Planning for Grasp Selection of Partially Occluded Objects

Sung-Kyun Kim, Maxim Likhachev
The Robotics Institute
Carnegie Mellon University
{kimsk,maxim}@cs.cmu.edu

***Abstract*— In a cluttered scene, an object is often occluded by other objects, and a robot cannot figure out what the object is and perceive its pose exactly. We assume that the robot is equipped with a depth sensor and given a database of 3D object models and their grasping poses, but yet there is uncertainty about object's class and pose. In this paper, we study the problem of how to predict the class and pose of an occluded object by carefully taking a sequence of observations. To find the best sequence of viewpoints by the robot, we construct hypotheses of the states of the target and occluding objects, and update our belief state as new observations come in. Every time selecting the next robot pose, we greedily choose the one that is expected to reduce the uncertainty the most. Based on the theoretical analysis of adaptive submodular maximization problems, this process is guaranteed to find a near-optimal sequence of robot poses in terms of observation and traverse costs. To validate the proposed method, we present simulation and robot experiments using a PR2.**

I. Introduction

It happens frequently that an object is occluded by another when there are multiple objects on a table as shown in Fig. 1. It is not easy for a robot to make a decision how to grasp the occluded object even in the case that we know that the object is one of objects in our database with predefined desirable gripper poses to grasp. There can be several objects in the database in several different poses that roughly match the partially observed point cloud. So we need to consider them as hypotheses that can possibly be the true state (class and pose) of the target object. The problem is that there would be many hypotheses.

If a single observation of a cluttered scene contains insufficient information to infer the state of the target object, we need to take another observation to gather more information. From a sequence of observations, we can update the probability of each hypothesis by comparing it with the new point cloud and declare a hypothesis as the most likely state when we think we have found sufficient evidence.

There are two questions that need to be answered in this approach. One is how to extract information from point cloud observation and update our belief states, i.e., the probability of hypotheses. The other is how to determine a next action that would allow us to get the most informative observation.

For the belief state update, we basically utilize Bayesian recursive estimation. In our specific problem of cluttered environment, it is important to build good observation model

*This research was sponsored by ARL, under the Robotics CTA program grant W911NF-10-2-0016.

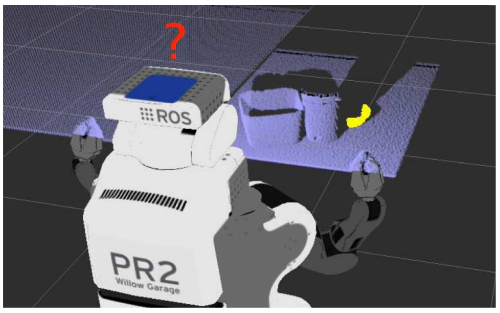


Fig. 1. A scene of objects in clutter. The cluster of yellow points is the target object assigned by the user.

that can represent relationship of the target object and the other objects (called *obstacles* hereafter) sufficiently and efficiently. We abstracted the geometric information of obstacles as simple 3D shape primitives, such as a sphere, a cylinder or a box. Furthermore, there is key prior knowledge that is useful in belief update; a target object is in a stable pose on a table, does not collide with (overlap) obstacles, and occludes the table surface where it exists. More details are presented in section V.

In action selection, we applied adaptive submodularity to our problem. By representing the amount of uncertainty of the true state of the target object as an adaptive submodular and monotonous utility function, a simple greedy algorithm that evaluates just one step ahead and takes the action with the maximum expected margin of the utility function can lead us a theoretically provable near-optimal solution of action sequence. Section IV provides detailed formulation and pseudo-code to understand how belief update and action selection are executed.

To validate our approach, we conducted experiments in simulation and in real world with a robot. In section VI, four test cases are presented and analyzed, and the attached video shows the overall process of active object recognition in a cluttered scene.

II. Previous Works

Active recognition of an occluded object can be formulated as a partially observable Markov decision process (POMDP) because the state of the object is unknown and the sequence of actions should be selected carefully for efficient recognition [1], [2], [3], [4]. Hsiao *et al.* applied POMDP to active object localization problem in [5], which is in a

similar form to active recognition. However, in general, the time complexity to optimally solve a POMDP problem is exponential in the search depth, which makes it intractable. As a workaround, it is approximated by limited horizon or assumed to have a small branching factor.

Recently, adaptive submodularity strategy was proposed and applied to several robotics problems [6], [7], [8]. Submodular function maximization can be regarded as a discrete version of concave maximization, and a simple greedy algorithm can give a near-optimal solution for monotonous submodular problems [9], [10]. This paper adopts this framework and theoretical analysis to the active object recognition problem.

Chen *et al.* proposed an active object detection method based on adaptive submodularity, which determines a sequence of queries (actions) for user interventions (observations) to improve the detection performance in the runtime [6]. This is a different framework from ours in the notions of "state" and "action". In our approach, the robot only considers actions for autonomous observations such as repositioning and re-scanning the object.

Rather, a tactile localization problem that Javdani *et al.* addressed as adaptive submodular maximization in [7] is closer to our problem. Because possible states of poses of a (known) object are represented as particles, touch-based observation for each state is straight-forward in their formulation, while it is more complex to consider 3D point observations in the presence of obstacles as in our problem.

Another related work is an underwater vehicle inspection problem presented by Hollinger *et al.* [8]. The goal is to find an informative path to reconstruct 3D meshes of a vehicle, and the metric for uncertainty is defined by the mean and variance of observed meshes nearby. Our paper deals with uncertainty of object identification and its 3D pose, mostly coming from occlusion by obstacles, which can be more unstructured than continuous surface regression.

