Guided Pushing for Object Singulation

Tucker Hermans James M. Rehg Aaron Bobick

***Abstract*— We propose a novel method for a robot to separate and segment objects in a cluttered tabletop environment. The method leverages the fact that external object boundaries produce visible edges within an object cluster. We achieve this singulation of objects by using the robot arm to perform pushing actions specifically selected to test whether particular visible edges correspond to object boundaries. We verify the separation of objects after a push by examining the clusters formed by geometric segmentation of regions residing on the table surface. To avoid explicitly representing and tracking edges across push behaviors we aggregate over all edges in a given orientation by representing the push-history as an orientation histogram. By tracking the history of directions pushed for each object cluster we can build evidence that a cluster cannot be further separated. We present quantitative and qualitative experimental results performed in a real home environment by a mobile manipulator using input from an RGB-D camera mounted on the robot's head. We show that our pushing strategy can more reliably obtain singulation in fewer pushes than an approach, that does not explicitly reason about boundary information.**

I. Introduction

The ability to detect previously unseen objects and separate them from surrounding objects holds great value for autonomous robots operating in household environments. This problem, termed object singulation, provides the most benefit by allowing a robot to segment novel objects of potential interest from the scene. This is especially true for a robot operating in a home, which will experience a wide variety of objects during its lifespan.

Singulation provides many benefits to a general purpose mobile manipulator beyond the identification of previously unknown objects. Separating objects form one another can be used to assist grasping methods by creating fewer obstacles for an arm to navigate, while providing more clearance for the robot's end effector. For uses that desire specific object identification or categorization, separating objects from one another gives recognition systems a better view of the object and provides detection locations avoiding the need to search the entire image.

Additionally, performing object segmentation through pushing allows us to understand objects at the operational level of the behaviors used during singulation. Such an identification is important for tasks such as organizing the objects on a table through pushing, where the available behaviors may not be capable of separating all objects at their semantic level. For example, a basket on a kitchen table may be filled with apples. While object recognition systems could potentially recognize each apple independently, a robot attempting to push the basket does not need this information if the fruit remains inside the basket while its being pushed.

We propose a method for separating an unknown number of objects on a surface by performing selective pushes in such a way as to disambiguate potential object boundaries. Our method is predicated on the idea that such boundaries tend to produce visual edges in captured images. Thus visual edges represent hypotheses of plausible locations, where a single object cluster may split if it is indeed more than one object. Through successive pushes we accumulate evidence that such edges do or do not correspond to object boundaries. We do so without explicitly tracking edges during the push sequence. Instead, we introduce the use of a histogram of orientations as an aggregated representation of potential boundary edges: only the histogram has to be propagated through the push sequence. Furthermore we design a decision process for systematically testing these hypotheses, which is aware of the constraints of the robot and workspace. Figure 1a illustrates an example initial scene for our system to singulate. Figure 1b shows the robot performing a singulation push.

(a) (b)

Fig. 1: Example initial cluttered scene encountered by the robot. Partial singulation result as the robot performs a pushing action.

We proceed in Section II with an investigation of prior research relevant to our work. Section III gives an explanation of our approach, followed by a detailed description of the implementation in Section IV. We give extensive experimental results in Section V including in depth quantitative comparisons and qualitative evaluation for a number of different scenarios. We conclude in Section VI with an overview of our findings and planned future work.

II. Related Work

Object detection constitutes a major research effort in the computer vision community. State of the art object recognition and categorization methods can detect object instances or categories with high precision, but require large training sets enumerating all possible object categories of interest [1].

Tucker Hermans, James M. Rehg, and Aaron Bobick are with the Center for Robotics and Intelligent Machines and The School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA. {thermans, rehg, afb}@cc.gatech.edu

Additionally the execution time of such methods grows with the number of object classes presented to the system. The burden of obtaining training sets large enough to cover all objects a robot may potentially encounter means that alternative methods are needed for determining where objects are in the environment. Attempts at creating generic object detectors have been made, but they still require large training sets and computation time [2]. Methods have also been proposed for determining the manipulation capabilities of novel objects [3–5]. These methods, however, suffer the same issues as object recognition requiring a large training set containing examples similar in appearance to objects with which the robot is expected to interact.

While singulation has been attempted using grasping, this requires dexterous capabilities to perform grasps on unknown objects and is limited in only being able to manipulate objects small enough to be grasped [6–8]. In contrast, nonprehensile pushing actions requires less precision, can be performed using much less capable manipulators, and can operate on a wider range of objects that may be too large to be grasped by the robot. As such, we restrict our discussion to those singulation works relying on pushing, where less dexterity is necessary in the manipulator.

