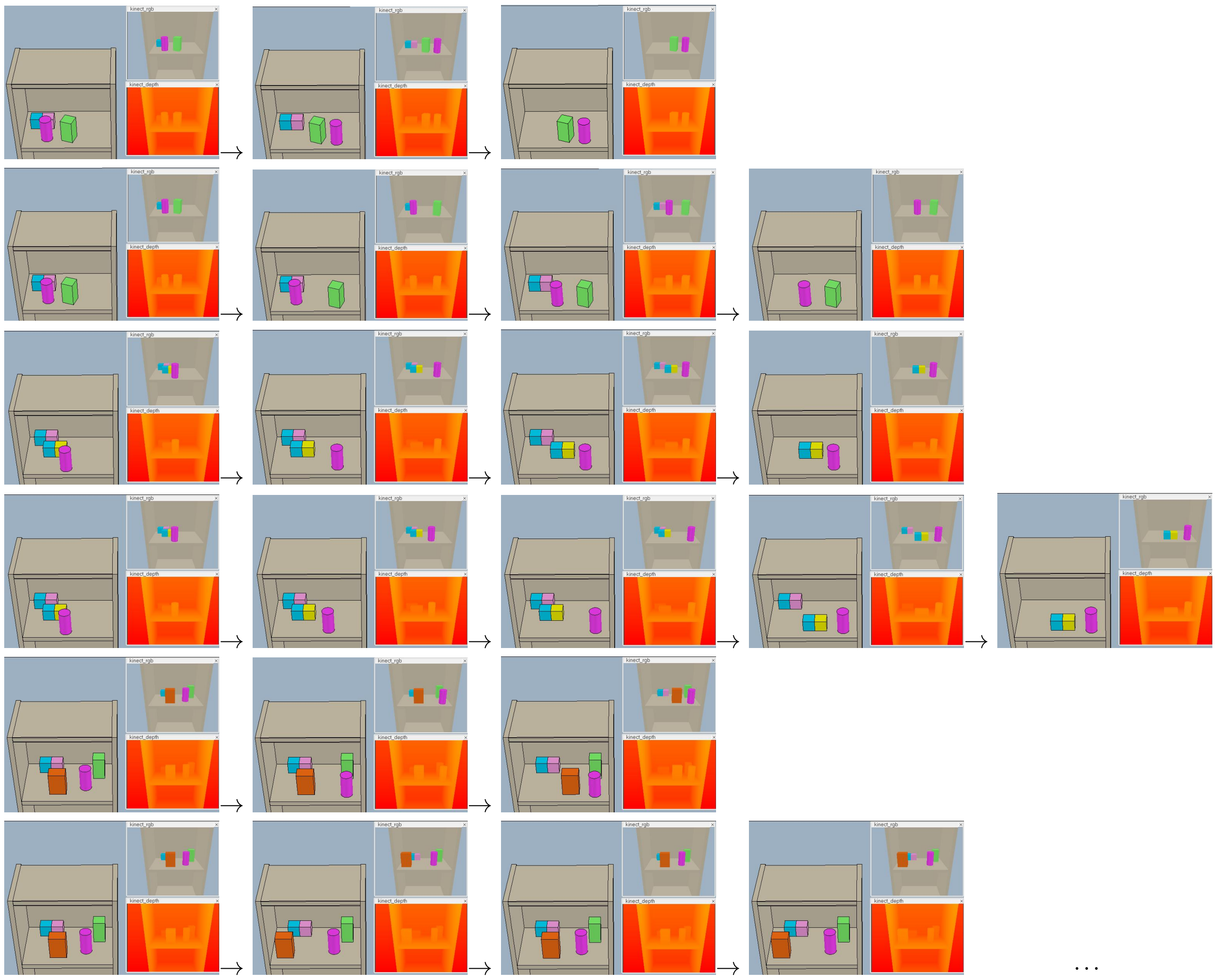


Fig. 5: Sequence of moves (from top to bottom: case (3) with our approach; case (3) with greedy approach; case (4) with our approach; case (4) with greedy approach; case (8) with our approach; case (8) with greed approach).

spaces. These experiments will be conducted on a set of real household objects from the YCB Object and Model Set[4].

ACKNOWLEDGMENT

This research was supported in part by A*STAR Industrial Robotics Program grant (R-252-506-001-305).

REFERENCES

[1] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "DESPOT: Online POMDP planning with regularization," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 1772–1780, 2013.

[2] J. Pajarinen and V. Kyrki, "Robotic manipulation of multiple objects as a POMDP," *Artificial Intelligence*, 2015.

[3] D. Nieuwenhuisen, A. F. van der Stappen, and M. H. Overmars, "An effective framework for path planning amidst movable obstacles," in *Algorithmic Foundation of Robotics VII*, pp. 87–102, Springer, 2008.

[4] J. Van Den Berg, M. Stilman, J. Kuffner, M. Lin, and D. Manocha, "Path planning among movable obstacles: a probabilistically complete approach," in *Algorithmic Foundation of Robotics VIII*, pp. 599–614, Springer, 2009.

[5] M. Stilman, J.-U. Schamburek, J. Kuffner, and T. Asfour, "Manipulation planning among movable obstacles," in *Proc. IEEE Int. Conf. on Robotics & Automation*, pp. 3327–3332, 2007.

[6] L. L. Wong, L. P. Kaelbling, and T. Lozano-Pérez, "Manipulation-based active search for occluded objects," in *Proc. IEEE Int. Conf. on Robotics & Automation*, pp. 2814–2819, 2013.

[7] M. R. Dogar, M. C. Koval, A. Tallavajhula, and S. S. Srinivasa, "Object search by manipulation," *Autonomous Robots*, vol. 36, no. 1-2, pp. 153–167, 2014.

[8] M. Gupta, T. Ruhr, M. Beetz, and G. Sukhatme, "Interactive environment exploration in clutter," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pp. 5265–5272, 2013.

[9] M. J. Schuster, J. Okerman, H. Nguyen, J. M. Rehg, and C. C. Kemp, "Perceiving clutter and surfaces for object placement in indoor environments," in *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pp. 152–159, IEEE, 2010.

[10] Y. Jiang, M. Lim, C. Zheng, and A. Saxena, "Learning to place new objects in a scene," *Int. J. Robotics Research*, 2012.

[11] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 2164–2172, 2010.

[12] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online POMDP planning for autonomous driving in a crowd," in *Proc. IEEE Int. Conf. on Robotics & Automation*, pp. 454–460, 2015.

[13] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[14] A. Teichman, J. T. Lussier, and S. Thrun, "Learning to segment and track in RGBD," *Automation Science and Engineering, IEEE Transactions on*, vol. 10, no. 4, pp. 841–852, 2013.

[4]http://rll.eecs.berkeley.edu/ycb/