A Task-Oriented Approach for Retrieving an Object from a Pile

Vinay Pilania, Dale M^{c}Conachie, Brent Griffin, Jason J. Corso, and Dmitry Berenson

***Abstract*— We present a framework for object retrieval tasks in extremely cluttered environments. The core of the framework, the Next Best Option (NBO) planner, plans for the (task oriented) next best object to be removed from a pile in order to facilitate the quick retrieval of the target object. The plan of removal of objects from the pile is intended to a) reveal more graspable area of the target, b) find out if an object is underneath another adjacent object. The latter is captured by constructing a tree based on the RGB and depth segmentation of the current scene. While for the former, first either ICP or a modified version of TPS-RPM, a non-rigid registration algorithm, is used to estimate the occluded portion of the target object. The estimated shape is then used to compute the graspable area that will be revealed by the removal of each object. We provide experimental results using real sensor data that show our planner takes fewer actions to finish the task as compared to a baseline planner.**

I. Introduction

Many applications in domestic service robotics and industrial robotics would benefit from a method which allows retrieving objects from a pile. For instance, imagine a domestic service robot retrieving a specific toy from a bin of other toys. However, this task is very difficult because 1) the target object may be severely occluded by other objects (or even not visible); and 2) the objects blocking access to the target may be in a complex arrangement, where deciding what to remove in what order is not trivial. The blocking objects could also be occluding each other.

We seek to develop a framework that addresses tasks like those above, which requires integrating perception, shape estimation, task planning, and manipulation algorithms. Our approach starts by segmenting an RGBD image of the pile into regions which represent different objects. If the target object is not visible we first determine the list of top objects and among them the object with larger surface area is selected for removal. If any part of the target is visible we first estimate the location of the occluded parts of the object as best as possible given the available information. We then search over grasps for the target, finding the grasp that would be valid if the target were unoccluded which has minimal overlap with occluding obstacles. We then seek to remove the obstacles that occlude this grasp.

To remove grasp-occluding obstacles we compute the occlusion relationships between all objects relevant to the task; i.e. which objects are on top of which other ones. These occlusion relationships are then used to construct a tree rooted at the target object with nodes at the other objects.

Electrical Engineering and Computer Science Department, University of Michigan, Ann Arbor, Michigan, 48109, USA
vpilania, dmcconac, griffb@umich.edu;
jjcorso, berenson@eecs.umich.edu

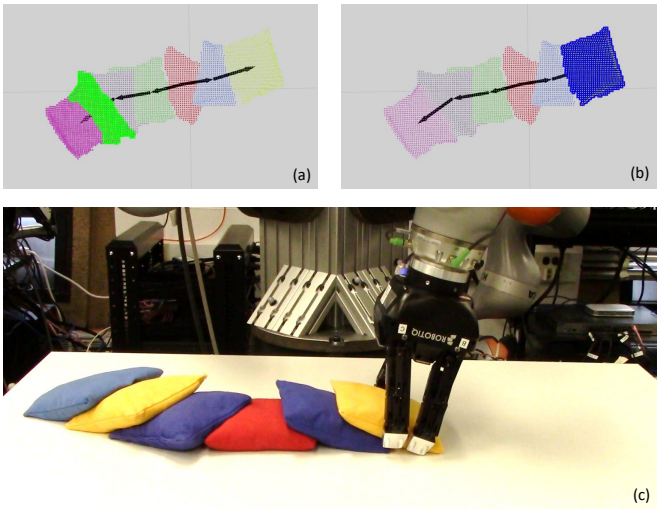

Fig. 1. Example of the NBO planner building a tree representing the scene. (a) Considering two adjacent regions to determine which region is on top. (b) Selecting the next best option to maximize the graspable area. (c) Removing the selected object.

The tree specifies which object should be removed before which other one. Our Next Best Option (NBO) planner then selects the path which will reveal the most graspable area on the target. We then execute the first action in this sequence. However, we do not assume that execution will always succeed, as the grasps performed cannot be guaranteed to be reliable when we do not know occluded parts of the objects. After each action is executed, we re-perceive the environment and re-plan the sequence of actions.

The key contributions of our work are a way to infer occluded portions of the target object and to plan a sequence of actions based on the real-world visibility and graspability constraints of the robot. This distinguishes our work from the planning literature for domains like the classic Blocksworld [1], where constraints on what actions can be done are manually-specified and perception is not considered.

However, because the problem we consider is extremely difficult—requiring near-human capabilities of perception, reasoning, and manipulation in the general case—we restrict our domain so that we are able to work with current algorithms for perception and grasping. First, our tests consider only solid-color bean-bag objects. These are chosen because they make segmentation tractable and because there is no risk of breaking them if we make a manipulation error. Second, our approach to grasping considers only grasps that are directly computed from the point cloud of the object. In the future, we will use methods such as [2], which can be

trained to perform grasps more robustly. Nevertheless, we believe that the integration of these methods along with our approaches for reasoning about occlusions are a significant step forward for robot systems.