Early work on pushing developed control and planning algorithms relying on a strong understanding of the physics of the object being pushed [9–11]. While pushing has been used for a number of tasks, such as aiding in grasping [12, 13] and pick-and-place tasks [14, 15], our work builds on previous approaches to interactive segmentation [16–20].

The problem of interactive segmentation was introduced by Fitzpatrick and Metta, where a robot makes a sweeping motion inside its view frame and detects the objects that move [16, 21]. The robot detects its arm by calculating optical flow in the scene and segmenting the pixels with flow corresponding to the controlled motion of the arm [16]. Using this initial segmentation the end-effector of the arm is estimated at the farthest point of the arm being moved and any flow that appears in the image beyond the end effector is assumed to be an object moving as result of the sweeping motion. Fitzpatrick extends this work in a graph-cut framework to better segment the object being moved, allowing for textureless regions to be segmented in addition to those locations where optical flow occurs [21]. A limitation of this approach, not present in our work, comes from the assumption that all motion not explained by the robot's arm belongs to a single object. Additionally no method is given for determining where to push, thus pushing actions must exhaustively search the space in order to detect all objects present.

Li and Kleeman use small pushes, termed nudges, to segment symmetric objects [18]. Symmetry lines are detected in stereo cameras and then group to form hypothesis for locations of symmetric objects. For a given hypothesis the robot performs a pushing action that will move the object in the stereo view allowing frame differencing to seed a segmentation constrained by the symmetric property of the object.

Similar to Fitzpatrick's work, Kenney et al. use image differencing to localize the robot arm in the scene and segment motion not belonging to the arm as objects [19]. Templates are built from the resulting object locations allowing the tracking of objects over time. This enables the objects to collide without their identity being lost. However, it still fails to separate objects in cases where multiple objects are in contact prior to the robot discovering them. Furthermore, these templates require highly textured objects to produce good results. Additionally it is assumed another process tells the robot where to initially push.

Katz et al. learn kinematic models of articulated objects through pushing, although segmentation of the objects in the scene is assumed known [17, 22]. More recently this has been extended to extracting 3D models of the objects [23].

The work of Chang et al. is most similar to ours. Chang et al. perform object singulation through pushing of object piles [20]. Groups of objects are first segmented by removing the supporting surface from the input point cloud. Pushes are planned to push through the object centroid in directions avoiding contact with other piles of objects. Rigid body hypotheses are proposed using point correspondences from the images taken before and after the push action and are confirmed by matching in the point cloud. Evidence for singulation is accumulated by consistent strong rigid body matches. Our work differs in that we explicitly form hypotheses for splitting locations in object clusters based on edge locations and orientations and accumulate evidence related to these locations in order to build confidence more quickly. Additionally, we do not rely on texture being present in the objects for correspondences nor do we need to perform pick-and-place operations to clear the workspace of singulated objects.

III. Approach

We base our approach on the fact that object boundaries in the world tend to generate visual edges in captured intensity and depth images. While edges in the depth image give the strongest cues as to the location of object boundaries, we can not rely on them alone as object boundaries often do not appear as depth edges, especially when the objects are in contact. As such we use the complementary evidence provided by intensity edges, where differing appearance information from two neighboring objects creates a strong edge in the intensity image.

Of course, visual edges also arise as the result of other phenomena such as internal texture on an object surface, cast shadows, or specular reflection. While computer vision techniques have been develop that attempt to classify image edges as object boundaries (c.f. [24]), a robot can much more reliably test if an edge corresponds to an object boundary by attempting to separate the objects that would generate such a visual edge. Thus for any edge that corresponds to a potential boundary we determine explicitly where the objects generating the edge would lie in the scene, if the objects exist. Then for a given object hypothesis the robot determines and performs a push action, which will generate

evidence helping to confirm or deny the hypothesis. After each push action we segment the scene in such a way that physically separated objects will belong to separate clusters, but neighboring objects may be grouped together. Thus a push that successfully splits a cluster results in an increased number of clusters from the segmentation procedure. When no new clusters are formed, we have confirming evidence that the cluster may be a single object, but the robot must still test the remaining image edges, to verify that each of the segmented clusters represent singular objects.

A naive application of this approach would attempt to split the clusters at each candidate image edge in order to exhaustively exclude every such edge from being a boundary between two objects. However, this exhaustive approach has a number of shortcomings. First, edge detection is not stable across illumination changes; some edges will appear or disappear after pushes. This causes great difficulty in tracking individual edges over time. More importantly, pushing actions that attempts to cleave a cluster along a particular edge will likely reveal any object boundaries parallel to that edge. As such we approximate this procedure by examining the distribution of boundary candidate orientations associated with each object cluster and accumulate evidence by tracking the history of pushes for each cluster. This allows us to achieve singulation with fewer pushes than the total number of candidate edges in the scene.

