Mechanical Search: Multi-Step Retrieval of a Target Object Occluded by Clutter

Michael Danielczuk*[1], Andrey Kurenkov*[2], Ashwin Balakrishna[1], Matthew Matl[1], Roberto Martín-Martín[2], Animesh Garg[2], Silvio Savarese[2], Ken Goldberg[1]

***Abstract*— When operating in unstructured environments such as warehouses, homes, and retail centers, robots are frequently required to interactively search for and retrieve specific objects from cluttered bins, shelves, or tables. Distinct from prior work on grasping in clutter, *Mechanical Search* describes a class of tasks where the number of actions required to locate and extract the target object scales with the size of the clutter set. In this paper we formalize this class and study an instance where distractor objects are heaped over the target object in a bin. The robot uses an RGBD perception system and control policies to iteratively select among, parameterize, and perform 3 actions – push, suction, grasp – until the target object is successfully retrieved. We present a detailed study of 5 algorithmic policies for mechanical search, with over 15,000 simulated trials and 250 physical trials for heap sizes ranging from 10 to 20 objects. Results suggest that success can be achieved in this long-horizon task with algorithmic policies in over 95% of instances and that the number of actions required scales approximately linearly with the size of the clutter set.**

I. Introduction

In unstructured settings such as warehouses or homes, robotic manipulation tasks are often complicated by the presence of dense clutter that obscures objects of interest. Whether a robot is trying to retrieve a can of soda from a stuffed refrigerator or pick a customer's order from warehouse shelves, the target object is often either not immediately visible or not easily accessible for the robot to grasp. In these situations, the robot must interact with the environment to localize the target object, manipulate the environment to uncover available grasps, and plan a grasp on the target (see Figure 1). *Mechanical Search* describes a class of tasks where the number of actions required to locate and extract the target object scales with the size of the clutter set. These situations require robots to move or remove occluding objects to uncover grasps on the target object, making it important to consider a number of sub-problems, including visual reasoning, task, motion, and grasp planning, and action execution.

Significant progress has been made in recent years on these sub-problems. Deep-learning methods for segmenting and recognizing objects in images have demonstrated excellent performance in challenging domains [16, 31, 43]. In a similar vein, recent grasp planning methods have leveraged convolutional neural networks (CNNs) to plan and execute high-quality grasps directly from RGB-D sensor data [13, 24, 26]. By combining object segmentation and recognition methods with high-level action selectors that decide (A) which objects to manipulate and (B) which low-level grasping or pushing policies to use for planning, multi-step policies can interactively search for a target object and extract if from clutter.

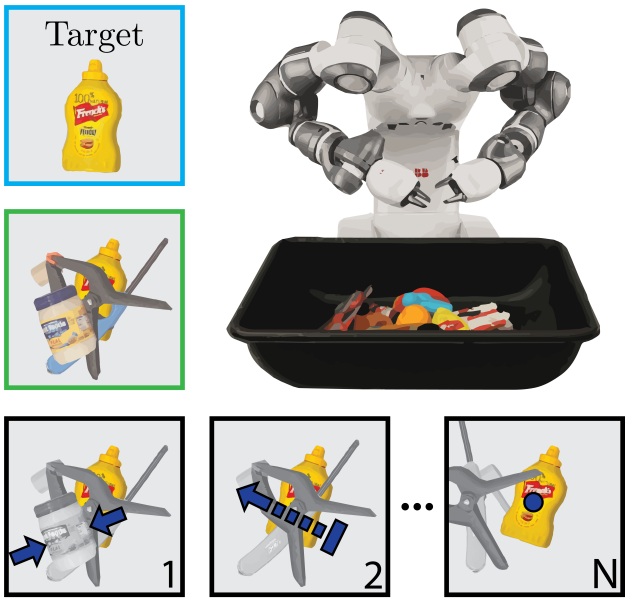

Fig. 1: In order to uncover and extract a target object from the bin, the system selects between grasping objects with a parallel-jaw gripper, pushing objects, or grasping objects with a suction-cup gripper until the target object is removed or a termination condition is met.