III. Problem Description

A. Partially Observable Markov Decision Process Model

As mentioned in section I, we can formulate the active object recognition problem in a cluttered scene as a partially observable Markov decision process (POMDP) problem. First of all, we define two goals in the problem. One is to estimate the true state (class and pose) of the target object given a series of observations,

$$p(\phi|o_a^{0:t},\, a^{0:t}), \tag{1}$$

where $\phi \in \Phi$ is a random variable that represents a state, and $o_a^{0:t}$ is a sequence of observations after individual actions, a, from step 0 to t. This probability distribution can be updated by recursive Bayesian estimation (Bayes filter).

The other is to find a policy to select a next action, a^{t+1}, based on observations and executed actions so far,

$$a^{t+1} = \pi(o_a^{0:t},\, a^{0:t}), \tag{2}$$

such that it can minimize the cost until we gain sufficient confidence in the state estimation, which can be formulated as the following.

$$\begin{aligned} &\text{minimize}_\pi\ \mathbb{E}_\Phi[c(A(\pi,\Phi))] \\ &\text{subject to } f(A(\pi,\phi),\phi) > Q, \end{aligned} \tag{3}$$

where π stands for a policy that depends on $o_a^{0:t}$ and a^{t+1}. $A(\pi,\phi)$ is the sequence of actions selected by π when the true state is ϕ, and $c(\cdot)$ is a cost function of $A(\pi,\phi)$. $f(\cdot)$ is a utility function which is defined as 1 minus a sum of probability over hypotheses (see section IV for more details) so that it can encode confidence in the state estimation. Q is a threshold for the utility function.

This problem can be formalized and specified as a POMDP model as follows.

- *State*: Target object class and its pose, and obstacles and their poses. We assume that the target object is one of the objects in the database while we do not limit classes of the obstacles.
- *Observation*: 3D point cloud measurement from the robot sensor that contains a part of a target object that is not occluded by obstacles and visible parts of obstacles as well.
- *Action*: One dimensional motion of the robot base moving around the table that the objects are placed on, and one dimensional vertical motion of the robot torso. We set the angle of rotation around the table and the change in torso height as discretized.
- *Transition model*: Assumed to be deterministic without any variance, that is, there is no probability for the robot to transit to any poses other than the controller assigned pose. See Discussion 1 for more detail.
- *Observation model*: Probability of an observation given a state. In our problem, it is associated with an error metric how a partial observation matches to given object class and pose. We assume the sensor measurement of point cloud pose and its color map is deterministic. See Discussion 2 for more detail.
- *Utility function*: Adaptive monotonous submodular function that encodes the amount of confidence (uncertainty reduction) in the state estimation. A lower value represents higher uncertainty.
- *Reward function*: Expected marginal utility per cost for an action at a state, that is, expected amount of uncertainty removal by an observation obtained from an action over the cost to execute that action.

Discussion 1: Deterministic transition model

This issue mainly originated from the dimensionality of state. Since our application deals with 3D point cloud, the observation model itself is computationally expensive. If we have high dimensional state, one step look-ahead expansion for next action selection requires a lot of computation. In order to reduce the dimension of state, we are using quantized action space which makes the robot to consider only discrete poses. If we want to consider probabilistic transition model, possible poses of a robot should be continuous to make sense, but the problem would be intractable in our framework. We still can use discrete robot poses with non-zero variance, but

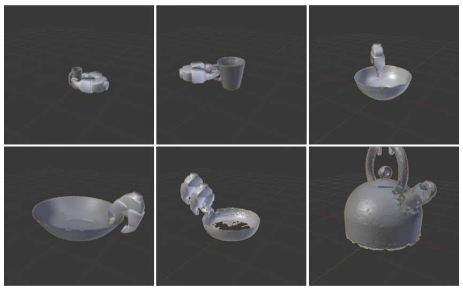


Fig. 2. Object database of 3D geometric model and predefined desired gripper poses for grasping.

in this work, we just assume the transition variance to be zero for simplicity.

Discussion 2: Deterministic sensor measurement

This assumption means each sensor measurement from any (discrete) possible robot pose gives the exact pose (but not necessarily the point cloud of partially observable object geometry) of the true object in robot frame. This allows us to consider the hypothetic object pose state as a relative pose with respect to the true sensor measurement and to re-align it according to the latest sensor measurement at each step. As a result, we do not care about the object pose uncertainty that came from robot motion and sensor measurement error in our action selection framework. Then, the uncertainty we consider is only from "partial" observation of target object and obstacle shapes, which lets us focus on the core issue of occlusion.

B. Object Database

In this work, it is assumed that we have a database with full information about object geometry, symmetry, and desired grasp poses as shown in Fig 2. Object geometry is described by a 3D point cloud, and its symmetric property, such as rotational symmetry of a bowl about a vertical axis, is represented by a symmetry label which is assigned as an additional field value for each point in the point cloud. Provided that the robot has a simple one degree-of-freedom gripper, a grasp pose is described as a transformation matrix of 3D position and orientation of the gripper.

IV. Formulation of Belief State Update and Action Selection

A. Properties of Adaptive Monotone and Submodularity

Before developing our problem as an adaptive submodularity maximization problem, we introduce the properties of adaptive monotone and submodularity first. If a utility function, f, satisfies the following properties for $A \subseteq B \subseteq \mathbb{A}$ and $a \in \mathbb{A} \backslash B$, it is called monotonous and submodular, respectively.