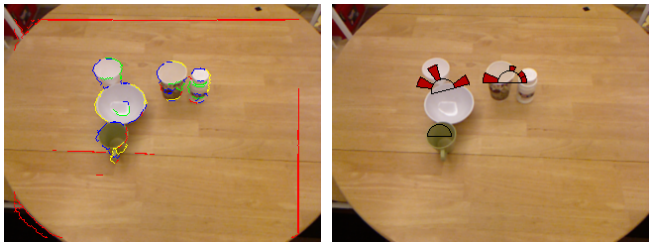
For each object cluster we quantize all candidate edges by orientation into one of n bins, creating a histogram of edge orientations. Each push performed by the robot is similarly quantized and the appropriate bin in the cluster's push history histogram is incremented. By using this quantization we avoid the need to explicitly track image edges and instead must only repopulate the edge orientation histogram after each push. This relies on the ability to track the transformations undergone by the clusters, so that the push history and edge orientation histograms can maintain consistent alignment with the cluster's internal frame of reference over time. A cluster is deemed singulated once all push history bins corresponding to non-empty bins in the boundary orientation histogram have been populated. By recursively applying this procedure to all clusters in the scene the robot asserts that all objects have been separated.

IV. Implementation

This section details our specific implementation of the general approach described in Section III. We first describe the geometric segmentation and boundary detection methods used to form object hypotheses. We then explain how to generate an abstract push vector for a specific hypothesis. After this we introduce a mechanism for explaining the outcome of the pushing action and how the robot can use this explanation to update the push history for each object cluster. Finally, we detail how the robot translates an abstract push into a real-world action.

A. Object Hypothesis Generation

The first step in proposing the current set of objects requires segmenting spatially separate regions from one another. The algorithm takes as input a point cloud of the scene and fits a plane to the supporting surface using RANSAC [25]. We then remove these table points and all points below or beyond the edges of the table from the point cloud. The remaining points correspond to objects supported by the estimated plane. These points are grouped based on the euclidean distance between neighbors into a set of "object clusters."



(a) (b)

Fig. 2: (a) Example extracted and labeled image edges. Green edges correspond to candidate splitting edges. Yellow edges represent boundary edges between an object and free space. Blue edges have too few 3D points to stably estimate a 3D line, while red edges do not overlap with object clusters. (b) Boundary orientation histograms associated with the current object clusters.

Each resultant cluster contains at least one separate object. In order to identify potential objects within these clusters we extract edges from the intensity and depth images and associate them with the object clusters. We compute image derivatives both on the RGB and depth images in both the x and y directions using Scharr filters [26]. We threshold the resulting depth and color derivative images and combine them into a single binary edge image. We then remove isolated pixels thin blobs and link neighboring edge pixels in the binary image to form edges. Finally we associate edges with the clusters they fall on, discarding those edges that do not lie on any of the object clusters or have too few 3D points associated with them. Extracted edges are shown in Figure 2a.

Each remaining edge corresponds to a specific boundary hypothesis. We generate the object hypothesis by splitting the object cluster point cloud by a vertical plane defined by the chosen edge's location and dominant orientation. We determine the edge orientation by fitting a 3D line to the set of edge points using RANSAC.

The vector, v, formed in the direction of the x-y component of the 3D line in the table plane defines the direction of the splitting plane. We compute the normal to the splitting plane, n, by taking the cross product of v and the vertical axis vector $(0, 0, 1)$. We then specify the splitting plane as the plane with surface normal, n passing through the point p returned from the 3D line fitting process.

Points from the object cluster are then labeled as belonging to hypothetical object A or B according to which side of the plane they fall on. We discard from consideration edges where A or B have few points, as edges generating such splits correspond to object boundaries on the outside of the object cluster neighboring free space. An example object

hypothesis is shown in Figure 3.

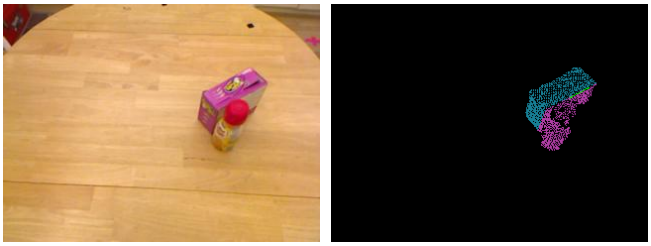


Fig. 3: Object hypothesis generated by the image edge highlighted in green.

The 2D orientation of each vertical splitting plane can be described in the internal reference frame of each object cluster with a single angle in the range $(-\frac{1}{2}\pi, \frac{1}{2}\pi]$ radians. We compute this angle for all edges associated with a given object cluster. We then construct a boundary estimate histogram for each object cluster by quantizing each edge orientation into one of n bins. Figure 2b illustrates these boundary orientation histograms for a scene with three object clusters.

