Robotics: Science and Systems 2013
Berlin, Germany, June 24-28, 2013

Perceiving, Learning, and Exploiting Object Affordances for Autonomous Pile Manipulation

Dov Katz, Arun Venkatraman, Moslem Kazemi, J. Andrew Bagnell and Anthony Stentz
The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA
{dkatz, arunvenk, moslemk, dbagnell, tony}@andrew.cmu.edu

***Abstract*—Autonomous manipulation in unstructured environments presents roboticists with three fundamental challenges: object segmentation, action selection, and motion generation. These challenges become more pronounced when unknown man-made or natural objects are cluttered together in a pile. We present an end-to-end approach to the problem of manipulating unknown objects in a pile, with the objective of removing all objects from the pile and placing them into a bin. Our robot perceives the environment with an RGB-D sensor, segments the pile into objects using non-parametric surface models, computes the affordances of each object, and selects the best affordance and its associated action to execute. Then, our robot instantiates the proper compliant motion primitive to safely execute the desired action. For efficient and reliable action selection, we developed a framework for supervised learning of manipulation expertise. We conducted dozens of trials and report on several hours of experiments involving more than 1500 interactions. The results show that our learning-based approach for pile manipulation outperforms a common sense heuristic as well as a random strategy, and is on par with human action selection.**

Figure 1: Perceiving and manipulating unknown objects in a pile: Each detected object has a set of affordances (pushing, pulling or grasping). The robot selects the best next interaction for clearing the pile of unknown objects. The orange boundaries mark reachable space. The robot cannot grasp an object behind the white boundary, but may push or pull on it.

I. Introduction

Simple everyday tasks such as clearing a pile of toys in the living room, tidying up a messy dinning table, and sorting a box of unused items in the garage remain challenging for robots. These pick-and-place tasks of unknown objects in a pile require careful integration between perception, planning, and motion generation. Furthermore, the robot must move with care to avoid damage to itself and the environment, perform the task quickly, and make as few assumptions as possible.

There are several key prerequisites for manipulating a pile of unknown objects. First, the robot must acquire pertinent knowledge for interacting with individual objects in the pile. This is difficult because object segmentation remains an open problem, and is particularly challenging for a pile of overlapping and unknown objects. Because we cannot rely on prior object models, as the pile may contain natural objects, debris, and parts, the robot must hypothesize a segmentation of the environment into objects and compute for each object a set of affordances [1], [2]. We address object segmentation in a pile by extending prior work on segmenting unknown objects using geometric properties. A contribution of this work is the implementation of a GPU-accelerated version of the segmentation algorithm proposed in [3].

Second, the robot must be able to choose which one of the affordances to execute next. Uninformed action selection can lead to slow performance, or worse, may damage the robot or objects in the pile. An intuitive heuristic—a set of rules—may be helpful in determining the next action, but is likely to fail often as it is difficult to anticipate the behavior of objects in a pile and the outcome of interaction. We propose a learning approach to manipulation. Our object representation exposes the structure of the pile and the affordances of the individual objects. Using this representation within a supervised learning framework, our robot is able to learn the necessary manipulation expertise to efficiently and reliably clear a pile of unknown objects (see Figure 1). This learning-based approach for pile manipulation is our most important contribution.

And third, the robot must generate motion plans that both avoid collision with other objects and carefully interact with the target object. Another contribution of this work is a library of novel compliant controllers for poking, pulling and grasping unknown objects. These controllers are executed within a state-of-the-art motion planing pipeline.

We evaluated our solution by conducting extensive experiments. In our experiments, the robot interacts with a pile of unknown objects placed on a table. The robot's task is to pick up individual objects and place them in a bin. We used both man-made and natural objects of varying shape, size, and appearance. We conducted several hours of experiments consisting of over 1500 interactions. Our results demonstrate that perceiving object affordances and learning to rank these affor-

dances to determine the best next action facilitates a robust, efficient, and reliable pile clearing behavior. For transparency, we have uploaded many unedited videos of our experiments to: http://www.youtube.com/user/pileRSS2013/videos.

II. Related Work

Our approach for manipulating unknown objects in a pile has three main components: perception (to segment objects and compute relevant features), action selection (to determine object affordances, calculate the corresponding manipulation actions, and choose the best next action), and motion generation (to instantiate and execute the appropriate compliant controllers). We now review the most relevant works in each area.

A. Object Segmentation

To determine the affordances of objects in the pile, we must first segment individual objects. Segmentation algorithms [4], [5] process an image and divide it into spatially contiguous regions sharing a particular property. These algorithms assume that boundaries between objects correspond to discontinuities in color, texture, or brightness—and that these discontinuities do not occur anywhere else. These assumptions are easily violated in a pile because of the significant overlap between objects. Thus, existing methods become brittle and unreliable.

Segmentation from motion algorithms leverage a different cue for segmentation: relative motion. This motion is either assumed to occur [6], [7], [8] or can be induced by the robot [9], [10]. Although relative motion is a strong cue for segmentation, generating this motion in an unknown pile is oftentimes dangerous and undesirable. Our proposed method does not generate motion for the purpose of segmentation, but does utilize relative motion when it occurs.