In this paper, we propose a unified, flexible framework for integrating perception, action selection, and manipulation policies to address an instance of the mechanical search problem. Specifically, the main contributions are:

1) A generalized formulation of the family of *Mechanical Search* problems and one particular problem instance, in which we retrieve target objects from cluttered bins using a combination of grasping and pushing actions.
2) A system framework for creating mechanical search policies from algorithmic sub-components.
3) An instantiation of this framework using depth-based object segmentation, single-shot image recognition, pre-existing low-level grasp and push planners, and implementations of five high-level action selection methods that choose which object to grasp or push.
4) Extensive simulation and physical experiments evaluating the performance of the five implementations on diverse sets of objects. Our evaluation also compares these implementations to human supervision for action selection.

* Authors have contributed equally and names are in alphabetical order.
[1]University of California, Berkeley. [2]Stanford University

II. Background and Related Work

This paper presents a generalized definition of the Mechanical Search problem and a unified algorithmic framework for a particular instance of the problem: retrieving a target object from a cluttered bin of unknown objects. This problem requires multi-step interaction with the environment with interactive perception and reactive policies for manipulation. We review the progress in these problems independently and contextualize them for Mechanical Search.

Perception for Sequential Interaction: Searching for an object of interest in a static image has been a central problem in active vision [30, 40, 47]. There has also been work on exploring camera positioning for improving visual recognition (i.e. *active perception* [1, 2]) and embodied interactions to explore (i.e. *interactive perception* [4, 14]). However, these approaches do not consider long grasping sequences and manipulation of arbitrary objects as would be required in Mechanical Search.

Semantic segmentation is one approach for localizing a target object in clutter. Recent deep learning based methods achieve remarkable success in segmentation of RGB [38, 41] and depth images [6], as well as in localizing visual templates in uncluttered [22, 49] and cluttered scenes [31, 43]. Furthermore, one-shot learning approaches using Siamese Networks for matching a novel visual template in images [22, 49] can translate well to pattern recognition in clutter [31, 43]. We build on Mask R-CNN [16] by training a variant for depth-image based instance segmentation and leverage a siamese network for target template matching for localization.

Grasping and Manipulation in Clutter: The other essential piece for enabling Mechanical Search is manipulation in clutter. Past approaches to this problem can be broadly characterized as model-based with geometric knowledge of the environment [3, 32, 44] and model-free with only raw visual input [20, 25, 42]. Recent studies have leveraged CNNs for casting grasping as a supervised learning problem with impressive results [10, 18, 19, 24, 27, 48, 52]. Additionally, methods that do not require access to privileged information, such as 3D geometric models, at test time have shown impressive generalization performance [19, 27]. Pushing and singulation can facilitate grasping in cluttered scenes [5, 8, 17]. Techniques for grasping in clutter, either as open-loop prediction or as closed-loop continuous control have been studied, but none execute multi-step plans, which is critical to attain successful grasps on target objects that are either not visible or inaccessible [19, 33]. In contrast, we formulate Mechanical Search as an interactive search problem in significant clutter, necessitating a multi-step process combining grasping and pushing actions.

Sequential Decision Making: Sequential composition of primitives to enable long-term environment interaction is a critical challenge in Mechanical Search. A common way to manage task complexity is to impose hierarchy on the control policy. The idea of using hierarchical models for complex tasks has been widely explored in both reinforcement learning and robotics [21, 45]. In RL, the *options* framework composes primitive actions into multi-step actions, enabling policy learning at higher-level semantic and/or temporal abstraction [12, 23, 35, 46]. Training such multi-level models can be computationally expensive and has been limited to either simulated or elementary physical tasks [9, 51].

Search Based Methods: Traditional task planning approaches abstract away perception and focus on high-level task plans and low-level state spaces [11, 44]. For instance, in robotic applications, hierarchical methods have been used to learn task planning strategies while abstracting away low-level motion planning [37, 44, 50].

However, high-level planning requires complete domain specification a priori, and complex geometric and free space reasoning makes this approach applicable only to uncluttered environments with few objects, such as a tabletop with one or two objects. It is worth noting that similar results are difficult to achieve in model-free settings.

A similar problem has been studied in the context of mobility under problem domains of target-driven and semantic visual navigation [15, 34, 53]. These studies look at finding visual targets in unknown environments without maps through sensory pattern matching. The work by Gupta et al. [14] is the closest to the approach considered in this paper. Their work also considers the problem of searching for a specific object using pushing and grasping actions, but when the objects are arranged in a shelf. This work builds on the problem in [14] to significantly more cluttered settings, and evaluates performance in terms of target object retrieval efficiency.