We evaluated our method on scenarios consisting of single-color bean bags in piles. Our experiments were performed with real RGBD sensor data and a KUKA robot arm. We found that our method outperforms a naive baseline planner that picks objects from the top in terms of the number of actions.

II. RELATED WORK

In this section, first, we review the related work on task planning, perception, non-rigid registration techniques.

A. Task and Manipulation Planning

Task planning for piles of objects was most notable considered as the Blocksworld domain [1]. Researchers in Artificial Intelligence used this domain as a test-bed for symbolic planning algorithms, but the constraints on which blocks are manipulable were manually defined and perception was not considered. Our work computes these constraints automatically.

Several methods for sequencing manipulation tasks in cluttered environments have been developed [3], [4], [5], [6]. These methods considered geometric constraints in deciding the sequence of objects to manipulate. However, the entire scene is assumed to be known, which is not realistic for manipulating objects in a pile, where perception constraints are key.

B. Perception

A similar perception problem to ours exists in the domain of bin-picking, where a robot must retrieve a specific part from a pile of parts. Perception algorithms for bin-picking [7], [8], [9] generally seek to register a CAD model of the target object to the scene. However, we use a segmentation approach in our work because our objects are more readily identified by color. Regardless, such perception approaches could be easily integrated into our framework if necessary for a set of objects.

Perception can also be an active process. Recent work [10], [11] has explored using interaction with the environment to infer object boundaries and kinematic models. We do not currently consider active perception as part of framework, but we seek to integrate such methods in the future.

C. Non-rigid Registration

To estimate the shape of a partial-occluded object in a pile, we use techniques from non-rigid registration. Iterative Closest Point (ICP) [12] is a popular rigid registration algorithm, however, it (and other variants) does not account for deformation, as can occur with objects like bean bags. A class of methods like TPS-RPM [13] use Gaussian mixture models and a kernel function to deal with deformation. [14] uses Delaunay triangulation and TPS-RPM to deal with outlier rejection. In contrast to earlier approaches CPD [15] is agnostic to the transformation model. Though these approaches deal with noise and other factors, the performance of these methods deteriorates for large occlusions. We modified TPS-RPM to better handle occlusions in our application by accounting for points that should not be visible from the camera pose.

III. PROBLEM STATEMENT

We use r_i to represent i^{th} region (object) in the RGB and depth based segmentation of the current scene. Let $R = \{r_i\}$ denote the set of n regions. We assume that the geometrical shape of target object is known and denoted by T. Let $b_j = \{r_t, r_1, ..., r_k\}$ denote the j^{th} branch of a tree rooted at the target region r_t. Let $T_e = \textsc{Registration}(T, r_t)$ denote the estimation of target region shape using a registration algorithm. If the target object is not occluded then T_e is basically same as r_t.

Given R and T, the objective of our NBO algorithm is to find a region ${}^*r_i$ such that

$$ {}^*r_i = \max_{r_i, b_j} Overlap\{{}^*BBox, b_j\} \tag{1} $$

where r_i is the first graspable region along the branch b_j from root node r_t and

$$ {}^*BBox = \min_{x,y,\theta} Overlap\{R \cap T_e, BBox(x, y, \theta)\} \tag{2} $$

$$ \text{subject to } ValidGrasp(BBox(x, y, \theta), T_e) \tag{3} $$

In the above equation, $BBox$ denotes a bounding box of length l, width w and positioned at (x, y, θ) which is constrained by T_e. Routines `Overlap` computes the overlap area between two regions, and `ValidGrasp` validate the grasping of T_e at a bounding box pose, assuming that there are no obstacles. This validity check is used to define a set of potential grasp locations; we then select a single grasp from this set which has minimum overlap with other objects in the scene. In this way we define an bounding box $*BBox$ which we must then make obstacle free. If this bounding box is obstacle free, then we expect that we can grasp and retrieve the target object, even if other portions of the target object are still occluded. Our implementation of `ValidGrasp` is discussed in Sec. IV-D.

IV. THE NBO ALGORITHM

In this section, we provide our Next Best Option algorithm that takes the segmentation (see Section V) of current scene, index `tindex` of target region if the target object is detected and segmented out by the segmentation algorithm and the template (geometrical information, for example, in term of point cloud) of the target object as inputs and plans for the best object (region) to be removed from the piles of stuff (all the segmented regions). Our NBO is iteratively called until the target object is searched, i.e., get current scene segmentation, invoke NBO, remove the object and repeat the cycle. This section describes the one NBO call.