Segmentation can also be computed by considering 3-D geometry to determine the boundaries between objects [11], [12]. Here, a boundary is defined as a depth discontinuity, and objects are modeled using parametric representations of predetermined shapes such as spheres, cylinders, and planes. These methods assume that objects can be described using a single basic shape. In practice, this is rarely the case. Our method also relies on geometric segmentation. However, it uses a non-parametric approach (similar to [3]), and considers both depth discontinuities and continuity in surface normals orientation.

Without prior knowledge, every segmentation algorithm, including ours, becomes less reliable in clutter. Thus, for manipulation, any segmentation should be considered with caution. We complement our segmentation algorithm with learning, which enables the robot to identify unreliable segments, effectively increasing the reliability of segmentation.

B. Learning Manipulation Expertise

For every object segmented by perception, our method instantiates a controller (or several controllers) to safely interact with the object. These potential interactions represent the object affordances [2], [1]. Choosing which of the possible actions to take is important: an action may be more or less likely to succeed, safe or dangerous, free up space around an object or condense the pile. The sequence of actions determines the number of interactions necessary to clear the pile. Thus, choosing the next best action is crucial for efficiency. Our method uses supervised learning to score and rank the objects' affordances.

Learning manipulation expertise is challenging because of the large state space associated with perceiving and manipulating objects. It is virtually impossible to encounter the same state twice. Interesting examples in the literature that apply learning to manipulation tasks include using relational reinforcement learning to learn a policy for modeling articulated objects [13] or for manipulating basic objects such as cubes, cylinders and spheres [14]. Additionally, supervised learning has been used to find and rank multi-contact grasp locations on objects in partially cluttered scenes [15]. However, learning grasping among other manipulation skills in densely cluttered unstructured environments, such as in piles of objects, remains largely unsolved.

Recent work on pile manipulation [16], [17] relies on hard-coded heuristics for removing objects from a pile. In [16], flat objects are clustered together and the robot pokes objects until individual objects are singulated and can be picked up. And in [17], Lego blocks are singulated and removed. Here, the robot is provided with *a priori* knowledge of the object type (a Lego block). Our work extends [16], [17] by considering more complex objects (unknown, natural, and complex shapes), to consider more complex clutter (piles), and by introducing learning to guide the interaction.

C. Motion Generation

To execute a desired action, we first generate and execute a feasible trajectory to position the hand close to the target object. Then, we instantiate a compliant controller designed to achieve the desired manipulation behavior (pushing, pulling or grasping). We use CHOMP [18] to generate smooth trajectories and rely on a library of force feedback compliant motion primitives that are safe and appropriate for manipulation under uncertainty [19].

III. System Overview

Our proposed system for manipulating unknown objects in a pile has three main components (Figure 2): perception, learning-based action selection, and manipulation. Perception generates a set of object hypotheses ("facets"). Action selection considers the affordances of each object (using SVM classifiers) and chooses the best next action with the objective of clearing the pile safely and efficiently. And the manipulation pipeline computes a motion plan and executes the appropriate compliant controller.

IV. Perceiving Objects

Our perception pipeline is composed of two parts. The first computes a segmentation of the scene into facets (hypothesized

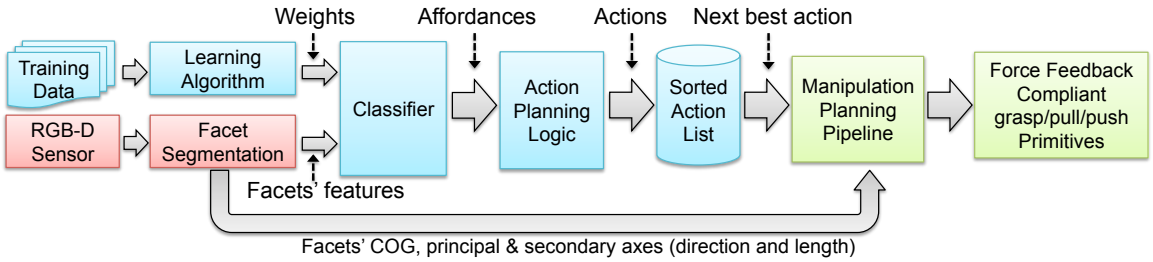


Figure 2: System overview: Perception (red) generates a segmentation of the scene into facets. Information about individual facets is used by learning (blue) to classify and score the affordances of each facet. Then, actions are ranked according to their scores, and the selected action is executed by instantiating a compliant controller (green).

object surfaces). For every facet, we extract the necessary information to instantiate our compliant controllers for pushing, pulling or grasping. The second part of the perception pipeline computes a set of visual features for each facet that is later used within a supervised learning framework to classify the affordances of each object.

A. *Facet Segmentation*

To interact with unknown objects in a pile, we must first identify individual objects. Using 3-D information measured with an RGB-D camera (Kinect), our algorithm segments the scene into hypothesized object facets. A facet is an approximately smooth circumscribed surface. An object facet is not necessarily a flat surface (plane), but rather a region maintaining continuity in both depth and the orientation of its surface normals. Dividing an object into facets is intuitive and repeatable under changes of perspective, lighting condition, and partial occlusion.