- *Monotonicity*:

$$f(A \cup \{a\}) - f(A) \geq 0 \tag{4}$$

- *Submodularity*:

$$f(A \cup a) - f(A) \geq f(B \cup a) - f(B) \tag{5}$$

In adaptive setting, they are modified as follows.

- *Strong Adaptive Monotonicity*:

$$\mathbb{E}[f(A \cup \{a\}, \Phi)|\psi, o] - \mathbb{E}[f(A, \Phi)|\psi] \geq 0 \tag{6}$$

- *Adaptive Submodularity*:

$$\begin{aligned} &\mathbb{E}[f(A \cup \{a\}, \Phi) - f(A, \Phi)|\psi_A] = \Delta_f(a|\psi_A) \\ \geq &\mathbb{E}[f(B \cup \{a\}, \Phi) - f(B, \Phi)|\psi_B] = \Delta_f(a|\psi_B) \end{aligned} \tag{7}$$

where ψ_A and ψ_B are sequences of observations according to sequences of actions, A and B, respectively.

B. Formulation as Adaptive Submodular Maximization

Now we develop our problem as an adaptive submodular problem. First, more concrete problem description and definitions are provided in the following.

- *State realization*: We denote a state as a tuple of object class, object pose, and obstacle point cloud in global frame.

$$\phi = (ID_{obj}, T_{obj}, \overline{X}_{obs}) \tag{8}$$

- *Realization (hypothesis) probability*: The probability of $\phi \in \Phi$ is

$$p(\phi) = \mathbb{P}[\Phi = \phi] \tag{9}$$

where Φ is a random variable over all realization. Every $\phi \in \Phi$ is regarded as a *hypothesis*, and our goal is to find a hypothesis that is most likely to be the true state.
- *Observation*: $o \in O$ is a possible observation which depends on a state of relative poses between a object, obstacles, and a robot.
- *Action*: $a \in \mathbb{A}$ encodes a discretized rotating motion of the robot base about the table and a discretized vertical displacement of the torso. $A \in \mathbb{A}$ stands for a sequence of actions selected so far, and it is assumed that $\mathbb{A}$ is a sufficient set of actions to find a solution.
- *Cost function*:

$$c(a^t) = c_\theta |\Delta\theta^t| + c_h |\Delta h^t| + c_0 \tag{10}$$

$$c(A) = \sum_{a \in A} c(a) = \sum_{t=0}^{|A|-1} c(a^t) \tag{11}$$

where $\Delta\theta^t$ is a base rotation angle about z-axis centered at the table, and Δh^t is a change in torso height at t-th step, respectively. c_θ and c_h are positive weights for base rotation angle and torso height change, respectively, and c_0 is a positive constant to penalize the number of actions which is related to the time to obtain and process an observation. $|A|$ indicates the cardinality of A, i.e., the number of actions that have been taken.

Conceptually, the overall algorithm is as follows: 1) run 3D point cloud processes for a new observation, 2) update the belief state according to fitness to each hypothesis, 3) compute expected marginal utility for each action, 4) greedily select and take the action with maximum expected marginal utility per cost, and 5) iterate above processes until sufficient

information is gathered. In Appendix, the pseudo-code is presented to show how belief state update and action selection are conducted iteratively.

Then we formally introduce, with the notion of belief state, a utility function that is adaptive, monotonous and submodular, and a greedy strategy for action selection.

- *Belief state*: Let us denote a partial realization, by ψ which is a sequence of observations corresponding to a sequence of actions, A. The probability of a hypothesis for a partial realization, ψ, is as follows.

$$p_\psi(\phi) = p(\phi, \psi) \tag{12}$$

  Note that this is an unnormalized probability distribution over Φ such that $\sum_{\phi\in\Phi} p_\psi(\phi) \leq 1$ because we curtail probability mass of $p_\psi(\phi)$ individually without normalization to encode the uncertainty reduction.
- *Weight function for hypothesis pruning*: A down-weight determined by an action, a, and a corresponding partial observation, o_a, is defined as

$$w_{o_a}(\phi) = w(o_a, o_\phi) = 1 - \min\left(\frac{e(o_a, o_\phi)}{e_T}, 1\right) \tag{13}$$

  where o_ϕ is an ideal, full observation when the true state is ϕ. $e(o_a, o_\phi)$ is a fitness error function that returns the mean of the Euclidean distances between corresponding points of o and o_ϕ, and e_T is a threshold for selecting candidate object classes. So, the weight will be 1 if o_a matches to o_ϕ exactly and 0 if the fitness error is greater than a threshold.
- *Prior violation penalty on weights*: As mentioned in section I, prior knowledge can provide strong reasoning to prune unlikely hypotheses. If a hypothesis collides with observed obstacles or does not occlude the table surface that should have been occluded, the weight gets a harsh penalty.

$$w_{o_a}(\phi) = \epsilon_p w_{o_a}(\phi) \tag{14}$$

  where ϵ_p is a prior violation penalty constant which is positive but much less than 1.
- *Belief state update*: Belief state is updated by multiplying the weight for o_a to the previous belief state.

$$p_\psi(\phi) = \left(\prod_{\{o_a\}\in\psi} w_{o_a}(\phi)\right) p(\phi) \tag{15}$$