First, we state the following routines: $\textsc{RegionWithMaxArea}(\{r_i\})$ takes a set of regions as input and returns the region with maximum surface area

which is approximated by the number of points in the region. The validity of a region to be graspable in the presence of obstacles is carried out using VALIDGRASPOBST($R, tindex$) which takes all regions and the index `tindex` of considered region as inputs. This algorithm estimates if the given region $R[tindex]$ can be grasped and removed by the robot; it is described in more detail in Sec. IV-D. B(r) computes the boundary points of a region r [16]. CLOSEBPOINTS($\{p\}^r, \{p\}^{r_i}$, ADJREGIONDISTTH) takes two sets of points corresponding to region r and region r_i as inputs and returns the points in these sets which are with in the distance threshold of ADJREGIONDISTTH. REGIONPOINTS($\{p\}^r$, REGIONINSIDEDISTTH) takes some points (not all, for example boundary points) of a region r as input and returns all the nearest neighbours (within REGIONINSIDEDISTTH) for them from the same region. Finally, routine CENTROID($\{p\}^r$) computes the centroid of a region r (with points $\{p\}$ in 3D) and returns the z component of the centroid.

A. Algorithm Description

The NBO algorithm is described in Algorithm 1. First, `tindex` is checked and if `tindex`$= \emptyset$, i.e., the target object is not visible or not segmented out in the current scene, then the selection of next best object to be removed is decided as follows (lines 1-3): routine TOPADJACENTREGION, explained in Algorithm 2, is used to compute the top regions $\{r_i^{top}\}$ and the region with maximum surface area is reported as the best object for removal (denoted by *mRegion* for manipulation region). However, if the target object is segmented out and graspable, the NBO simply reports it. This is done in lines 4-5.

If the target object is visible but not graspable then a detailed procedure is carried out to search for the best object. In order to plan intelligently, it is important for the search mechanism to know where the occluded portion of the target object is residing. A modified registration algorithm (explained in Section IV-C) is used to estimate the occluded portion that takes the known geometrical shape T of the target object to match with the reported target region $R[tindex]$ (line 7). A bounding box $BBox$ that captures the grasping affordance is used to find a valid grasp point in $R[tindex]$ such that it minimizes the overlap of $BBox$ centered at a valid grasp pose with T_e and R (excluding target region r_t). Considering the grasp affordance and removing obstacles to ensure the grasp is accessible allows us to focus on removing only the obstacles that are necessary for grasping; i.e. we do not need to expose the entire target object to grasp it. $^*BBox$ (computed using Equation 2) is used to find the overlap area for each region which is a deciding factor in the cost metric (please see Equation 1).

After obtaining the occluded portion of the target region and the overlap of each region with $^*BBox$, the planning is carried out using a tree structure that connects the layout of different regions in term of adjacency and underneath, i.e., finding all the adjacent neighbours for a region which are lying on the top of it. This is done from lines 9-17. A node structure has three components in order: a region, overlap of the region with $^*BBox$, and a pointer to its parent node. A tree rooted at the target region is constructed. Start node n_s is inserted into a search queue (line 9). At each iteration of the while loop (line 10), a node n is popped out from search queue and nodes with top and adjacent regions are added into the tree if there are no cycles. Based on our cost metric (see Equation 1), the first graspable region (object) along the branch that has maximum overlap with $^*BBox$ is reported as the next best object for removal. We now describe routine TOPADJACENTREGION.

B. Computation of Top And Adjacent Regions

This routine (explained in Algorithm 2) takes all the segmented regions R as input. For each region (excluding the background and the region r_i under consideration), boundary points are computed and compared with the boundary points of a region r_i. Only those boundary points $\{\{p\}_c^r, \{p\}_c^{r_i}\}$ from both the regions are retained which are close enough based on the distance threshold of ADJREGIONDISTTH (lines 2-3). Furthermore, all the points in region r which are lying close to $\{p\}_c^r$ (i.e., close to retained boundary points of r), based on another distance threshold REGIONINSIDEDISTTH, are selected. We do the similar filtering for region r_i and $\{p\}_c^{r_i}$. Based on this, we now have point sets $\{p\}_{ins}^r$, $\{p\}_{ins}^{r_i}$ for regions r and r_i, respectively. From lines 6-7, we compute the z component of the centroid of point sets $\{p\}_{ins}^r$, $\{p\}_{ins}^{r_i}$ and use it to decide (line 8) if region r is adjacent and on the top of region r_i.

C. Registration Method

The registration method that we use to estimate the geometry of the target object is described in Algorithm 3. Please note that in registration context, we use `fixed` and `movable` terms for referring to two point sets T and r. Also, we use boundary points for alignment instead of full movable and fixed point sets (line 1). However, at the end, we do transform full movable point set using computed transform function. An overview of our registration method is as follows: first, we determine which (rigid or non-rigid) point set matching technique is more appropriate for the current situation. This is because non-rigid registration algorithms do not perform well with occlusion. For that we filter out the boundary points of fixed point set and check if enough points are available to pursue non-rigid registration alignment as we are dealing with deformable objects. This is done in lines 2-4. We use Iterative Closest Point (ICP) [12] augmented with RANSAC [17] based outlier rejection for situations where non-rigid approach is not a good option. However, if there are enough filtered boundary points left then a modified version of TPS-RPM is used for alignment (lines 7-13). Note that the initialization of affine transformation d (defined in next paragraph) in TPS-RPM is based on ICP (line 7).