III. Mechanical Search: Problem Formulation

In Mechanical Search, the agent's objective is to retrieve a specific target object from a physical environment. Specifically, it involves a robotic system with at least one sensor such as an RGB-D camera, a set X of known objects with information on their geometric (shape) and physical properties (color, texture, mass), a (possibly unbounded) set Y of unknown objects, and a set of available actions A. Given a bounded environment E, a desired object x from X, and a collection (e.g., unordered heap or stack) of objects H_S in E comprised of objects from X and Y, the goal of Mechanical Search is to *retrieve* x: move x from H_S to a desired final position outside E or determine that x is not in H_S.

In this paper, we focus on a specific instance of Mechanical Search in the context of a cluttered set of objects in an single bin. For this problem, we further define:

- **Input:** A set of k images of the target object $x \in X$ in varied poses. No further information about either the target or the environment (such as 3D models) is provided.
- **State:** The environment E, a rectangular industrial bin, containing N objects $\{\mathcal{O}_{1,t}, ..., \mathcal{O}_{N,t}\}$ at time t.
- **Observation:** An RGB-D image of the scene at time t, taken from an overhead RGB-D sensor.
- **Actions:** There are three valid actions in this domain, defined in continuous space:

– **Parallel Jaw Grasping**: A center point $\mathbf{p} = (x, y, z) \in \mathbb{R}^3$ between the jaws, and an angle in the plane of the table $\phi \in \mathcal{S}^1$ representing the grasp axis, as defined in [27].

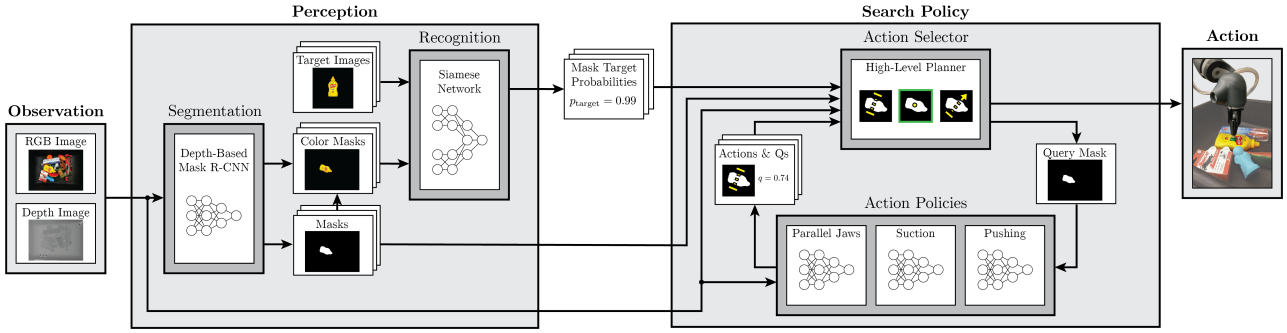

Fig. 2: System architecture for mechanical search to retrieve a target object from a cluttered bin of unknown objects. At each timestep, the RGB-D image of the bin is segmented using a variant of Mask R-CNN trained on synthetic depth images. The colorized masks are each assigned a probability of belonging to the target object using a Siamese Network as a pattern comparator. These masks are then fed to an Action Selector. The selector chooses which object to manipulate and passes that object mask as context to all the action policies in a given set, comprised of grasping, suction and pushing action policies in this case. These policies each compute an action with an associated quality score and pass them back to the action selector. The action selector then chooses an action and executes it on the physical system. This continues until the target object is retrieved. In simulation, the perception pipeline is removed and planners operate on full-state information rather than object masks.

– **Suction Grasping**: A target point $\mathbf{p} = (x, y, z) \in \mathbb{R}^3$ and spherical coordinates $(\phi, \theta) \in \mathcal{S}^2$ representing the axis of approach of the suction cup, as defined in [28],
– **Pushing**: A linear motion of the robot end-effector between two points $\mathbf{p}$ and $\mathbf{p}' \in \mathbb{R}^3$.

• **Objective:** Retrieve x from E, by directly grasping x and lifting it out of the bin, or by manipulating the environment until object x is accessible for grasping.

IV. Perception and Decision System

As shown in Figure 2, we implement a full robotic system both in simulation and the physical world with separate stages for perception and search policy execution.

A. Perception

The system first processes the RGB-D image via object segmentation followed by target recognition. This outputs a set of object segmentation masks and a target mask if a high confidence match is found. If no high confidence match is found, the perception system reports that no masks match the target object.