B. Push Vector Selection

For any candidate boundary we generate a set of four possible abstract push vectors. We produce push vectors for all candidate boundaries associated with the object cluster of interest. The robot then ranks this set of push vectors and performs the highest ranked push. For a given boundary a push vector points parallel to the splitting plane running through the centroid of either hypothetical object A or B. Two potential push vectors run through each hypothetical object centroid giving us four candidate push vectors. We determine the start location and length of each push vector by locating the intersection of the line through the centroid with the hull of the hypothetical object point cloud. We present one such abstract push vector in Figure 4.

We rank the resulting set of abstract push vectors by preferring those that have start and end locations that do not leave the tabletop or robot workspace. After this, we rank higher those pushes that are less likely to collide with other object clusters. The remaining push options are ranked in decreasing order by the ratio, R_{AB}, between the number of points in the hypothetical objects:

$$R_{AB} = \frac{\min(|A|, |B|)}{\max(|A|, |B|)}$$

Thus objects with near equal splits score close to 1 and splits that generate one object much larger than the other receive scores closer to 0. We prefer even splitting sizes for three main reasons. First objects in a given environment tend to be of relatively equal size and thus tend to generate an equal number of points in the point cloud split. Second smaller object sizes have less surface area and are thus more difficult to make contact with while pushing. Lastly, even if the generated split does not align with the actual object boundary pushing farther from the middle of the cluster, in the correct direction of an object boundary, will more likely push only one of the two objects present.

Fig. 4: Example push vector generated for a single object cluster containing five objects.

C. Outcome Explanation and Evidence Accumulation

Once a push action is performed the robot first segments the table and clusters the remaining points as described in Section IV-A. It can then determine which clusters have moved by taking the difference of the point clouds before and after the push action. We estimate the motion underwent by each moved cluster using iterative closest point (ICP) [27]. ICP generates a rigid body transformation for each cluster as well as a score describing the goodness of fit between the two input point clouds. We maintain object cluster identification by matching each moved object cluster after a push with the best fitting cluster from before the push, while maintaining uniqueness in correspondence. We compose the most recent transformation estimate with the previous estimates of the cluster's motion, so that its rotation since initialization can be recovered. The recovered transformation is then used to rotate the cluster's internal frame of reference in order to keep the push history aligned over time, as well as correctly populate the edge orientation histogram for the image observed after a push.

Before updating the clusters push history we must analyze the resultant scene. If the number of clusters remains the same, that is no clusters merged or split, we can possibly increment the push history. We first check that the intended object was in fact moved. If this is the case we only update the push history if the fitness score returned by ICP is below a pre-determined threshold. We avoid updating the history for bad ICP fits as it likely means the push began to disrupt the configuration of multiple objects within a cluster, but was unsuccessful in separating them. Thus another attempt may well break the objects apart. For the case where clusters split, confirming an object boundary hypothesis, we discard the previously split object cluster and initialize two or more new object clusters associated with the split. For each new object cluster we set the push histories to be empty and define their tracked transformations to be the identity matrix. In rare cases clusters will unfortunately merge do to unaccounted for pushing dynamics or accidental collisions of the robot arm with the scene. We take a conservative view of these cases and reinitialize the merged cluster with no push history or tracked rotations.

We acknowledge that this tracking method will produce drift in the object orientation over time. While more sophis-

(a) GRIPPER_PUSH Before (b) GRIPPER_PUSH After (c) ARM_SWEEP Before (d) ARM_SWEEP After

(e) OVERHEAD_PUSH Before (f) OVERHEAD_PUSH After (g) OVERHEAD_PULL Before (h) OVERHEAD_PULL After

Fig. 6: The four implemented push behaviors used by the robot.

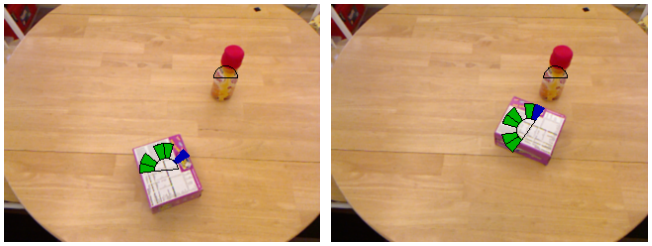


Fig. 5: Example push history before and after pushing a cluster corresponding to a single object. The blue bin corresponds to the direction of the next selected push.

ticated methods could be used to mitigate this issue, we find the precision of the rotational estimate is good enough for our purposes, since we are operating at the relatively coarse scale of quantized boundary and push orientations.