First, we introduce some notations to clearly outline our modification in the TPS-RPM algorithm. We omit the full details of TPS-RPM and our notation follows that of Chui and Rangarajan [13]. Let V and X denote movable and fixed

Algorithm 1: NBO Algorithm

Input: regions R = $\{r_i\}$, *tindex* $\in \{1, ..., n\}$ or $\emptyset$, T
Output: region ${}^*r_i$

if *tindex* $= \emptyset$ **then**
  $\{r_i^{top}\} := r_i \in R \ni$ TopAdjacentRegion(R, r_i) is $\emptyset$
  mRegion = RegionWithMaxArea$(\{r_i^{top}\})$
else if *tindex* $\neq \emptyset$ **&** ValidGraspObst$(R, tindex)$ **then**
  mRegion = R[*tindex*]
else
  $T_e :=$ Registration$(T, R[tindex])$
  compute ${}^*BBox$ as per Eq. 2 & 3
  $Q \leftarrow n_s := \{$R[*tindex*]$, 0, \emptyset\}$
  while $Q \neq \emptyset$ **do**
    $n :=$Pop(Q)
    $\{index\} =$TopAdjacentRegion$(R, n[r_i])$
    for ***each*** $index$ **do**
      $overlap = Overlap({}^*BBox, R[$*index*$])$
      $n' := \{$R[*index*]$, overlap), n\}$
      $Q \leftarrow Q \cup \{n'\}$
  mRegion = compute ${}^*r_i$ as per Eq. 1
return *mRegion*

Algorithm 2: TopAdjacentRegion()

Input: regions R = $\{r_i\}$, region r_i under consideration
Output: *indices* of top and adjacent regions for r_i

for ***each*** $r \in R$ **do**
  get boundary points $\{p\}^r = B(r)$; $\{p\}^{r_i} = B(r_i)$
  $\{\{p\}_c^r, \{p\}_c^{r_i}\} =$ CloseBPoints$(\{p\}^r, \{p\}^{r_i},$ AdjRegionDistTH)
  $\{p\}_{ins}^r =$ RegionPoints$(\{p\}_c^r,$ RegionInsideDistTH)
  $\{p\}_{ins}^{r_i} =$ RegionPoints$(\{p\}_c^{r_i},$ RegionInsideDistTH)
  $z_{cent}^r =$ Centroid$(\{p\}_{ins}^r)$
  $z_{cent}^{r_i} =$ Centroid$(\{p\}_{ins}^{r_i})$
  if $z_{cent}^r > z_{cent}^{r_i}$ **then**
    indices$\leftarrow$ index of r in R
return *indices*

Algorithm 3: Registration()

Input: movable point set T, fixed point set r,
TPS-RPM variables $(t_0, \lambda_1, \lambda_2, annealRate)$,
λ_3, occluding point set $\{R\} - \{r\}$
Output: aligned movable point set T_e

get boundary points $\{p\}^T = B(T)$; $\{p\}^r = B(r)$
$\{p\}_{filtered}^r =$ remove points in $\{p\}^r$ close to directly occluding regions using threshold AdjRegionDistTH
if *size*$(\{p\}_{filtered}^r) <$ TPSMinPointsTH **then**
  T_e = ICP(T, r)
else
  Initialize parameters $t_0, \lambda_1, \lambda_2, \lambda_3, annealRate$
  Initialize parameters $M, f(d, w)$
  Movable $V := \{p\}^T$, Fixed $X := \{p\}_{filtered}^r$
  for $i = 1$ **to** NumIterations **do**
    Compute correspondence matrix M
    Compute new fixed point set Y
    Update transformation parameters d, w
  T_e = $f(V, d, w)$
return T_e

point sets with N and K points, respectively. M denotes a correspondence matrix of size NXK and m_{ai} denotes the correspondence probability of a^{th} point v_a in V with i^{th} point x_i in X. Let $f(v_a, d, w)$ denote the mapping function, where d and w represent affine transformation and warping coefficient matrices, respectively. Our modifications (lines 10 and 11, Algorithm 3) to the original method are as follows:

1) Compute Correspondence Matrix M: Let d_1 and d_2 denote the distance of transformed movable point $f(v_a)$ to the nearest point in fixed point set and occluding point set (see input in the algorithm), respectively. $d_1 > d_2$ implies that the transformed movable point should match to some point (nearest one) in occluding point set. Therefore, we put $m_{ai} = 0$ for all i from 1 to K. Otherwise, we use the same approach for assigning m_{ai} as in TPS-RPM.

2) Compute New Fixed Point Set Y: In TPS-RPM, $y_a = \sum_{i=1}^{K} m_{ai} x_i$ for all a. However, in our case (only if $d_1 > d_2$), $y_a = c - \lambda_3 \hat{n}$, where c is the nearest occluding point to $f(v_a)$ and $\hat{n}$ is the normal vector at c.