Object Segmentation: We first compute a mask for each object instance. Each mask is a binary image with the same dimensions as the input RGB-D image. These masks are computed with SD Mask R-CNN, a variant of Mask R-CNN trained exclusively on synthetic depth images. It converts a depth image into a list of unclassified binary object masks, which can then be used with new object sets without retraining. Recent results suggest that depth cues alone may be sufficient for high-performance segmentation, and that this network can generalize to novel objects, which is beneficial in a scenario where only the target object is known.

Target Recognition: Next, the set of masks is combined with the RGB image to create color masks of each object. Then each color mask is then evaluated against the target object image set to estimate the probability of the mask being of the target object. Each of the m color masks is cropped, scaled, rotated, and then compared to each of the k images in the target object image set using a Siamese network [22]. For each pair of inputs, the Siamese network outputs a recognition confidence value between 0 and 1, and a mask's recognition confidence score is its maximum recognition confidence value over the k target object images. If the mask with the highest score has a score above recognition confidence threshold t_r, the mask is labeled as the target object. Otherwise, the component reports that no masks match the target object.

B. Search Policy

Given the RGB-D image and the output of the perception pipeline, the system executes the next action in the search procedure by selecting the object to act on and the action to perform on it. Our approach to the instantiation of mechanical search described in Section III for bin picking includes searching for actions in three continuous spaces (parallel jaw grasp, suction grasp and push). However, more complex instances of mechanical search (e.g. search for an object in a house) could have even more complex search spaces (e.g. navigation). To allow our method to scale to these more complex instances, we propose a hierarchical approach: (1) an action selector that queries a set of action policies on a specific object for a particular action and associated quality metric and (2) three action policies that correspond to the three possible actions in the problem formulation.

Action Selection: The search policy first determines which object masks to send to the action policies. Then, using the actions and associated quality metric returned by the low level policies, the high level planner determines whether to execute the action in the environment.

The action selector takes as input from the perception system the set of all m visible object masks ($[o_1, \ldots o_m]$), possibly including an object mask that is positively identified as the target object (o_T), from the perception system. It then selects an action policy and a goal object o_{goal} from $[o_1, \ldots o_m]$ and sends the action policy a query $q(o_{goal})$. The action policy p_i responds with an action $a_i = p_i(o_{goal})$ and a quality metric $Q(a_i, o_{goal})$ for the action, which is used to decide whether to execute the action.

Action Policies: Each action policy p_i takes as input an object mask from the action selector (o_{goal}) and the observation from the perception pipeline and returns an action $a_i = p_i(o_{goal})$ and a quality metric $Q(a_i, o_{goal})$. In simulation, observations are ground-truth object masks, while in physical

experiments, observations are depth images obtained using a depth sensor. The set of action policies in our system are:

Parallel Jaw Grasping: For simulation experiments, pre-computed grasps are indexed from a Dex-Net 1.0 policy [29], and the grasp with the highest predicted quality on o_{goal} is returned as the action. The returned quality metric is the grasp quality estimate from Dex-Net 1.0.

For physical experiments, parallel-jaw grasps are planned using a Dex-Net 2.0 Grasp Quality CNN (GQ-GNN) [26]. To plan grasps for a single object in a depth image, grasp candidate sampling is constrained to the goal object's segmentation mask. The GQ-CNN evaluates each candidate grasp and returns the grasp with the highest predicted quality and its associated quality metric from GQ-CNN.

Suction Grasping: For simulation experiments, grasp planning is done with a Dex-Net 1.0 policy [29] the same as Parallel Jaw Grasping. For physical experiments, suction cup grasps are planned with a Dex-Net 3.0 GQ-CNN [28], with mask-based constraints to plan grasps only on object goal object's segmentation mask. The GQ-CNN evaluates each candidate grasp and returns the grasp with the highest predicted quality and its associated quality metric.

Pushing: As introduced in Section III, a pushing action is parameterized by a start point $\mathbf{p} \in \mathbb{R}^3$ and an end point $\mathbf{p}' \in \mathbb{R}^3$. The pushing action policy, similar to one in [8], selects $\mathbf{p}'$ as the most free point in the bin. This point is computed by taking the signed distance transform of a binary mask of the bin walls and objects, finding the pixel with the maximum signed distance value, and deprojecting that pixel back into $\mathbb{R}^3$. Given an object to push, $\mathbf{p}$ is then selected so that the gripper is not in collision at $\mathbf{p}$, the line from $\mathbf{p}$ to $\mathbf{p}'$ passes through the object's center of mass, and the push direction is as close as possible to the direction of the most free point in the bin. The pushing policy returns the push satisfying the above constraints as its action if one exists. The returned quality metric is 1 if a valid push exists and 0 if not. If no valid push exists, this is reported by the policy.