Facet detection is composed of the following three steps: computing depth discontinuities, estimating surface normals, and color-based image segmentation. This process is illustrated in Figure 3. We compute depth discontinuities by convolving the depth image with a non-linear filter. This filter computes the maximal depth change between every pixel and its immediate 8 neighbors. If this distance is larger than 2cm, the pixel is marked as a depth discontinuity. The 2cm threshold is due to the resolution of our RGB-D sensor (Kinect). The surface normal at every point of the 3-D point cloud is estimated by fitting a local plane to the neighborhood of the point. We then compute the normal to that plane using least-square plane fitting. Figure 3 provides a visualization of the surface normals. The three Euler angles of every normal are represented using the three color channels (RGB). Finally, we overlay the depth discontinuities onto the color representation of the surface normals (to form a color discontinuity where there is depth discontinuity). Now, extracting facets becomes a color segmentation problem of extracting contiguous color regions. Therefore, we extract facets using a standard color segmentation algorithm (mean-shift segmentation). More details and an experimental evaluation of facet detection is available in [3]. Our contribution compared to that in [3] is algorithmic. Our version is more efficient and uses GPU acceleration where possible. This leads to a x10 runtime speedup, which is essential for real-world manipulation.

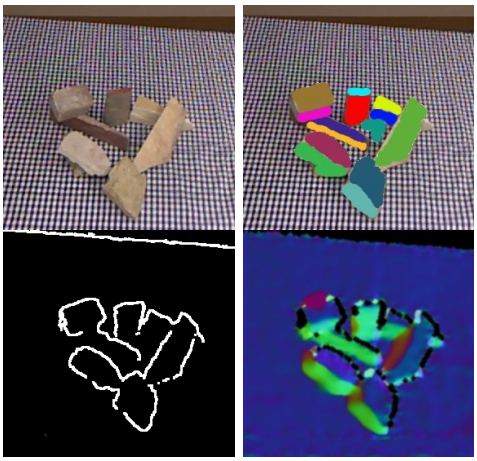


Figure 3: Facet detection algorithm: The input (top left) is an RGB-D image. The algorithm extracts geometric discontinuities: depth discontinuities (bottom left) and normal discontinuities (bottom right). Finally, we merge depth and normals into a single RGB image. Object facets (top right) are extracted by computing color segmentation on that image.

Every segmented facet represents a hypothesized region where the robot can interact with the pile. For every facet, we compute its center of gravity (COG), the principal and secondary axes, and the length of each axis. We compute the COG of a facet by averaging the 3-D positions of the associated point cloud. We determine the principal and secondary axes by performing principal components analysis (PCA) on the 3-D point cloud. The length of each axis is the largest distance between a pair of points on or very close to each axis. Figure 4 illustrates the output of this process. With this information we can instantiate any one of our 3 types of controllers for pushing, pulling or grasping. Our controllers are compliant and use force control to compensate for partial and noisy perception.

Facet detection has two main limitations. First, our sensor (Kinect) cannot perceive reflective materials. And second, our method is not able to distinguish between two objects that are touching each other and have similar surface normals. This could be solved by considering color, texture, and experience. Because the robot disturbs the pile throughout its interactions, this case does not persist, and therefore has limited impact on our performance.

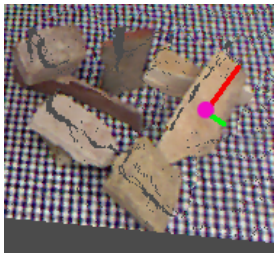


Figure 4: Extracting information for manipulation: our algorithm computes the COG (pink circle), principal axis (red) and secondary axis (green) for every facet. This information together with the length of each axis suffices to instantiate our compliant controllers for pushing, pulling or grasping.

B. Facet Affordances

With the list of segmented facets and the information necessary to instantiate any of our controllers for every facet, we must now decide what is the next best action. Not every action is desired. For example, grasping a facet may not be possible because of other objects around the facet, or pulling an object underneath other objects may fail and disturb the pile significantly, risking both the robot and the objects. Thus, we must determine what are the affordances of each facet, accounting for its surrounding and and create a ranking in order to determine what action to take next. These affordances depend not only on the facet itself, but also on its surroundings and the robot's capabilities.

Table I lists the 41 features we compute to determine the affordance of each facet. This list can be easily extended to include additional features. In the next section, we use these features within a supervised learning framework to determine the actual affordances of a facet: can it be pushed, pulled, and/or grasped along its principal or secondary axis.