- *Probability mass*: As a quantity for uncertainty in state estimation, we define probability mass over Φ for given partial realization, ψ, by

$$M_\psi = \sum_{\phi'\in\Phi} p_\psi(\phi') \tag{16}$$

  Furthermore, updated probability mass after action a with observation o_a is denoted as

$$M_{\psi,o_a} = \sum_{\phi'\in\Phi} w_{o_a}(\phi') p_\psi(\phi') \tag{17}$$

- *Utility function*: We define a utility function as

$$f(A, \phi) = 1 - M_\psi \tag{18}$$

  Since the weight function can only remove the probability mass and never add to it, this is strong adaptive monotonous and adaptive submodular, which follows directly from Theorem 1 in [7] and Theorems 5.8 and 5.9 in [10].
- *Marginal utility for action a and observation o*:

$$f_{\psi,o_a} = f(A\cup\{a\}, \phi) - f(A, \phi) \tag{19}$$
$$= M_\psi - M_{\psi,o_a} \tag{20}$$

- *Probability of observation o_a from action a over Φ*:

$$\mathbb{P}(o_a = o_\Phi|\psi) = \sum_{\phi\in\Phi} \mathbb{P}(o_a = o_\Phi|\phi, \psi)\mathbb{P}(\phi|\psi) \tag{21}$$

  where $\mathbb{P}(o_a = o_\Phi|\phi, \psi) = \mathbb{P}(o_a = o_\phi|\psi) = w_{o_a}(\phi)$ can be regarded as the weight function, and $\mathbb{P}(\phi|\psi) \propto p(\phi|\psi) = p_\psi(\phi)$ is a normalized version of belief state.
- *Expected marginal utility for action a*:

$$\Delta_f(a|\psi) = \mathbb{E}[f_{\psi,o_a}] \tag{22}$$
$$= \sum_{o_a\in O} \mathbb{P}(o_a = o_\Phi|\psi) f_{\psi,o_a} \tag{23}$$

- *Simple greedy algorithm*: Finally, a simple greedy algorithm selects the action with the maximum expected marginal utility per cost, $\Delta_f(a|\psi)/c(a)$, by inspecting just one step ahead. Again, the adaptive monotonous submodular property guarantees that a simple greedy algorithm with horizon 1 gives a near-optimal solution.

C. Theoretical Performance Guarantee

In this section, we briefly summarize the theoretical performance guarantee of the proposed method. The utility function is adaptive submodular and strongly adaptive monotone due to the property of our down-weight function [7]. As cost of an action includes traverse cost that depends on the current pose of the robot, this problem is a special case of adaptive submodular problem with arbitrary costs. From the results of [10] and [11], we can prove that it has $\left(\ln\left(\frac{Q}{\min_\phi p(\phi)}\frac{c_{max}(a)}{c_{min}(a)}\right)+1\right)$-approximation for optimal worst case policy.

Note that, in our problem, observation cost dominates traverse cost when cost is simply taken as duration of time to perform an action, and thus, $\frac{c_{max}(a)}{c_{min}(a)}$ can be close to 1. As a cost minimization problem, this problem has large adaptivity gap [8], so the ratio of observation/traverse costs can effectively play a role in trading off *exploration* of the scene to get more information and *exploitation* of the current belief to reduce the total cost [12], [13].

V. 3D Point Cloud Process

In this section, the overall process of 3D point cloud as shown in Fig. 3 is described. There are five main subprocesses that arise while applying adaptive submodularity scheme to 3D point cloud data.

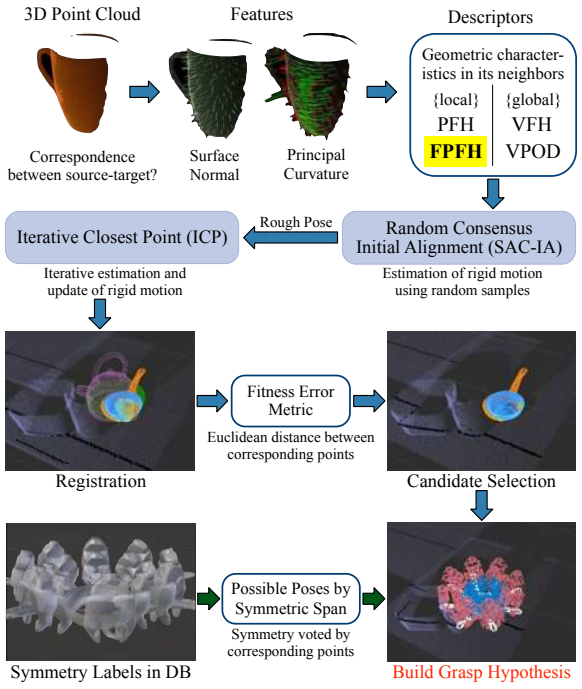


Fig. 3. Flowchart of 3D point cloud processing.

A. Object Template Registration

In order to evaluate each hypothesis with observation, we need to register the object templates in the database to the observed point cloud first.

3D point cloud contains purely geometric information, such as surface normal and principal curvature. There are several feature descriptors for 3D point cloud, and we used Fast Point Feature Histogram (FPFH) which is a histogram descriptor for variation of surface normal around the query point [14].

Iterative Closest Point (ICP) algorithm is widely used for finding rigid-body transformation between two point clouds but can be easily stuck to local minima [15]. Since Random Consensus Initial Alignment (SAC-IA) algorithm can find a globally approximate but coarse solution [14], ICP is used to refine the registration after SAC-IA is applied.

Based on the prior knowledge that any objects lie on the table in stable poses, we can reduce the dimension of search space for finding a pose of a rigid body from 6 to 3, which can be applied to both of SAC-IA and ICP.

B. Weight Computation

For registered object templates by the previous process, we need to prune away implausible hypotheses with high fitness errors. Fitness error metric is defined as the mean of Euclidean distances between corresponding points in source and target point clouds, and the object classes with high errors are removed from the candidate list as described in (13) and (15).