V. Action Selection Policies

All action selection methods have criteria for action execution and use input from the perception system as described above to generate an object priority list. Each action selection method generates a priority list in a different way but all have the same action criteria. For all action selection methods described here, a grasp action is executed if the quality metric returned by the action policy exceeds $t(o)$, the grasp confidence threshold for object mask o. The grasp confidence threshold for the object positively identified as the target object o_T is given by $t(o_T) = t_{\text{thresh}}$. For policies without pushing, $t(o) = t_{\text{thresh}} \forall o$, while for policies with pushing, $t(o) = t_{\text{high}} \forall o \neq o_T$. Policies with pushing can be more conservative in their choice of grasps, so they use a higher grasp confidence threshold t_{high} for non-target objects. A push action is performed if the returned action has a quality metric of 1 (the returned action represents a valid push).

Each action selection method iterates through its priority list, queries the grasping action policies for each object mask, and executes the returned action with the highest quality metric among the two grasping policies if it satisfies the action criteria. If the target object is grasped, the policy terminates and reports a success. If no grasping action satisfies the criteria and the policy does not have pushing, the policy terminates and reports a failure. If the policy does have pushing, it iterates through its priority list, queries the pushing action policy for each object mask, and executes the first action that satisfies the criteria. If no pushing action satisfies the criteria, or if a pushing action has been selected more than three consecutive times, the policy terminates with a failure.

Action Selection Methods: The action selection methods are distinguished by whether or not they have pushing as an available action policy and by their generated object priority list. The latter is as follows:

1) **Random Search:** This method prioritizes objects randomly, with no preference for the target object.
2) **Preempted Random Search (with and without pushing):** It always prioritizes o_T, the object positively identified as the target object, and orders the other objects randomly.
3) **Largest-First Search (with and without pushing):** This method always prioritizes o_T, the object positively identified as the target object, and ranks the other objects by their visible area. If the target object isn't visible, this strategy will increase the likelihood of removing objects that may be occluding the target object.

Termination Criteria: In addition to the termination criteria outlined above (terminate and return success if target object grasped, return failure if no good grasp/valid push found), we impose two more termination conditions on our policies which cause them to return a failure: (1) $2N$ timesteps have elapsed, where N is the initial number of objects in the bin and (2) The target object is inadvertently removed from the work space when another object is grasped or pushed.

VI. Experiments

A. Simulation

Heap Generation: Three datasets of simulated heaps are generated, each containing 1000 heaps of N objects, for $N \in \{10, 15, 20\}$. Heaps are generated by sampling: 1) N objects from a dataset of over 1600 3D models, 2) a heap center around the center of the bin, and 3) planar pose offsets for each object around the heap center. Then, using the Bullet Physics Engine [7], sampled objects are dropped one by one into the bin from a fixed height at their pose offset from the heap center, and all objects are allowed to come to rest (i.e. all velocities of all objects go to zero). Once all objects have been added to the heap, the modal and amodal segmentation masks for each object are rendered from the camera's perspective. The modal segmask of an object is a segmask of the portion of the object visible from the perspective of the camera (accounting for occlusions), while the amodal segmask of an object is a segmask of the object's exact position in the scene given ground truth information from the simulation environment. Using these masks, the target object is chosen to be the object with the smallest ratio between modal and

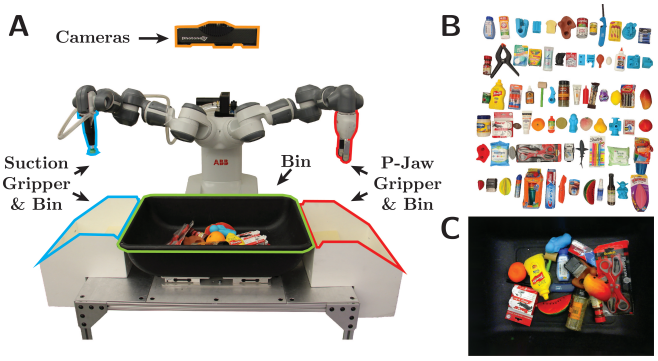

Fig. 3: **(A)** Front view of the robot and bin setup. The black bin is the primary bin in which heaps are initialized, and the white bins provide space for the robot to deposit grasped items. **(B)** The 75 objects used in physical experiments. **(C)** A sample heap of 15 objects used in the physical experiments.

amodal segmask area as the target (i.e. the least visible object in the bin).