D. Grasping

The key idea behind our grasping algorithm is edges; if the target object has sufficiently strong edges within the gripper bounding box, then we can grasp the object. To apply this edge detection, we first reduce the target estimate to an image in the X-Y plane, and then generate a edge detecting kernel $I_{f,e}$ that matches the dimensions of our gripper (w, l), at the given local (x, y, θ). We can then measure the response of the filter to the projected target estimate in order to determine if the edges are sufficiently strong. In addition to an edge detector, we also pad the area around the grippers with negative values in order penalize locations where errors in sensing and actuation can degrade the grasp quality. An example of our filter and response at a given orientation are shown in Fig. 3. If the response at a particular (x, y, θ) is beyond a given threshold GraspQualityTH then the grasp is valid as shown in Alg. 4.

In order to account for obstacles in ValidGraspObst (Alg. 5), we create two additional images, I_o which records the presence or absence of other objects, and $I_{f,o}$ which penalizes the presence of obstacles within the grasp box. As with ValidGrasp we pad the grasp box to penalize obstacles that are nearby to account for noise in the system, then threshold on the response to determine if a specific grasp is valid. Iterating over a discretized set of grasp poses as in (2) and (3), we can estimate if a given region is graspable, even with the presence of other objects around it.

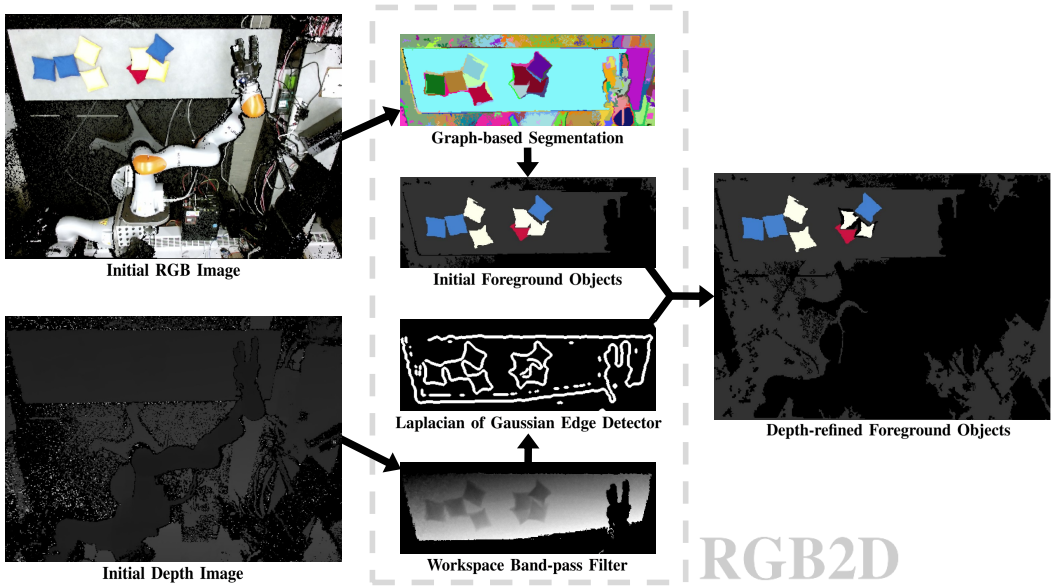

Fig. 2. RGB2D Framework

Algorithm 4: VALIDGRASP()

Input: Target estimate T_e, bounding box $BBox(x, y, \theta)$
Output: true/false

I_e = REDUCETOIMAGE(T_e)
$I_{f,e}$ = GENERATEGRIPPERFILTER(x, y, θ)
response = $dot(I_e, I_{f,e})$
return *response* $\geq$ *GRASPQUALITYTH*

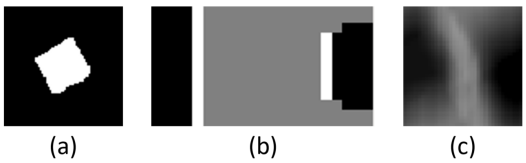

Fig. 3. Grasping edge detection example. (a) Target estimate T_e reduced to a XY image. (b) The grasp filter $I_{f,e}$ which looks for an edge on the sides of the grasp box. (c) The final result, convolved across the input target estimate. The stronger the response, the more robust the grasp pose.

V. RGB2D OBJECT SEGMENTATION

RGBD data is collected from an overhead Microsoft Kinect and integrated into ROS using [18], [19]. Our segmentation algorithm initializes using the efficient graph-based **RGB** image segmentation method of [20], then proceeds **to** our novel **D**epth-based refinement process (**RGB2D**). The depth-refinement process separates objects that are initially grouped based on spatial proximity and color but also exhibit a detectable depth boundary (e.g., the two rightmost yellow bean bags in Figure 2). Depth boundaries are detected using a Laplacian of Gaussian edge detector on a band-pass filtered depth image (i.e., the depth image is offset and scaled given the bounds of the robot's workspace). Finally, segments are classified as foreground or background objects based on the difference between their mean RGB values ($\in \mathbb{R}^3$) and the RGB descriptions for objects of interest. The RGB2D framework is depicted in Figure 2.