#	Feature	Description
1	cloudSize	Number of 3-D points associated with the facet
2	facetArea	Projected 2-D area associated with the facet
3	distance	Facet's Euclidean distance from the robot
4	height	Facet's Euclidean distance from the support surface
5	length	Distance between the farthest points along the principal axis
6	width	Distance between the farthest points along the secondary axis
7	LW-ratio	Ratio between the length and width of the facet
8	surfaceAngle	Angle between the facet and the support surface. The facet is represented as the surface defined by the principal and secondary axes.
9	moveMatch	Robot's confidence in the facet segmentation. This is computed by considering two consecutive frames. If a facet was disturbed and it can be retrieved in the second frame, the robot's confidence in its segmentation increases. For more details about matching facets across view see [3].
10-41	freeSpace	Density of 3-D points around a facet determines the amount of free space around it. For efficiency, we only consider the area close to the extreme points of both the primary and secondary axes. Free space is represented by measuring the number of 3-D points in 8 small cylinders for each end of each axis. The cylinders are of radius 0.5cm, start at 2cm below the facet and end at 5cm above the facet. This feature is motivated by the notion that an empty or nearly empty cylinder indicates room for the fingers.

Table I: List of facet features associated with affordances. These features are used within our supervised learning framework to select the best next action.

V. Learning Object Affordances

We developed a supervised learning approach to manipulation for computing facet affordances. This is the most significant contribution of our work. Learning relies on the 41 features computed by perception (see Table I). For training data, we labeled 37 scenes containing a total of 550 facets. For each scene we used two image frames. We initially setup the scene (first frame), and in some cases disturbed the scene (second frame). Labeling was done for the second frame. The motion caused by disturbing the scene (if any) was used to compute feature #9 in Table I. We developed a graphical user interface for displaying the segmented facets; the user assigned to each facet 5 binary labels: actionable, push, pull, grasp-P and grasp-S (grasping along the principal or secondary axis). The labels are not mutually exclusive and do not represent a preferred action. Instead, they indicate whether a facet can be interacted with, pushed, pulled, or grasped along either axis.

To classify the affordances of a facet, we use simple linear support vector machines (SVMs). Each feature is normalized by its variance and thresholded outside of two standard deviations[1].

We trained each classifier with 450 randomly selected instances, and tested on the remaining 100. The resulting classification rates and the distribution of positive and negative labels in the training set are summarized in Table II. Grasping affordances are correctly classified in 80% of the cases and pushing and pulling are correctly classified in over 90% of the cases. We are also able to detect when a facet is invalid (not actionable) in 81% of the cases. This is important for recovering from segmentation errors. A more careful analysis of the results shows that most of our misclassifications ($\geq 90\%$) are true negatives, meaning that the learner is conservative in deciding to act, which results in safer behavior.

Given a new scene, the robot is now ready to compute a segmentation, determine facet affordances, and rank the actions according to the score computed for every $< facet, action >$ pair by the classifiers. In our experiments, we create an action list by first adding the top 3 grasping actions followed by

[1]We scale each feature f_i using its mean $E(f_i)$ and variance $V(f_i)$. $f_i^{scaled} = (f_i - E(f_i))/\sqrt{Var(f_i)}$. If a scaled feature is more than two standard deviations away from the mean, we cap f_i^{scaled} at either -2 or 2. Finally, we divide f_i^{scaled} by 2 to guarantee that all features are in the range $[-1, 1]$.

Class	%Positive	%Negative	%Classification Rate
Actionable	75	25	81.20
Push	43	57	91.75
Pull	59	41	93.45
Grasp-P	26	74	80.34
Grasp-S	37	63	80.10

Table II: Classifying facet affordances: we compare the distribution of positive and negative instances in the training examples to the classification rate achieved after training. The results show significant improvement of 24.8%, 80.8%, 84.0%, 24.4%, and 46.2% in the misclassification rate respectively for each of the aforementioned classes compared to the naive approach of selecting the most probable label for each class.

the top 3 pushing or pulling actions. Finally, we add all remaining actions (sorted by score). When an action cannot be performed (either because the planner detects a possible collision or because a trajectory to the goal configuration is infeasible), we continue to the next action in the list. In future work, we intend to replace this action planning logic with reinforcement learning. This will enable us to develop simple strategies and learn from experience the appropriate scaling between the scores of the different classes.

VI. Compliant Motion Primitives

To interact with the environment, we propose three types of parameterized controllers: pushing, pulling and grasping. Each controller is instantiated by perception based on the computed COG, principal and secondary axes, and the length of each axis. These controllers are inspired by and extend on the compliant grasping primitives developed in [19].

Interacting with unknown objects in a pile is challenging because the robot has only partial and inaccurate knowledge of the shape and configuration of objects. Thus, our controllers must be robust to uncertainty in modeling and localization. Our system uses CHOMP [20] to plan a collision free trajectory to an action launch pose (i.e., robot hand pose) based on the COG and orientation of the facet. Then we execute a compliant controller which maintains proper contact with the environment by responding to the detected contact forces. Our compliant controllers support pushing, pulling and grasping (either along the principal axis or the secondary axis). They are velocity-based operational space controllers, relying on force feedback acquired by a force-torque sensor mounted on the robot's wrist. During the interaction, the robot's fingers are coordinated and position-controlled.

To grasp an object, we servo the hand along the palm's normal, until contact is detected between the fingertips and the support surface or the object. Then, we close the fingers, while the hand is simultaneously servo controlled in compliance with the forces measured at the wrist. This ensures safe and proper contact between the fingertips and the support surface. Figure 5 illustrates this process for grasping a block. Note that the palm is aligned with the facet and centered above the facet's COG. Also, the hand's aperture is determined by the length of the facet along the relevant axis.