Rollouts: To simulate grasp actions, we use the same approach as in [25]; using wrench space analysis, we determine whether or not an object can be lifted from the heap [36, 39]. If the object can be lifted, a constant upward force is applied to the object's center of mass until it leaves the bin, and the remaining objects are allowed to come to rest. To simulate push actions, we check that the gripper can be placed in the starting location without collisions, and only execute pushes if this is the case. Then we place a 3D model of the closed gripper in the physics simulator and move it from the start point to the end point of the push.

B. Physical Experimental Procedure

Heap Generation: We randomly sample 50 heaps of 15 items each from a set of 75 common household objects with relatively simple shapes, such as boxes and cylinders, as well as more complex geometries, such as plastic climbing holds and scissors (see Fig. 3). We also included several 3D-printed items, which presented a challenge for both segmentation and target object recognition due to their unusual shapes and uniform texture. A target object is chosen at random from each 15 item heap. Then, in order to generate adversarial bin configurations, each rollout was initialized by first shaking the target object in a box to randomize its pose and dumping into the center of the bin, and then shaking the other fourteen objects and pouring them over the target object.

Policy Rollouts: We execute pushing and grasping actions on an ABB YuMi robot equipped with suction-cup and parallel-jaw grippers (see Figure 3). Actions generated by the search policy are transformed into a sequence of poses for the robot's end-effectors, and we use ABB's RAPID linear motion planner and controller to execute these motions.

Human Supervisor Rollouts: For comparison, we also benchmark a human supervisor's performance as an action selector. The human, who does not know the identity of the target object, is given the same perceptual output as the robot (an RGB-D image and a set of scored masks) and uses a point-and-click interface to select an object and choose whether to grasp or push that object. Grasps and pushes are planned and executed with the same action policies as described above.

C. Evaluation Metrics

We evaluate each policy according to its reliability and efficiency in target object retrieval. Reliability is defined as the frequency at which the target object is successfully extracted, while efficiency is defined as the mean number of actions taken to successfully extract the target object. For each experiment (simulated or real, number of objects in the heap), we recorded the number of successes and failures, as well as statistics regarding the number of actions taken.

VII. Results

A. Simulation Results

We tested each action selection method (Sec. V) in simulation using heaps generated using the method described in VI and running each policy until termination on each heap. A total of 15000 simulation experiments were conducted over all policies. The results for each of the five policies are shown in Figure 4. Fig Figure 4(C) shows the mean number of actions needed for each policy as a function of heap size. These results suggest that (1) the four policies are significantly better than random, (2) that extracting the target object becomes more difficult as heap size increases, and (3) the mean number of actions needed for each policy appears to scale linearly with heap size, although the rate of increase is not constant across policies.

Figure 4(B) shows cumulative successful extractions for a given amount of actions (number of successful extractions in that many actions or fewer). All policies have success rates of 90% or higher on 15 object heaps. The cumulative success plot suggests that grasping the target object when possible provides improvement over the random policy, and that a high-level policy that intelligently chooses objects to grasp when the target object does not have a good grasp further increases efficiency. Given 5 or fewer actions to extract the target object, the largest-first policies succeed on 50% of the heaps, whereas the preempted random and random policies are only able to succeed on 30% and 10% of the heaps.

The plot also suggests that pushing can increase overall success rate, as the policies that included pushing succeeded on at least 3% more heaps than those policies without pushing. Pushing can be beneficial in cases where grasps are unavailable, and that there is a trade-off between choosing grasp actions and push actions.

In these rollouts, only 5% of all actions attempted are pushes, as opposed to 29% parallel jaw grasps and 66% suction grasps, on ten object heaps and the percentage of push actions decreases further with increasing number of objects to just 3% for 15 object heaps.

In simulation, failures account for 6-12% of all rollouts. These failures fall into three categories: 1) the policy fails to plan an action (e.g., no actions are available on any remaining objects with a quality metric above the threshold), 2) the target object is inadvertently removed from the bin, despite an attempted action that was not a grasp on the target object, or 3) the policy reaches the maximum number of timesteps given to extract the object ($2N$ timesteps, where N is the initial number of objects in the heap).