Note that as discussed in (14), violation of prior knowledge imposes high penalty on weights. For collision detection with

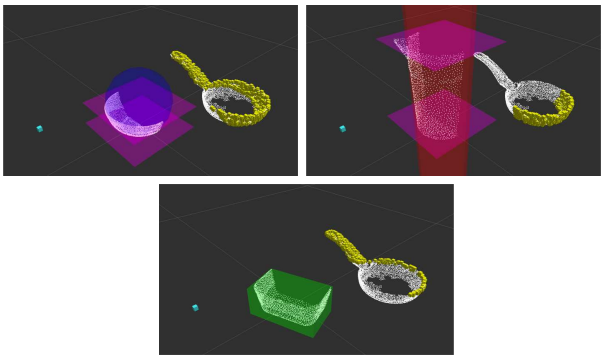


Fig. 4. Examples of obstacle primitive estimation and corresponding expected observation of the target object. For sphere and cylinder cases, the obstacle is regarded to occupy the volume which is inside the primitive and between the two parallel planes at the same time. A cyan square on the left represents the sensor position, and yellow squares are the visible points of the target object that are not occluded by the obstacle primitives.

obstacles, we check for collision between a hypothesis and actually observed obstacle point clouds. Since it needs to be conservative in checking for violation, obstacle primitives, which are approximate models, are not used. For visibility violation, we first sample points from a hypothesis that do not have any corresponding points in the observed target cloud, and check if those points are also occluded by obstacle primitives or not. If not for many sampled points, this hypothesis is considered to violate the visibility prior.

C. Grasp Hypothesis Construction

State hypotheses are constructed from the remaining candidate object classes and their possible poses. If all the template points with correspondence to the observation have the same symmetry label, it implies that the observation has no evidence to determine the pose of the template about that symmetry. For example, if a frying pan template has correspondence with an observation only in the circular part, it cannot be decided where the handle should be located.

Using the symmetry labels in the object database and the correspondence information, possible object poses are generated for each candidate object classes. As a result, the cardinality of the total grasp hypotheses becomes $|\Phi| = \sum_{i=1}^{n} j_i$, where n is the number of valid candidate objects, and j_i is the number of possible object poses by symmetry for i-th object class.

D. Obstacle Primitive Estimation

In order to compute expected marginal utilities as in section IV, we need to enumerate all possible actions and observations. In other words, we need to compute a fitness error between every possible $o_\phi \in O$ and $o_a \in O$. To generate all possible observations, O, over all possible object classes and poses, and robot poses, we need to build an obstacle map.

For computational efficiency, parametric shape primitives rather than a full obstacle occupancy grid are used in this work to represent the observed obstacles. As shown in Fig. 4, we use primitives of sphere, cylinder, and bounding box [16].

For better representation, two parallel planes offer additional boundaries for sphere and cylinder cases.

When an observed, probably partial, obstacle point cloud is given, RANSAC fitting for sphere and cylinder is tried first and selected if a large portion of points are inliers to the model. If both have bad fitnesses, bounding box, which is the most conservative estimation for the occluded geometry of the obstacle, is selected as a last resort.

E. Expected Marginal Utility Computation

After all obstacles are represented by shape primitives, then we can generate an expected observation for each hypothesis, ϕ, based on ray-tracing algorithm (Fig. 4). For given obstacle primitives, candidate object class and pose, and sensor pose, we first sample points from the candidate object point cloud, and check if the ray from the sample point to the sensor intersects with the obstacle primitives. With geometric/algebraic algorithms, we can efficiently check the intersection with these parametric primitives.

When we have all the expected observations, we can compute fitness scores, $e(o_a, o_\phi)$, accordingly, and finally get the expected marginal utility as in (20) to (23).

VI. Experimental Results

Experiments are conducted in simulation and with a real robot. For analysis purposes, we mostly present simulation results which allows us to explore different cluttered scenes precisely, while robot experiments are for validation in real world situations as presented in the attached video.

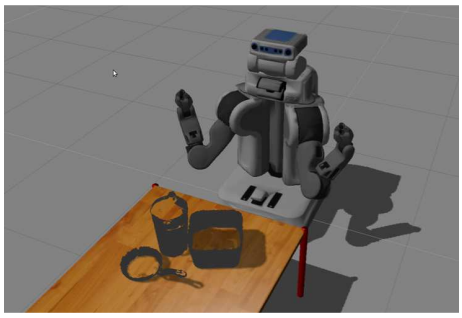
A. Simulation Environment Setup

- *System*: The simulation platform is Ubuntu 12.04 with ROS Groovy. Dynamic simulation of PR2, a dual-arm robot with a tilting laser scanner, is run by Gazebo 1.9.5 as shown in Fig. 5(a). Implementation of 3D point cloud processing utilized Point Cloud Library [17].
- *Database*: An object database that contains object geometry, symmetry labels, and grasp poses is prepared (Fig. 2). The mesh models of IKEA kitchen objects are obtained from [18], and their symmetry labels and grasp poses are generated from preprocessing and verified by a human.