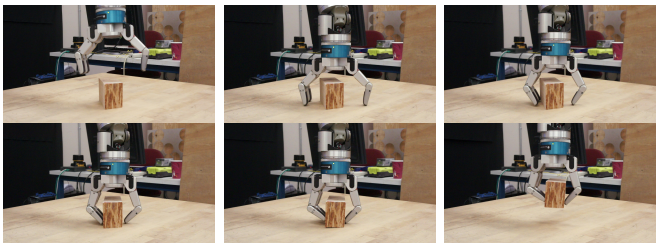


Figure 5: The steps of compliant grasping: the Barrett hand assumes a cup-like pre-shape on top of the facet's center of gravity and is parallel to an axis of the facet. It moves towards the object until contact is detected. The fingers close onto the object while the hand is servo controlled in compliance with the forces due to contact with the support surface/object.

Pushing and pulling begin in a similar way: we servo the hand along the palm's normal until contact is detected. To push an object, we continue moving along the normal until we either completed a trajectory of 5cm, or the forces exerted onto the hand or fingers exceed a safety threshold. To pull an object, we apply force along the normal (to maintain contact), while pulling the object. Again, the action ends after moving for 10cm or if an unsafe amount of force is detected. We have thoroughly tested the implementation of the three compliant controllers on a 7-DOF Barrett Whole Arm Manipulator (WAM) and a 3-fingered Barrett hand.

VII. Experimental Evaluation

To evaluate our system, we conducted dozens of experiments with a robotic manipulation system [20]. Videos of all of the experiments conducted for this paper are available at http://www.youtube.com/user/pileRSS2013/videos. In our experiments, a variety of unknown man-made and natural objects were placed in a pile on a table in front of the robot (e.g., Figure 1). The objects overlap and occlude each other to varying degrees. The robot is composed of a 7-DOF Barrett WAM and a 4-DOF hand equipped with force/torque sensor at the wrist. It acquires RGB-D measurements of the environment using a Kinect. The robot is tasked with clearing the table by removing all objects into a bin.

We conducted three types of experiments. First, we evaluated the performance of 5 methods for selecting the next action: our learning-based approach, 2 random action selection strategies, a common-sense heuristic, and human-operator selected actions. Second, we analyze interesting instances highlighting the benefits of our learning-based approach. And finally, we compare the affordance classification of our learning method to action selection by human subjects.

A. Clearing piles of unknown objects

The main contribution of this work is developing a learning-based approach to manipulation. Our learned classifiers rank the affordances of segmented facets and generate a sorted list of actions. We compare the performance of learning to three other methods for action selection: random, heuristic-based selection, and a human operator. For random, we consider two strategies: select a facet at random and then either select one

of our four action at random (all-random) or select only one of the two grasping actions at random (grasping-only-random). Our heuristic-based approach uses the following intuition:

1) Grasping the topmost object is safer and more likely to succeed
2) Grasping along the secondary (shorter) axis increases the chance of the object fitting into the robot's hand.
3) If an object is out of reach for grasping (behind the white line in Figure 1), pulling is required.
4) Pushing can disturb/reorganize the pile and is therefore useful if the above actions cannot be performed.

We call this a common-sense heuristic as it encodes simple and seemingly obvious rules. It is possible to hard-code a more complicated heuristic utilizing all of the 41 features from Table I; however, this can be difficult, time-consuming and brittle, in part due to errors such as noise and calibration offsets.

For the 'Human' experiments, the human operator selects the next action for the robot to execute using a graphical user interface to click on a facet and choose an action.

Table III and Figures 6 and 7 summarize the results of our experiments. We conducted extensive experiments consisting of 10 trials using each of our 5 methods for action selection. In our experiments, the robot attempted over 1500 actions. In all experiments we used a randomly shuffled pile of the same 10 objects. When using all-random, the robot was never able to clear the pile. For example, the robot was able to remove only 2 objects after 50 actions. In Table III we present the results for the other 4 selection methods. We count the number of actions in every trial. A successful action occurs when the robot is able to plan a trajectory, executes it, and achieves the manipulation objective. A failed action occurs when the planned trajectory cannot be executed because of collision, the goal configuration cannot be reached by the robot, or the action itself fails (e.g. object slips out of hand). For each trial, we report the percentage of failed actions due to planning (%PF) and failure to achieve the manipulation goal (%EF).

Figure 6 shows the average number of actions and a standard deviation for each action selection strategy. The performance achieved by the human operator is not significantly different than our learning-based approach. Using our heuristic, the average number of actions is about 50% higher than learning, and it increases by another 20% when randomly selecting a grasping action. These results show the strength of our learning-based approach.