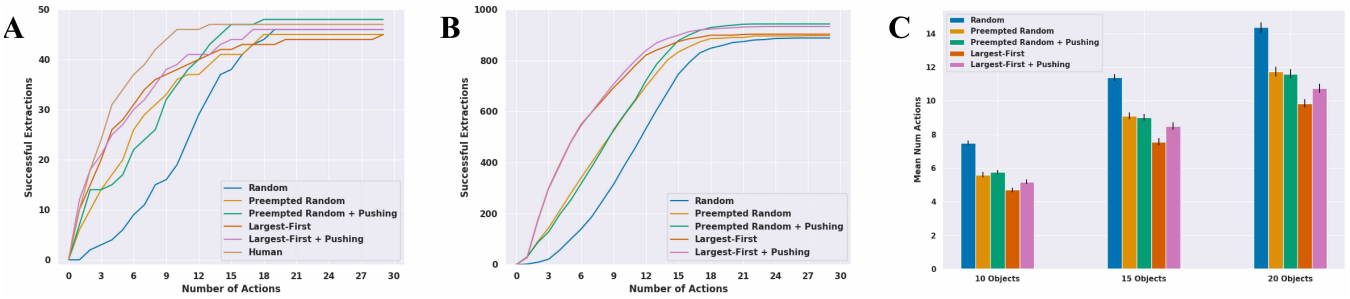

Fig. 4: Performance of policies on **(A)** real heaps of 15 objects, **(B)** simulated heaps of 15 objects, and **(C)** simulated heaps of 10, 15, and 20 objects over a total of 15,000 simulated rollouts and 300 physical rollouts. The largest-first search policies are the most efficient, and are able to extract the target object in the least number of actions. All policies have similar reliability, although pushing shows potential to avoid more failures in simulation. The human had no knowledge of the target object and was asked to choose a grasp or push action given the same input from the perception pipeline as the other policies.

Failure mode (1) accounts for 85-90% of all failures, while (2) accounts for nearly all of the remaining failures. Mode (1) typically happens because objects are moved to the corners of the bin, making it difficult to plan collision-free grasps, leading to lower quality metrics returned by the action policies. Mode (2) occurs more often for the policies that include pushing actions, since pushing in clutter can occasionally lead to objects being pushed up and over the bin walls. It is also possible that the target object is lying on top of the object being grasped, resulting in the target object being inadvertently removed from the bin. Mode (3) never actually occurs in any of our experiments, and the maximum timesteps cutoff is intentionally set high to exhaust the policy of actions.

B. Physical Results

Figure 4(A) shows results on the physical system. A total of 250 physical experiments were conducted over all policies. All policies retrieved the target object within the given number of timesteps at least 90% of the time, and success rates were not statistically different between policies. However, the cumulative success curves suggest similar trends to those seen in the simulation results, with the largest-first policies outperforming the preempted random and random policies in terms of efficiency. Given 5 or fewer actions, the largest-first policies can again extract the target on 50% of the heaps, while the preempted random and random policies can only extract the target object on 40% and 10% of the heaps in 5 or fewer actions.

In contrast to simulation, where the initial state of a heap can be replicated exactly for each policy, in the physical system, there is a distribution over the initial state for a given heap. This distribution can result in varying difficulty for different policies on the same heap. For example, a target object may be completely covered by other objects when one policy is presented with a given heap, but for another policy, the target may be partially or fully visible in the initial state. Thus, policies may occasionally get “lucky" or “unlucky" with respect to the target object visibility in the initial state.

Failure cases for the physical heaps are very similar to those in simulation: 85% of failures arise from the policy being unable to plan an action. However, in physical experiments, out of the 300 heaps evaluated for all policies, only 3 policies fail due to timing out. Failure to plan actions is almost always due to the target object lying flat on the bottom of the bin (e.g., the dice, sharpie pens, or another blister-pack object), making it difficult to obtain an accurate segmentation. Another common reason for failure to plan actions is when no mask is able to be identified as the target object with high confidence, which often occurs for 3D printed objects. In general, the perception system is the primary cause of most failures on the physical system; objects are always separated when the policy is unable to plan an action or repeatedly failed on a grasp (which resulted in the maximum timesteps threshold being reached).