Algorithm 5: VALIDGRASPOBST()

Input: Regions R, $tindex \in \{1, ..., n\}$
Output: true/false

$Obst = \emptyset$
for ***each*** $r \in R \setminus R[tindex]$ **do**
  $Obst = Obst \cup r$
I_o = REDUCETOIMAGE($Obst$)
for ***each*** (x, y, θ) **do**
  $I_{f,o}$ = GENERATEOBSTFILTER(x, y, θ)
  response = $dot(I_o, I_{f,o})$
  if *response* $\geq$ *GRASPOBSTTH* **&** *VALIDGRASP*($R[tindex], (x, y, \theta)$) **then**
    return *true*
return *false*

VI. EXPERIMENT SETUP AND RESULTS

To evaluate the performance of our planner, we test it against a simple baseline planner, using identical perception and manipulation systems. To compare the effectiveness of these planners, we record the total number of actions taken to move the target object (red bean bag) to a pre-specified location and average these results across 5 trials. These planners are tested on two example tasks (Fig. 4 and 5); a 4-square example where minor changes in the experiment setup are the primary difference in the performance of the algorithms, and a more complicated hierarchical line example where we can see the explicit benefit of our approach.

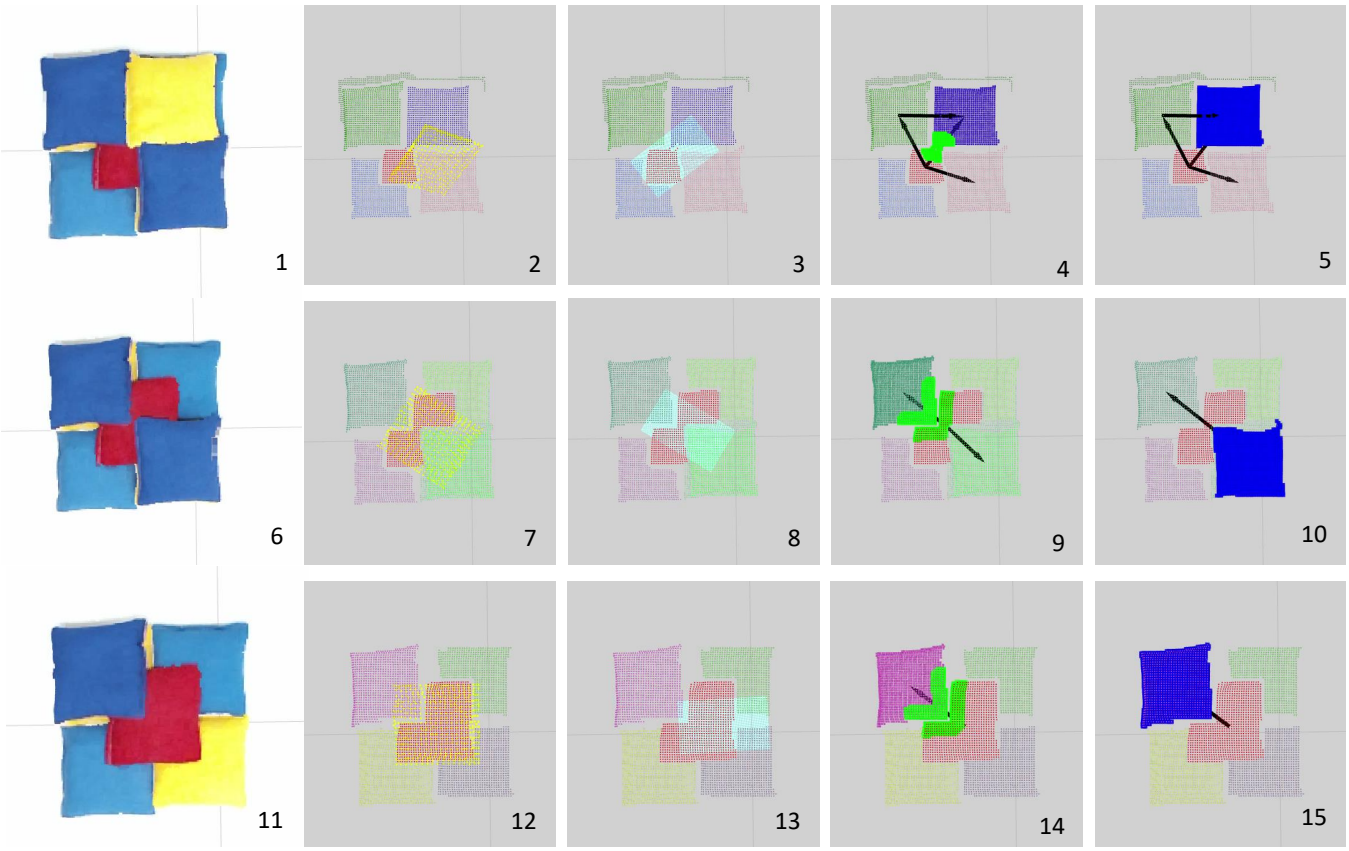

Fig. 4. Sequence of screenshots showing planning process for 4 square task. First column: view from the Kinect sensor. Second column: segmentation and target estimation (in yellow) (Alg. 3). Third column: best bounding box to be uncovered (in teal). Fourth column: points (in green) being considered by TOPADJACENTREGION (Alg. 2). Fifth column: final region (in blue) selected for removal