Figure 7 shows for each action selection strategy, the percentage of successful grasps out of the total attempted grasps. As expected, when a human selects a facet with a grasp action, the probability of success is the highest. Learning performs about 10% worse. The likelihood of executing a successful grasp drops dramatically for the heuristic-based approach as well as for random. We believe that the results indicate that the human prefers preparatory actions (push/pull) to singulate objects over attempting difficult grasps. While this results in a higher grasping success rate, it also leads to more actions. Learning is more adventurous in choosing grasps. Although this results in more frequent failures to grasp, this strategy pays off as the overall number of actions needed is similar to what a person requires.

	Actions					Failures		
Pile	**#**	**%GP**	**%GS**	**%PU**	**%PL**	**#**	**%EF**	**%PF**
Random (grasping only)								
1	12	50	50	-	-	4	50	50
2	31	26	74	-	-	22	36	64
3	44	48	52	-	-	35	54	46
4	32	44	59	-	-	23	65	35
5	35	60	40	-	-	24	42	58
6	55	52	47	-	-	44	77	23
7	32	57	43	-	-	28	67	33
8	23	52	47	-	-	15	46	54
9	47	51	49	-	-	40	60	40
10	43	48	51	-	-	36	83	17
Heuristic								
1	35	-	94	3	3	25	100	0
2	10	-	100	0	0	0	0	0
3	22	-	90	5	5	10	80	20
4	63	-	92	5	3	51	16	84
5	33	-	100	0	0	23	43	57
6	14	-	100	0	0	4	100	0
7	65	-	87	5	8	52	13	87
8	19	-	95	0	5	12	34	66
9	17	-	88	0	12	8	37	63
10	23	-	83	4	13	10	50	50
Learning								
1	15	20	66	7	7	6	17	83
2	15	7	87	0	6	4	75	25
3	12	25	67	0	11	1	100	0
4	33	12	79	6	3	19	15	85
5	28	11	78	11	0	16	18	82
6	13	8	85	7	0	3	67	33
7	14	28	50	15	7	2	100	0
8	36	14	78	3	5	24	84	16
9	8	12	88	0	0	0	0	0
10	17	18	70	0	12	9	45	55
Human								
1	16	25	50	0	25	3	67	33
2	12	0	83	0	17	1	100	0
3	13	77	0	8	3	3	67	33
4	26	27	46	15	12	13	77	23
5	22	23	32	18	27	8	75	25
6	17	24	41	12	24	5	80	20
7	23	26	57	9	9	12	83	17
8	18	33	39	6	22	5	60	40
9	20	15	50	20	15	7	29	71
10	21	5	67	5	24	7	100	0

Table III: Results for 10 consecutive trials using the same 10 objects in arbitrary piles with our four action selection strategies: **Random-Grasping-only**, **Heuristic-based**, **Learning-based**, and a **Human Operator-based**. The columns are (left to right): trial id, number of actions to clear the pile, percentage of actions that were grasping-principal-axis (%GP), grasping-secondary-axis (%GS), pushing (%PU), and pulling actions (%PL), the total number of failures and the percentage of failures due to either execution (%EF) or planning (%PF).

B. Doing the right thing

Our second set of experiments analyzes interesting instances that demonstrate the behavior that was learned from the training data. In Figure 8 (left), we presented the robot with a single large object (the detected facet marked in red). Learning classified this facet as negative for the "actionable" category, and did not attempt to interact with it. The other approaches (heuristic and random) kept interacting with the object without success.

The middle image in Figure 8 contains three facets (red and green for the box that is out of the reachable area for

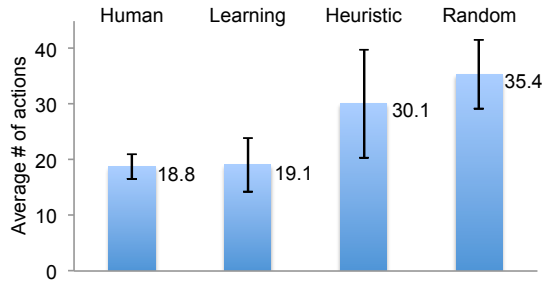


Figure 6: The average number of actions required to remove all objects from the pile of 10 objects for all 4 action selection strategies. The results show that learning and human-operator action selection have similar performance, and are significantly better than the simpler methods (random and heuristic-based).

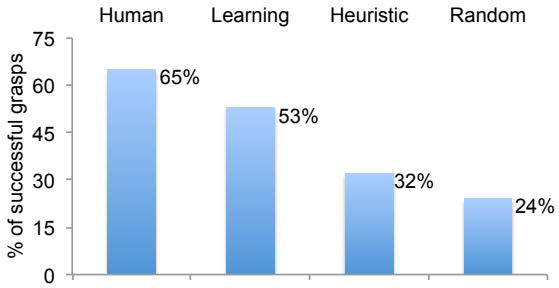


Figure 7: The average fraction of successful grasps out of the total number of attempted grasps. The human action selection is more conservative, leading to higher success rate. Learning attempts more difficult grasps, which leads to more failures. However, note that both strategies require a similar number of action on average (Figure 6). Heuristic-based and random action selection fail to execute a grasp in more than 60% and 70% of the cases respectively.

grasping and blue for the ball). The top three actions ranked by learning are: pushing the orange ball (blue facet) into the reachable area, pulling the green facet and grasping the red facet along the longer axis. The heuristic would try to grasp the ball (difficult configuration, likely to fail) or pull the green facet (good). The red facet cannot be grasped (planning failure because grasping along the short axis would result in collision with the table), and since it cannot be grasped but yet is not outside the graspable zone, the heuristic will not try to pull it closer. Instead, it will keep pushing it towards the non-graspable zone. The right image in Figure 8 shows cases where learning prefers pushing vs. pulling. As expected, learning classifies the green and red facets as positive for pushing and negative for pulling. The blue facet is classified as positive for pulling and negative for pushing.