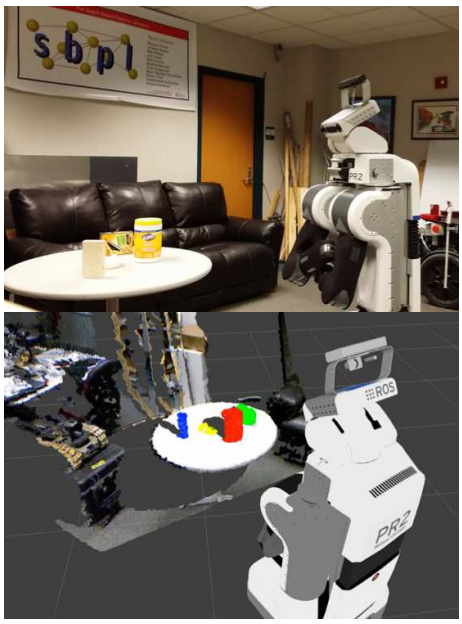
B. Simulation Results

Simulation results of the proposed method for four different cases are presented in Fig. 6, 7, and 8. To explain a case with scene 1, two object classes were selected as candidates from the initial observation based on the fitness error metric, and eight possible object poses were established as hypotheses for each object class. You can see that two hypotheses of object 1 have bad probability after the first action. This is because those hypotheses violated prior conditions.

By computing expected marginal utilities, an action of rotating 0.4 rad about the table was assigned to the robot. The utility function increased by 0.2 after the second observation. However, the evidence collected so far was not sufficient, and another action of rotation by 0.2 rad was executed.



(a) Simulation using Gazebo



(b) Robot experiment and its visualization

Fig. 5. Simulation and robot experiment environment in a cluttered scene.

Finally, small part of the handle appeared in the third observation, and thus, the bowl-shaped object was excluded from the candidates, and uncertainty was significantly removed so that a feasible grasp pose can be determined.

To demonstrate the effectiveness of the proposed method, breadth-first search (BFS) was selected as a baseline and compared with our method for the same task. BFS simply tries all the possible robot poses (nodes) by taking the minimum action of base rotation and torso height change until it finds a feasible grasp pose, and it was found to be clearly suboptimal compared to the proposed method.

VII. Conclusion

This paper investigated how to select a sequence of actions adaptively to estimate an occluded target object's class and its pose. The problem was formulated as adaptive submodular maximization problem so that a simple greedy algorithm can provide a provably near-optimal solution. Equipped with 3D point cloud processing algorithms for registration and fitness error metric, this framework was successfully applied to the active object recognition problem and showed its validity by simulation of several example tasks.

For the future work, there are remaining issues to improve this method to make it more general and efficient. Classifiers such as support vector machine can be applied to determine candidate objects rather than using a constant threshold of fitness error. For more general obstacle representation and reasonable generation of expected observations, 3D voxel occupancy grid can be used.

Additionally, we plan to incorporate multi-modal information of objects, such as color intensity or tactile property. Since multi-modal features would have multiple metrics to represent uncertainty, it would be important to study how to combine them into a single metric within the current framework.

REFERENCES

[1] D. Wilkes and J. K. Tsotsos, "Active object recognition," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1992, pp. 136–141.

[2] S. J. Dickinson, H. I. Christensen, J. K. Tsotsos, and G. Olofsson, "Active object recognition integrating attention and viewpoint control," *Computer vision and image understanding*, vol. 67, no. 3, pp. 239–260, 1997.

[3] B. Schiele and J. L. Crowley, "Transinformation for active object recognition," in *Proceedings of International Conference on Computer Vision (ICCV)*, 1998, pp. 249–254.

[4] H. Borotschnig, L. Paletta, M. Prantl, and A. Pinz, "Appearance-based active object recognition," *Image and Vision Computing*, vol. 18, no. 9, pp. 715–727, 2000.

[5] K. Hsiao, L. P. Kaelbling, and T. Lozano-Perez, "Grasping pomdps," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 4685–4692.

[6] Y. Chen, H. Shioi, C. F. Montesinos, L. P. Koh, S. Wich, and A. Krause, "Active detection via adaptive submodularity," in *Proceedings of International Conference on Machine Learning (ICML)*, 2014, pp. 55–63.

[7] S. Javdani, M. Klingensmith, J. A. Bagnell, N. S. Pollard, and S. S. Srinivasa, "Efficient touch based localization through submodularity," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 1828–1835.

[8] G. A. Hollinger, B. Englot, F. S. Hover, U. Mitra, and G. S. Sukhatme, "Active planning for underwater inspection and the benefit of adaptivity," *The International Journal of Robotics Research*, pp. 3–18, 2012.

[9] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *Journal of Artificial Intelligence Research*, vol. 34, no. 2, p. 707, 2009.

[10] D. Golovin and A. Krause, "Adaptive submodularity: Theory and applications in active learning and stochastic optimization," *Journal of Artificial Intelligence Research*, vol. 42, no. 1, pp. 427–486, 2011.

[11] A. Guillory and J. Bilmes, "Average-case active learning with costs," in *Algorithmic Learning Theory*. Springer, 2009, pp. 141–155.

[12] H. H. González-Banos and J.-C. Latombe, "Navigation strategies for exploring indoor environments," *The International Journal of Robotics Research*, vol. 21, no. 10-11, pp. 829–848, 2002.

[13] A. Singh, "Nonmyopic adaptive informative path planning for multiple robots," *Center for Embedded Network Sensing*, 2009.

[14] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 3212–3217.

[15] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, 1994.

[16] G. Barequet and S. Har-Peled, "Efficiently approximating the minimum-volume bounding box of a point set in three dimensions," *Journal of Algorithms*, vol. 38, no. 1, pp. 91–109, 2001.

[17] (2015) Point Cloud Library (PCL) website. [Online]. Available: http://pointclouds.org.

[18] (2014) Household object model repository. [Online]. Available: https://github.com/carlosjoserg/ObjectDB.