Algorithm 6: Baseline Algorithm

Input: regions $R = \{r_i\}$, *tindex* $\in \{1, ..., n\}$ or $\emptyset$, T
Output: region ${}^*r_i$

if *tindex* $\neq \emptyset$ **&** VALIDGRASPOBST$(R, tindex)$ **then**
    mRegion = R[*tindex*]
else
    $\{r_i^{top}\} := r_i \in R \ni$ TOPADJACENTREGION(R, r_i) is $\emptyset$
    mRegion = REGIONWITHMAXAREA$(\{r_i^{top}\})$
return *mRegion*

TABLE I
PARAMETER TABLE

ADJREGIONDISTTH	0.015
REGIONINSIDEDISTTH	0.02
TPSMINPOINTSTH	15
t_0	0.001
λ_1	12
λ_2	0.01
λ_3	0.0098
annealRate	0.98
GRASPQUALITYTH	0.8
GRASPOBSTTH	0.75

A. Baseline Planner

For our baseline algorithm, we use a simple planner (Alg. 6) which removes the target region if it is visible and graspable, otherwise it uses the routine TOPADJACENTREGION to compute the top regions and among them the region with the most area is chosen for removal.

B. Results

The first example task (Fig. 4) starts with the red target bean bag occluded, and requires at least 3 other bean bags to be removed before there is the possibility of grasping the target. Minor deviations in the starting position of the bags could lead to a 4th bag needing to be removed before the task could be completed. Because both planners need at least 3 bags to be removed before the target can be grasped, there is very little difference in their performance.

The second example (Fig. 5) shows a very different result, where the baseline planner typically has to remove all the other bean bags before the target is revealed enough to be grasped. In contrast, the NBO planner is able to select a single branch to be removed rather than alternating between them, significantly reducing the number of actions needed.

VII. CONCLUSION

This paper presented a framework for object retrieval tasks in piles. We used RGBD segmentation to find objects in the