D. Push Behavior Selection

The robot translates a given abstract push vector into a real-world action using one of four different pushing behaviors. This set of behaviors allows the robot to more effectively accomplish the intended separation of hypothetical objects, when possible, while accounting for constraints in the environment and its own joint limitations. Each behavior takes as input the abstract push vector defined as a start location (x, y) and push angle θ in the world frame as well as the distance d_p to push.

The GRIPPER_PUSH behavior, pictured in Figure 6a-6b, places the tip of the end-effector at the start pose (x, y, θ) keeping the top of the robot hand parallel to the table. GRIPPER_PUSH then moves the end effector the distance d_p along the straight line defined from the gripper wrist to its tip. ARM_SWEEP(Figure 6c-6d) uses the broad face of the hand to push objects while remaining close to the table surface. The center of the hand is positioned at the location (x, y) on the table with vector normal to the hand facing in the direction of θ. The arm is then controlled so that the hand moves in a straight line in the direction of the palm for the distance d_p. The third behavior, OVERHEAD_PUSH (Figure 6e-6f), where the robot bends its wrist downward and pushes in the direction of the back of the hand. Again we control the center of the hand to move in the direction normal to its face after positioning it at the (x, y, θ) pose.

These three behaviors all move to the start pose from a predefined home position following a straight-line trajectory in the work space. The fourth behavior, OVERHEAD_PULL (Figure 6g-6h) performs the same action as OVERHEAD_PUSH, except that it will pull towards the robot and navigates to its start pose by first moving to the location (x, y, Z_h), where Z_h is high above the table to avoid colliding with the intended object to pull.

Pushing behaviors are selected based on the input pose (x, y, θ). The OVERHEAD_PUSH behavior is used for initial locations close to the robot, where there is little clearance for the arm to fit between the robot torso and body. For lateral pushing angles the ARM_SWEEP behavior is chosen, while OVERHEAD_PULL is used for angles point towards the robot torso. GRIPPER_PUSH is used in the remaining cases. The decision between using the left or right arm is based on workspace limits of the arms. If either arm is capable of making the push, the arm is chosen that pushes away from its side in an attempt to keep the workspace less crowded.

V. Experimental Results

We report comprehensive qualitative demonstrations of our method as well as quantitative comparisons to other push selection methods. All of our experiments were performed by a Willow Garage PR2 robot with a Microsoft Kinect camera mounted on the head in the Georgia Tech Aware Home.

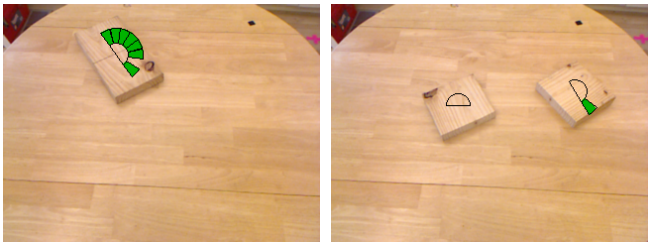
A. Qualitative Results

In order to validate our proposed claim and understand its limitations, we first present results on a set of qualitatively different groups and arrangements of objects.

(a) (b)

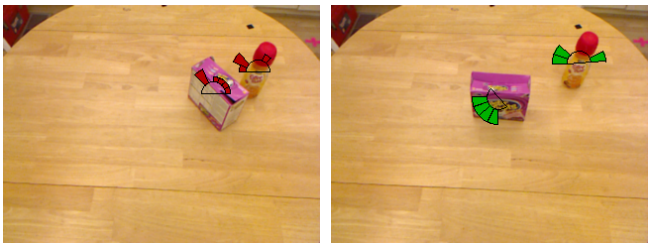
Fig. 7: Distribution of detected potential splitting boundaries. (a) A single wood block with a line drawn on it. (b) Two wood blocks placed together.

We begin by examining two visually similar situations. Figure 7 shows the initial view of two different scenarios. In one a single block of wood has a line drawn down its center in the other two blocks of wood are pressed together appearing as a single block. We note that the two different scenes generate quite similar images, which would be hard to discriminate with standard object recognition methods. Additionally, large color and texture overlap exists between the objects and table, making unsupervised segmentation based on low-level visual cues challenging. Beyond this the large flat surfaces would be difficult for most robots to grasp.