APPENDIX

Algorithm 1 Active Object Recognition Framework

Input: A sequence of sensor measurements of depth map $X_s^{1:t}$, a sequence of previous robot actions $a^{0:t-1}$, and object database containing point clouds $\{\hat{X}_i\}$ of n objects

Output: Robot base/torso motion or grasping command a^t

[Step 1] Segmentation of sensor measurement

$X_p^t \leftarrow$ *Segmentation*(X_s^t)

▷ X_p^t: partial point cloud segment of target object

$O^t \leftarrow$ *ObstaclePrimitve*$(X_s^t - X_p^t)$

▷ O^t: parametric model for obstacle shape

[Step 2] Registration of database to measurement

for each i in 1 to n **do**

  if $t = 1$ **then**

    $X_i^t \leftarrow$ *RansacInitialAlignment*$(\hat{X}_i, X_p^t)$

    ▷ X_i^t: registered $\hat{X}_i$ in world frame

  else

    $X_i^t \leftarrow$ *MotionUpdate*(X_i^{t-1}, a^{t-1})

  $X_i^t \leftarrow$ *IterativeClosestPoint*(X_i^t, X_p^t)

[Step 3] Belief state update

if $t = 1$ **then**

  for each i in 1 to n **do**

    $\{\phi_{ij_i}\} \leftarrow$ *HypothesisGeneration*(X_p^t, X_i^t)

    ▷ ϕ_{ij_i}: hypothesis of j_i-th instance of i-th object

  $N \leftarrow \sum_{i=1}^{n} \sum_{j_i=1}^{m_i} 1$

  for each i in 1 to n and j_i in 1 to m_i **do**

    $p(\phi_{ij_i}) \leftarrow 1/N$

for each i in 1 to n **do**

  $\{X_{ij_i}^t\} \leftarrow$ *PointCloudInstance*$(X_i^t, \{\phi_{ij_i}\})$

for each i in 1 to n and j_i in 1 to m_i **do**

  $e_{ij_i}^X \leftarrow$ *ErrorFunction*$(X_{ij_i}^t, X_p^t)$

  $w_{ij_i} \leftarrow 1 - \min(e_{ij_i}^X / e_{thr}, 1)$

  $w_{ij_i} \leftarrow$ *CollisionDetection*$(X_{ij_i}^t, X_s^t, X_p^t, w_{ij_i})$

  $w_{ij_i} \leftarrow$ *VisibilityViolation*$(X_{ij_i}^t, X_p^t, O^t, w_{ij_i})$

  $p(\phi_{ij_i}) \leftarrow w_{ij_i} p(\phi_{ij_i})$

$f^t \leftarrow 1 - \sum_{i=1}^{n} \sum_{j_i=1}^{m_i} p(\phi_{ij_i})$ ▷ f^t: utility function

[Step 4] One-step action expansion

if $f^t \geq Q$ **then** ▷ Q: threshold for termination

  $\hat{\phi} \leftarrow \text{argmax}_{\phi_{ij_i}} p(\phi_{ij_i})$

  return *GraspAction*$(\hat{\phi})$

for each k in 1 to l **do**

  for each i in 1 to n and j_i in 1 to m_i **do**

    $\tilde{X}_{ij_i}^{t+1} \leftarrow$ *MotionUpdate*$(X_{ij_i}^t, a_k)$

    $\tilde{O}^{t+1} \leftarrow$ *MotionUpdate*(O^t, a_k)

    ▷ $a_k \in \mathbb{A}$: base/torso action

    $\tilde{X}_p^{t+1} \leftarrow$ *VirtualOcclusion*$(\tilde{X}_{ij_i}^{t+1}, \tilde{O}^{t+1})$

    $\tilde{f}_{ij_i}^{t+1} \leftarrow$ *Step2-3*$(\tilde{X}_p^{t+1}, \tilde{O}^{t+1})$

    $\tilde{p}_{ij_i} \leftarrow 1/N$

  $\Delta_{f_k} \leftarrow \sum_{i=1}^{n} \sum_{j_i=1}^{m_i} \tilde{p}_{ij_i} (\tilde{f}_{ij_i}^{t+1} - f^t)$

  ▷ Δ_{f_k}: expected marginal utility for action a_k

[Step 5] Action selection

$a^t \leftarrow \text{argmax}_{a_k} (\Delta_{f_k} / \textit{ActionCost}(a_k))$

return *BaseTorsoAction*(a^t)