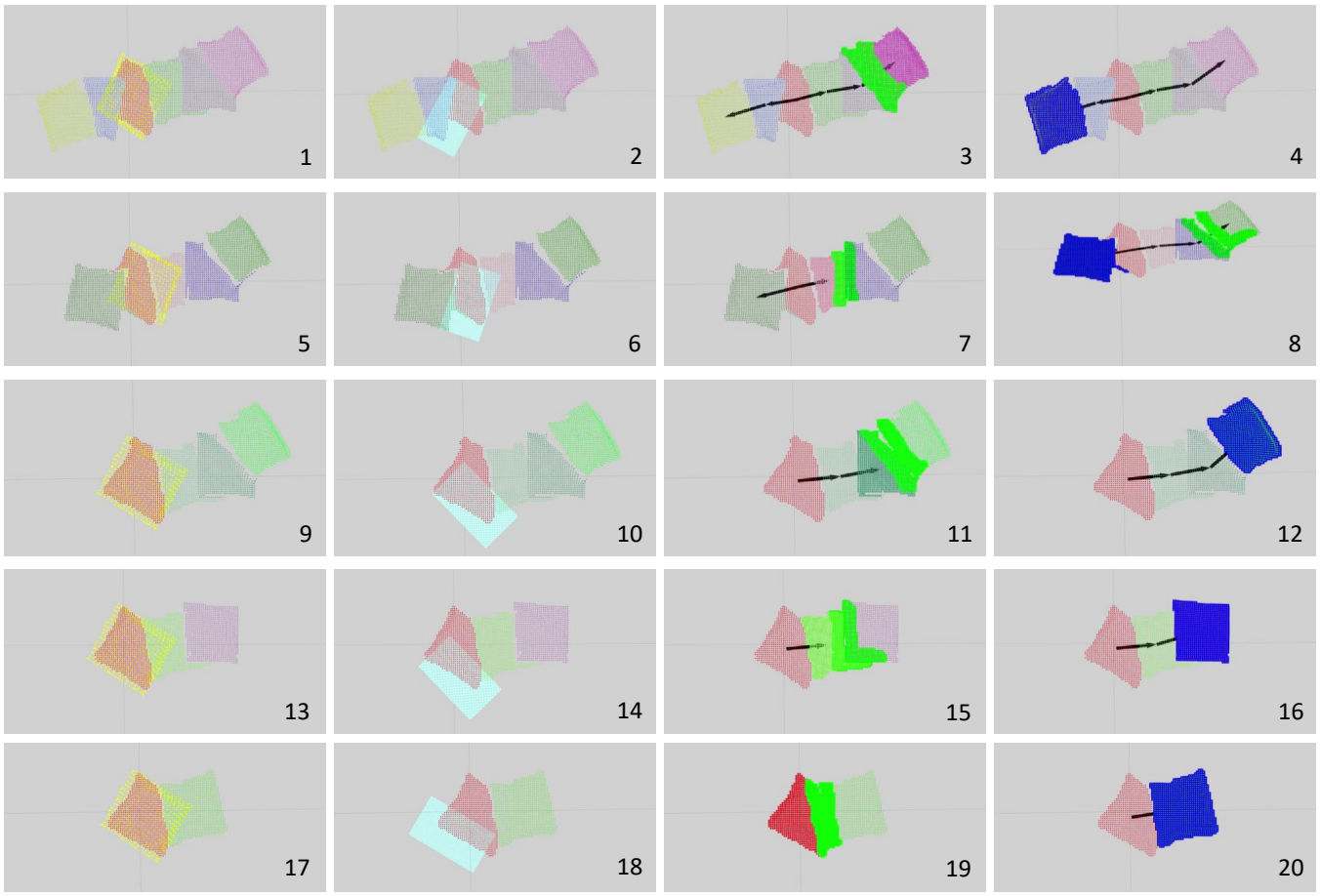

Fig. 5. Sequence of screenshots showing planning process for hierarchical line task. First column: segmentation and target estimation (in yellow) (Alg. 3). Second column: best bounding box to be uncovered (in teal). Third column: points (in green) being considered by TOPADJACENTREGION (Alg. 2). Fourth column: final region (in blue) selected for removal

TABLE II

AVERAGE ACTIONS TAKEN FOR EACH EXPERIMENT, ACROSS 5 TRIALS PER ALGORITHM

	Baseline Planner	NBO Planner
Four Square	4.8	4.4
Hierarchical Line	5.0	2.8

scene and computed occlusion relationships among obstacles and between obstacles and the target. Our NBO planner was used to find a sequence of removal actions to make a grasp on the target accessible. Our results on a KUKA robot arm showed that our method is more efficient than a naive baseline. In future work we seek to improve the perception and grasping algorithms, as well as to explore active perception methods for piles.

REFERENCES

[1] J. Slaney and S. Thibaux, "Blocks world revisited," *Artificial Intelligence*, vol. 125, no. 1, pp. 119 – 153, 2001.

[2] M. Gualtieri, A. ten Pas, K. Saenko, and R. Platt, "High precision grasp pose detection in dense clutter," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016.

[3] M. Stilman, J.-U. Schamburek, J. Kuffner, and T. Asfour, "Manipulation planning among movable obstacles," in *Robotics and Automation, 2007 IEEE International Conference on*, 2007, pp. 3327–3332.

[4] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 1470–1477.

[5] S. Cambon, R. Alami, and F. Gravot, "A hybrid approach to intricate motion, manipulation and task planning," *The International Journal of Robotics Research*, vol. 28, no. 1, pp. 104–126, 2009.

[6] A. Krontiris and K. E. Bekris, "Dealing with difficult instances of object rearrangement." in *Robotics: Science and Systems*, 2015.

[7] K. Ikeuchi, "Generating an interpretation tree from a cad model for 3d-object recognition in bin-picking tasks," *International Journal of Computer Vision*, vol. 1, no. 2, pp. 145–165, Jun 1987.

[8] M. Nieuwenhuisen, D. Droeschel, D. Holz, J. Stckler, A. Berner, J. Li, R. Klein, and S. Behnke, "Mobile bin picking with an anthropomorphic service robot," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 2327–2334.

[9] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, Y. Taguchi, T. K. Marks, and R. Chellappa, "Fast object localization and pose estimation in heavy clutter for robotic bin picking," *The International Journal of Robotics Research*, vol. 31, no. 8, pp. 951–973, 2012.

[10] H. van Hoof, O. Kroemer, and J. Peters, "Probabilistic segmentation and targeted exploration of objects in cluttered environments," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1198–1209, Oct 2014.

[11] D. Katz and O. Brock, "Manipulating articulated objects with interactive perception," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 2008, pp. 272–277.

[12] P. J. Besl, N. D. McKay, *et al.*, "A method for registration of 3-d shapes," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[13] H. Chui and A. Rangarajan, "A new point matching algorithm for non-rigid registration," *Computer Vision and Image Understanding*, vol. 89, no. 2-3, pp. 114–141, Feb 2003.
[14] D. Pizarro and A. Bartoli, "Feature-based deformable surface detection with self-occlusion reasoning," *International Journal of Computer Vision*, vol. 97, no. 1, pp. 54–70, March 2012.
[15] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2262–2275, December 2010.
[16] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *Robotics and automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–4.
[17] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, June 1981.
[18] T. Wiedemeyer, "IAI Kinect2," https://github.com/code-iai/iai_kinect2, Institute for Artificial Intelligence, University Bremen, 2014 – 2015, accessed June 12, 2015.
[19] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 1–4.
[20] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, Sep 2004.