(a) (b)

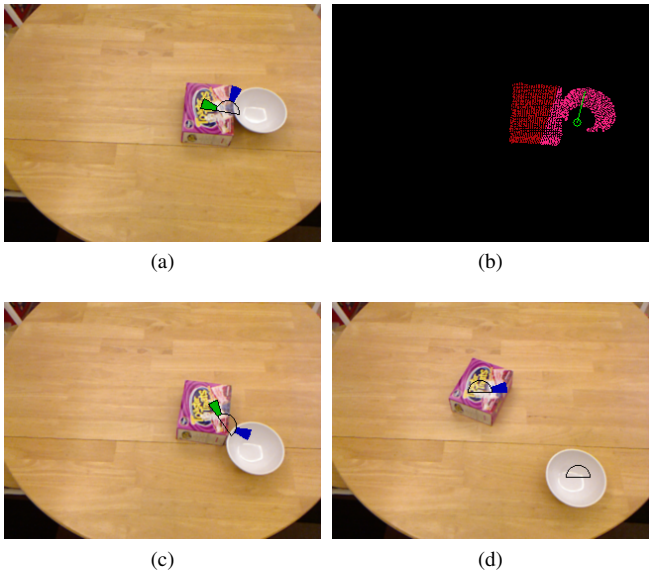
Fig. 8: Push histories for object clusters. (a) A single wood block with a line drawn on it after being determined to be a single object. (b) Two wood blocks correctly singulated after starting placed together.



(a) (b)

Fig. 9: (a) Distribution of candidate boundary orientations after one push. (b) Push history accumulated after an additional 13 pushes.

However our method successfully determines that Figure 7a corresponds to a single object, while Figure 7b shows two separate entities. Figure 8 shows the end result and push history of our singulation approach. The robot performed a total of seven pushes on the solid object, as a number of edges were detected on regions of different color in the wood pattern.



(a) (b) (c) (d)

Fig. 10: Green shows populated push history directions. Blue represents the direction of the chosen push. (a) After pushing perpendicular to the object boundary a push is chosen (b) that should separate the objects. (c) The objects failed to separate. However, ICP returns a bad score, so the push history is not updated. (d) The next push successfully separates the objects.

A less contrived example is shown in Figure 1. The scene contains five objects in collision that the robot separates with four pushes. Figure 4 shows the first push chosen by the robot to singulate the objects. While the chosen edge to test corresponds to a true object boundary, multiple objects, the bowl and green mug, lie on the other side of the plane defined through this edge. Thus, while the push separates the bowl from the textured coffee cup, other objects remain in contact and must be separated through subsequent pushes. Figure 2 depicts the scene after the initial push. While our method efficiently singulates the objects in this scenario, we note that the majority of objects have low surface texture and as such have few potential object boundaries. We now show results for high texture examples.

We show an example of textured objects in Figure 9, where the objects are separated on the first push. However, the robot must perform an additional number of pushes in order to accumulate evidence that all remaining boundaries are internal intensity edges and not object boundaries.

Figure 10 shows the importance of only updating the push history after successful ICP alignment. The attempted push in the direction of an actual object boundary does not push far enough to separate the two objects. However the system recognizes this and successfully separates the objects with the next push.

B. Quantitative Comparison

To test the robustness and efficiency of our method we compare directly to a push selection method, which does not reason about potential objects within object clusters. We compare to an unguided approach, similar to that of Chang et al., that pushes in a random direction through the centroid of a randomly chosen cluster. Pushes are discarded that are expected to push into other objects or out of the robot's workspace. We examine two termination criteria for this method. In the first we push each cluster a fixed number of times. In the analysis below, we term this method "fixed." Below we select the fixed number of pushes to be three per clusters. The second termination method is an attempt to have success with fewer pushes. We halt when the current set of clusters have all most recently been pushed without receiving a bad ICP fitness. We call this second method "rand-ICP." This second method is similar to that of Chang et al., as we rely on confirmation of a rigid body transform to assert object identity. However, rand-ICP is a simplification of their approach, since no image feature correspondences are used to seed the transform estimate and we do not compute an explicit likelihood calculation for classification of the cluster as a single object.

We experiment on a number of varying object sets and configurations. For each set of objects we randomly generate five configurations for each of the methods. We construct random configurations by generating a random 2D pose in the robot's workspace for each object and then push the objects together so that they are in contact with one another. We classify our different tests by the number of objects present and whether the objects have low or high amounts of texture. For each set of objects we report the number of successful singulation trials. We also show median, mean, and minimum number of pushes executed in all successful trials. Additionally, we note failures due to our segmentation system, where the push selection system is itself not at fault. For all results using our guided approach we report the number of histogram bins used.

Method	Success Rate	Min # Pushes	Median	Mean
Guided (4)	5/5	0	1	1
Guided (8)	4/5	0	1	1.7
Rand-ICP	4/5	2	2	2.5
Fixed	4/5	9	10	11

TABLE I: Results for sets of two objects with low texture. Number in parenthesis is the number of histogram bins used.