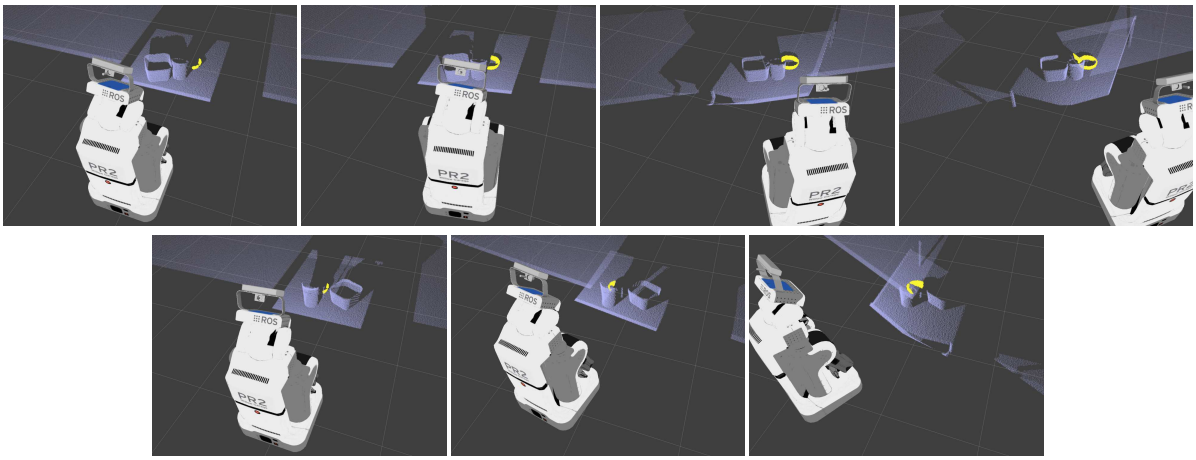


Fig. 6. Experiment results for scene 1 (upper) and scene 2 (lower) using adaptive-greedy algorithm.

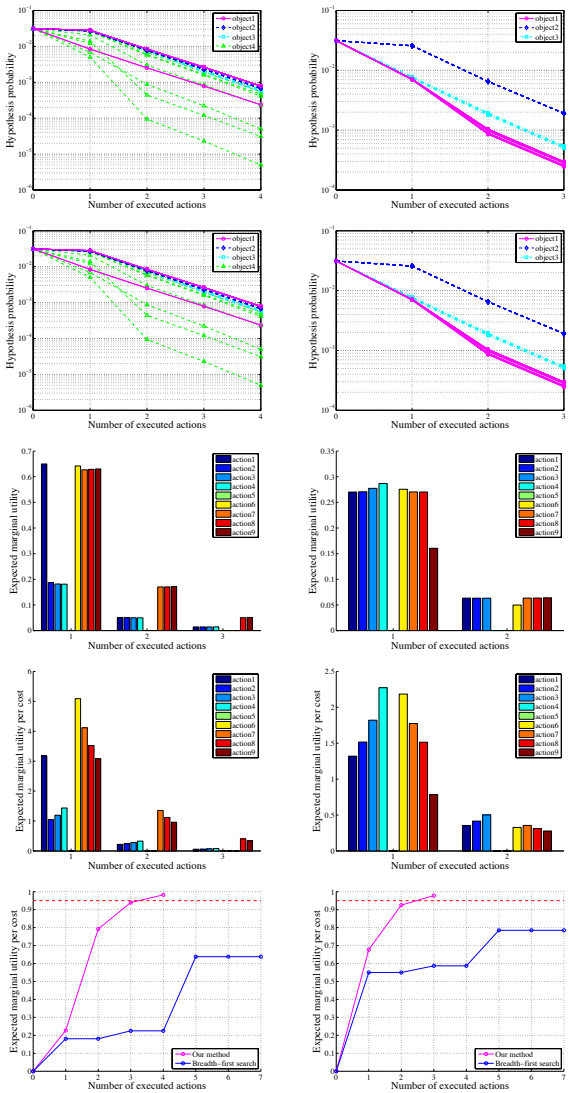


Fig. 7. Experiment results for scene 1 (left) and scene 2 (right) using adaptive-greedy algorithm, compared to breadth-first search (BFS) algorithm. Action(i) stands for base action index i.

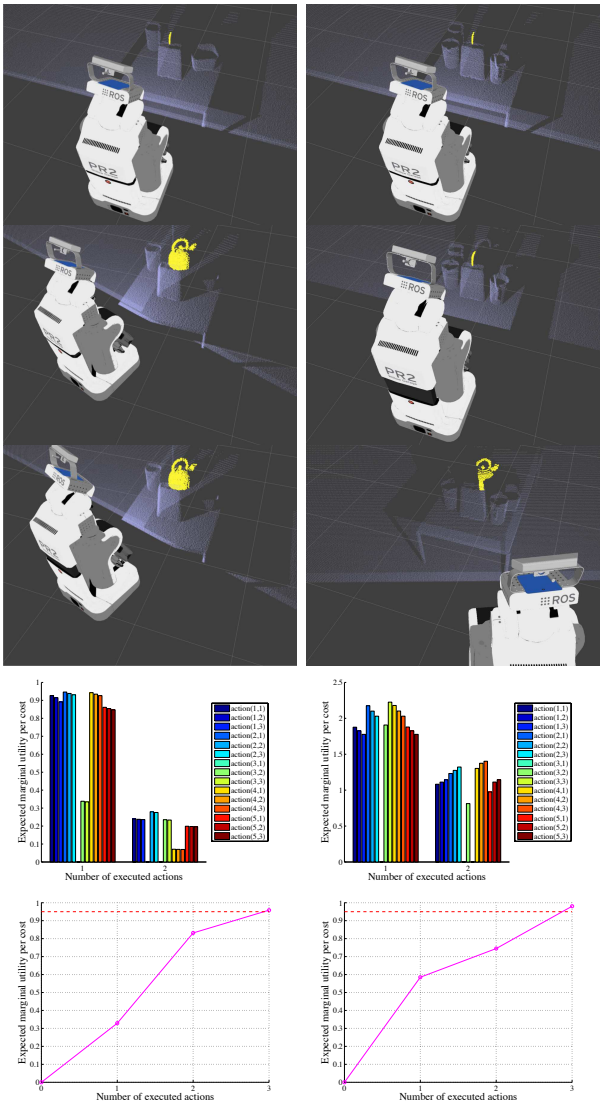


Fig. 8. Experiment results for scene 3 (left) and scene 4 (right) using adaptive-greedy algorithm. Action(i, j) stands for base action index i and torso action index j.