In Figure 9 we observe a frequent failure mode of our heuristic-based approach. Since it always grasps along the shorter axis and does not consider whether there is free space along this axis, it would randomly choose to grasp either the red or blue facets. The result strongly depends on the structure of the scene (left: success, right: failure).

In Figure 10 we demonstrate that learning oftentimes generates sequences of interaction that benefit the robot. In this example, learning classifies both types of grasping as negative (the objects are too long for principal axis grasp and too close to each other to grasp along the shorter axis). Learning ranks pulling the red facet as the best next action, and after executing it (right image), grasping both facets becomes possible.

C. Action Selection: Human vs. Learning

Figure 11 visualizes the ranking computed by our learning-based approach. For each affordance, the detected facets are color coded according to the output of the classifier: positive (green) and negative (red). For each affordance, the best facet is marked in bright green and the worst in bright red.

Interestingly, we informally asked 10 people to classify the facets into the 4 type of affordances. Qualitatively, we found that the classification suggested by the human subjects was similar to that computed by our learning framework.

VIII. Conclusion

We developed a learning-based approach for manipulating piles of unknown objects. We provided extensive experimental data demonstrating the merits of our approach. With our learned classifiers, the robot interacts with the environment more efficiently than what was achieved with random interaction or by the common-sense heuristic. In comparison with a human-operator selecting the next best action, our learning-based approach achieves, on average, the same number of actions necessary to clear the pile of objects.

Learning and generalizing manipulation knowledge enables the robot to autonomously interact with dense clutter. Learning becomes possible due to our novel algorithm for segmenting an unknown scene into hypothesized object facets, allowing extraction of a rich set of features. Finally, our compliant controllers overcome inevitable inaccuracies in perception and maintain safe interactions with the environment.

An immediate extension to our supervised learning approach is to use on-line self-supervised learning to adjust the learned weights of the classifiers based on the actual outcome of the robot's actions. We believe that this approach is essential for enabling autonomous manipulation in unstructured environments.

Acknowledgments

This work was conducted in part through collaborative participation in the Robotics Consortium sponsored by the U.S Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement W911NF-10-2-0016 and also in part by Intel (Embedded Technology Intel Science and Technology Center). The authors also gratefully acknowledge funding under the DARPA Autonomous Robotic Manipulation Software Track (ARM-S) program.

References

[1] J. J. Gibson, *The theory of affordances*. Lawrence Erlbaum, 1977, vol. Perceiving, pp. 67–82.

[2] C. Barck-Holst, M. Ralph, F. Holmar, and D. Kragic, "Learning grasping affordance using probabilistic and ontological approaches," in *ICAR*, 2009, pp. 1–6.

[3] D. Katz, M. Kazemi, J. A. Bagnell, and A. Stentz, "Clearing a pile of unknown objects using interactive perception," in *ICRA*, Karlsruhe, Germany, May 2013, pp. 154–161.

[4] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

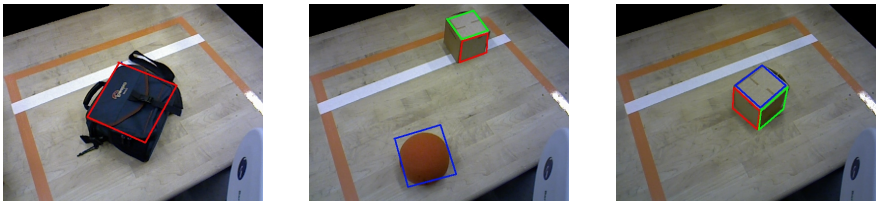


Figure 8: Doing the right thing: Analyzing the performance of our learning-based approach in interesting scenarios. **Left:** Learning recognizes the object is too big for grasping; it decides not to interact with it. **Middle:** Learning recognizes the red facet cannot be grasped (hand will collide with the table) and the green facet is outside the reachable zone grasping; it decides to pull the box (green facet) closer for grasping. The ball (blue facet) is too close for grasping; learning decides to push it towards the center. **Right:** Learning correctly classifies the blue facet as good for pulling but not for pushing. Conversely, learning recommends pushing the green and red facets and not to pull on them.

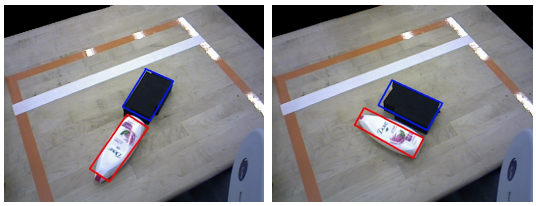


Figure 9: Because our heuristic-based approach always grasps along the shorter axis and does not consider collision, the success depends on the structure of the scene. It would work if there is no collision along the secondary axis (left) and fail otherwise (right). Our learning-based approach identifies the difference and can choose between primary axis and secondary axis as necessary.