Method	Success Rate	Min # Pushes	Median	Mean
Guided (4)	5/5	1	4	3
Guided (8)	4/5	1	2	3.5
Rand-ICP	4/5	4	4	4.75
Fixed	3/5	11	15	14.0

TABLE II: Results for sets of three objects with low texture. Number in parenthesis is the number of histogram bins used.

TABLE I and TABLE II show results for tests where all objects present had relatively low surface texture. We note that the guided pushing achieved the best singulation success rate when using four histogram bins. The guided approach with eight histogram bins performed as well as rand-ICP, but needed fewer pushes. Fixed performed worse in all respects. The one failure of our method in the two object case (TABLE I) was a result of the segmentation claiming that the two separated objects were actually three. Rand-ICP failed because it incorrectly asserted that the two objects that moved together after being pushed were one. The minimum scores of zero pushes resulted from the segmentation system correctly separating the objects even though they were in contact. Our method was able to recognize that no pushes were necessary, since no edges produced candidate boundaries. We show the estimated boundaries for this case in Figure 11. We note that the random methods would still have to perform at least two pushes in such situations, since they have no estimate of potential objects making up a given object cluster prior to pushing.

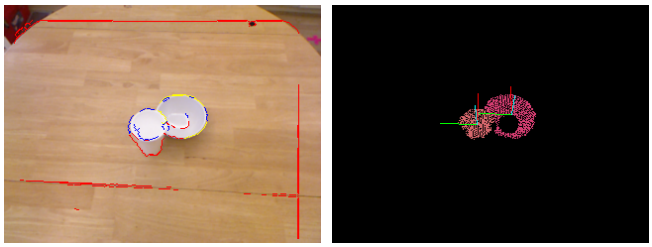


Fig. 11: Boundary estimates and segmentation relating to a correct segmentation of objects in contact. Axes represent the internal object frame axes of the two object clusters.

We report results for a set of two objects with high surface texture in TABLE III. Guided pushing using four histogram bins had the highest success rate, while guided with eight histogram bins produced success rates equal to rand-ICP method. However the high level of texture required more pushes to be made in order to singulate the objects. We note that all failure cases of our method and all failures of the fixed method were the result of objects being knocked over and falling off the table. One of the two failure cases for rand-ICP was due to the bottle present being knocked over and pushed off the table, but the other was a result of terminating prior to all objects being separated. Thus, while building evidence can increase confidence in the end result, it does so at the cost of performing more pushes that increases the chance of unintentionally knocking rollable items out of the scene.

Method	Success Rate	Min # Pushes	Median	Mean
Guided (4)	4/5	10	10	11.5
Guided (8)	3/5	14	14	15.3
Rand-ICP	3/5	2	3	2.7
Fixed	1/5	6	6	6

TABLE III: Results for sets of two objects with high texture. Number in parenthesis is the number of histogram bins used.

We report results for larger sets of five and six objects in TABLE IV and TABLE V. Following the success of using only four histogram bins on earlier experiments we do not

compare to the guided case of eight histogram bins. For both cases our approach performs the best. In the first, where five low textured objects are used, we have the highest success rate. The fixed method was unsuccessful in all trials, while rand-ICP consistently believed to have segmented all objects and halted early in all but one trial.

Method	Success Rate	Min # Pushes	Median	Mean
Guided (4)	3/5	6	24	18.3
Rand-ICP	1/5	7	7	7
Fixed	0/5	-	-	-

TABLE IV: Results for sets of five objects with low texture. Number in parenthesis is the number of histogram bins used.

Method	Success Rate	Min # Pushes	Median	Mean
Guided (4)	1/5	12	12	12
Rand-ICP	0/5	-	-	-
Fixed	1/5	30	30	30

TABLE V: Results for sets of six objects with varying levels of texture. Number in parenthesis is the number of histogram bins used.

For the challenging case of six objects with varying levels of texture we attain an equal success rate to fixed. However, our guided approach achieves the same performance level with less than half the number of pushes of fixed. Again rand-ICP regularly halted early, believing the scene to have as few as one object in it and never more than four. Fixed had similar issues, believing only three or four objects to be on the table. Our method failed once from believing only five objects to be present. All other failures of our method were do to items being knocked out of view of the robot during testing.

VI. Conclusions and Future Work

We have presented a novel approach to object singulation that explicitly hypothesizes about the orientation and location of potential separation boundaries between possible objects within a single object cluster. This reasoning allows us to state that singulation has occurred more confidently than methods which rely on random robot actions to serendipitously separate objects from one another. We have confirmed through experimentation that our method achieves singulation more reliably than unguided approaches and often does so with fewer pushes. Additionally our method works in situations where all objects present have low texture, a challenge not addressed by previously developed methods.