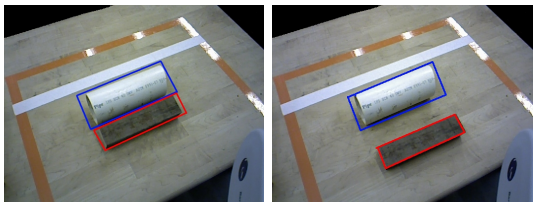
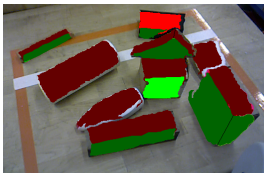
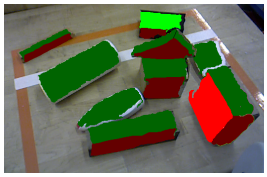


Figure 10: Here (left) grasping along both axes will fail because the primary axis is too long to fit in the hand and using the secondary axis will result in collision. Learning anticipates this failure and prefers to pull the red facet first. In the next two steps, learning will remove the red and blue facets that are now separated and easy to grasp.

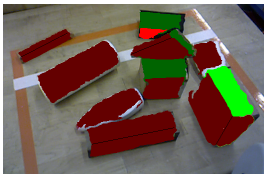
(a) Scene



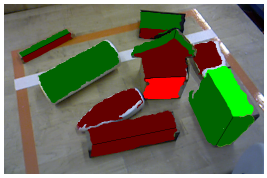
(b) Pushing



(c) Pulling



(d) Grasping - principal axis



(e) Grasping - secondary axis

Figure 11: An example scene composed of natural and man-made objects. We segmented the scene into facets and computed the classification assigned by learning for each affordance: positive (green) and negative (red). Bright green and bright red respectively represent the best and worst facet for each affordance. This classification was qualitatively similar to that suggested by 10 human subjects. Note that to simplify the task for the human subjects, this scene was constructed to be significantly simpler than those the robot was typically tasked with.

[5] L. Zappella, "Motion Segmentation from Feature Trajectories," Master's thesis, University of Girona, Girona, Spain, 2008.

[6] J. Zhang, F. Shi, J. Wang, and Y. Liu, "3D Motion Segmentation from Straight-Line Optical Flow," in *Multimedia Content Analysis and Mining*. Springer Berlin / Heidelberg, 2007, pp. 85–94.

[7] R. Stolkin, A. Greig, M. Hodgetts, and J. Gilby, "An EM/E-MRF Algorithm for Adaptive Model Based Tracking in Extremely Poor Visibility," *Image and Vision Computing*, vol. 26, no. 4, pp. 480–495, 2008.

[8] A. Goh and R. Vidal, "Segmenting Motions of Different Types by Unsupervised Manifold Clustering," in *CVPR*, June 2007, pp. 1–6.

[9] J. Kenney, T. Buckley, and O. Brock, "Interactive Segmentation for Manipulation in Unstructured Environments," in *ICRA*, 2009, pp. 1343–48.

[10] D. Katz and O. Brock, "Manipulating Articulated Objects with Interactive Perception," in *ICRA*, May 2008, pp. 272–277.

[11] C. J. Taylor and A. Cowley, "Segmentation and analysis of rgb-d data," in *RSS Workshop on RGB-D Cameras*, 2011.

[12] S.-W. Yang, C.-C. Wang, and C.-H. Chang, "Ransac matching: Simultaneous registration and segmentation," in *ICRA*, May 2010, pp. 1905–12.

[13] 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.

[14] T. Lang and M. Toussaint, "Planning with noisy probabilistic relational rules," *JAIR*, vol. 39, pp. 1–49, 2010.

[15] Q. V. Le, D. Kamm, A. F. Kara, and A. Y. Ng, "Learning to grasp objects with multiple contact points," in *ICRA*, 2010, pp. 5062–5069.

[16] L. Y. Chang, J. R. Smith, and D. Fox, "Interactive singulation of objects from a pile," in *ICRA*, 2012, pp. 3875–3882.

[17] M. Gupta and G. Sukhatme, "Using manipulation primitives for brick sorting in clutter," in *ICRA*, 2012, pp. 3883–89.

[18] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient Optimization Techniques for Efficient Motion Planning," in *ICRA*, 2009, pp. 489–494.

[19] M. Kazemi, J.-S. Valois, J. Bagnell, and N. Pollard, "Robust Object Grasping using Force Compliant Motion Primitives," in *RSS*, July 2012.

[20] J. Bagnell, F. Cavalcanti, L. Cui, T. Galluzzo, M. Hebert, M. Kazemi, J. Libby, T. Liu, N. S. Pollard, M. Pivtoraiko, J.-S. Valois, M. Klingensmith, and R. Zhu, "An integrated system for autonomous robotics manipulation," in *IROS*, October 2012, pp. 2955–2962.