An obvious extension to our approach would be the use of template matching of singulated objects to robustly deal with accidental collision of other objects with those already singulated, similar to the method used by Kenney et al. We currently only test vertical boundaries as vertical splitting planes. By fitting planes parallel to the table plane and choosing appropriate pushing actions, our method could potentially separate stacked objects. Additionally, we currently choose pushes based on a single edge hypothesis. Reasoning jointly about all edge hypothesis could more quickly accumulate evidence towards singulation. More broadly, we are interested in having the robot autonomously learn which behavior to select and to perform real-time analysis and feedback control while pushing to more quickly singulate the scene and deal with more challenging configurations.

VII. Acknowledgments

This work was supported in part by NSF Award 0916687.

References

[1] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models." *PAMI*, vol. 32, no. 9, September 2010.

[2] B. Alexe, T. Deselaers, and V. Ferrari, "What is an object?" in *CVPR*, 2010.

[3] A. Saxena, J. Driemeyer, and A. Y. Ng., "Robotic grasping of novel objects using vision," *IJRR*, vol. 27, no. 2, pp. 157–173, Feb 2008.

[4] J. Sun, J. L. Moore, A. Bobick, and J. M. Rehg, "Learning Visual Object Categories for Robot Affordance Prediction," *IJRR*, vol. 29, no. 2-3, pp. 174–197, 2010.

[5] T. Hermans, J. M. Rehg, and A. Bobick, "Affordance prediction via learned object attributes," in *ICRA Workshop on Semantic Perception, Mapping, and Exploration*, 2011.

[6] W. H. Li and L. Kleeman, "Interactive learning of visually symmetric objects," in *IROS*, 2009.

[7] M. T. Mason, S. Srinivasa, and A. S. Vazquez, "Generality and simple hands," in *ISRR*, July 2009.

[8] E. Klingbeil, D. Drao, B. Carpenter, V. Ganapathi, O. Khatib, and A. Y. Ng, "Grasping with application to an autonomous checkout robot," in *ICRA*, 2011.

[9] M. T. Mason, "Mechanics and planning of manipulator pushing operations," *IJRR*, vol. 5, pp. 53–71, September 1986.

[10] K. M. Lynch and M. T. Mason, "Controllability of pushing," in *ICRA*, 1995, pp. 112–119.

[11] ——, "Stable pushing: Mechanics, controllability, and planning," in *IJRR*, 1996, pp. 533–555.

[12] D. Omrcen, C. Böge, T. Asfour, A. Ude, and R. Dillmann, "Autonomous acquisition of pushing actions to support object grasping with a humanoid robot," in *Humanoids*, Paris, France, 2009.

[13] M. Dogar and S. Srinivasa, "Push-grasping with dexterous hands: Mechanics and a method," in *IROS*, 2010.

[14] ——, "A framework for push-grasping in clutter," in *RSS*, 2011.

[15] A. Cosgun, T. Hermans, V. Emeli, and M. Stilman, "Push planning for object placement on cluttered table surfaces," in *IROS*, 2011.

[16] P. M. Fitzpatrick and G. Metta, "Towards manipulation-driven vision," in *IROS*, 2002, pp. 43–48.

[17] D. Katz, Y. Pyuro, and O. Brock, "Learning to manipulate articulated objects in unstructured environments using a grounded relational representation," in *RSS*, Zurich, Switzerland, June 2008, pp. 254–261.

[18] W. H. Li and L. Kleeman, "Autonomous segmentation of near-symmetric objects through vision and robotic nudging," in *IROS*, 2008, pp. 3604–3609.

[19] J. Kenney, T. Buckley, and O. Brock, "Interactive segmentation for manipulation in unstructured environments," in *ICRA*, 2009, pp. 1377–1382.

[20] L. Y. Chang, J. R. Smith, and D. Fox, "Interactive Singulation of Objects from a Pile," in *ICRA*, 2012.

[21] P. M. Fitzpatrick and G. Metta, "First contact: an active vision approach to segmentation," in *IROS*, 2003.

[22] D. Katz and O. Brock, "Extracting planar kinematic models using interactive perception," in *In Unifying Perspectives In Computational and Robot Vision*, ser. Lecture Notes in Electrical Engineering, vol. 8. Springer Verlag, May 2008, pp. 11–23.

[23] D. Katz, A. Orthey, and O. Brock, "Interactive Perception of Articulated Objects," in *ISER*, 2010.

[24] D. R. Martin, C. C. Folkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color and texture cues," *PAMI*, vol. 26, no. 5, pp. 530–549, May 2004.

[25] 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, June 1981.

[26] H. Scharr, "Optimal Second Order Derivative Filter Families for Transparent Motion Estimation," in *EUSIPCO*, September 2007.

[27] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," *PAMI*, vol. 14, pp. 239–256, 1992.