C. Human Supervisor. The human supervisor outperforms all policies presented here, but still requires an average of about 4 actions to extract the target object. Reliability did not increase much due to the aforementioned failures in the perception system, but efficiency increased due to a more intelligent action selector. Specifically, we noticed that a human operator chose to push far more frequently (15% of all actions, compared to 5% for the other action policies with pushing), and especially when objects were heaped in the center of the bin and the target was not visible. These pushes tend to spread many objects out over the bottom of the bin, as opposed to a grasping action that would remove only a single object from the top of the heap.

VIII. Discussion and Conclusion

We present a general formulation for mechanical search problems and describe a framework for solving the specific problem of extracting a target object from a cluttered bin. For this setting, we introduce a variety of action selection methods and conduct detailed analysis of both their efficiency and reliability in both simulated and physical experiments. While the best action selection method (largest-first) is much more efficient than random search and provides a solid baseline, a human supervisor can still achieve 30% higher efficiency.

This performance gap suggests a number of open questions, such as: Can better perception algorithms improve performance? Can we formulate different sets of low level policies to increase the diversity of manipulation capability? Can we model Mechanical Search as a long-horizon POMDP and apply deep reinforcement learning in simulation and/or in physical trials to learn more efficient policies? We are actively exploring these and related questions.

REFERENCES

[1] A. Aydemir, K. Sjöö, J. Folkesson, A. Pronobis, and P. Jensfelt, "Search in the real world: Active visual object search based on spatial relations", in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, 2011, pp. 2818–2824.

[2] R. Bajcsy, "Active perception", *Proceedings of the IEEE*, vol. 76, no. 8, pp. 966–1005, 1988.

[3] D. Berenson and S. S. Srinivasa, "Grasp synthesis in cluttered environments for dexterous hands", in *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, IEEE, 2008, pp. 189–196.

[4] J. Bohg, K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, and G. S. Sukhatme, "Interactive perception: Leveraging action in perception and perception in action", *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1273–1291, 2017.

[5] L. Chang, J. R. Smith, and D. Fox, "Interactive singulation of objects from a pile", in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, 2012, pp. 3875–3882.

[6] X. Chen, K. Kundu, Y. Zhu, H. Ma, S. Fidler, and R. Urtasun, "3d object proposals using stereo imagery for accurate object class detection", *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 5, pp. 1259–1272, 2018.

[7] E. Coumans and Y. Bai, *Pybullet, a python module for physics simulation, games, robotics and machine learning*, `http://pybullet.org/`, 2016–2017.

[8] M. Danielczuk, J. Mahler, C. Correa, and K. Goldberg, "Linear push policies to increase grasp access for robot bin picking", in *Proc. IEEE Conf. on Automation Science and Engineering (CASE)*, IEEE, 2018.

[9] Y. Duan, M. Andrychowicz, B. C. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-Shot Imitation Learning", *arXiv preprint arXiv:1703.07326*, 2017.

[10] K. Fang, Y. Zhu, A. Garg, A. Kurenkov, V. Mehta, L. Fei-Fei, and S. Savarese, "Learning task-oriented grasping for tool manipulation from simulated self-supervision", *arXiv preprint arXiv:1806.09266*, 2018.

[11] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving", *Artificial intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[12] R. Fox, S. Krishnan, I. Stoica, and K. Goldberg, "Multi-Level Discovery of Deep Options", in *preprint arXiv:1703.08294*, 2017.

[13] M. Gualtieri, A. ten Pas, K. Saenko, and R. Platt, "High precision grasp pose detection in dense clutter", in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, IEEE, 2016, pp. 598–605.

[14] M. Gupta, T. Rühr, M. Beetz, and G. S. Sukhatme, "Interactive environment exploration in clutter", in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, IEEE, 2013, pp. 5265–5272.

[15] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation", *arXiv preprint arXiv:1702.03920*, vol. 3, 2017.

[16] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn. arxiv preprint arxiv: 170306870", 2017.

[17] T. Hermans, J. M. Rehg, and A. Bobick, "Guided pushing for object singulation", in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, IEEE, 2012, pp. 4783–4790.

[18] E. Jang, S. Vijayanarasimhan, P. Pastor, J. Ibarz, and S. Levine, "End-to-end learning of semantic grasping", in *Conf. on Robot Learning (CoRL)*, 2017.

[19] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, *et al.*, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation", *arXiv preprint arXiv:1806.10293*, 2018.

[20] D. Katz, A. Venkatraman, M. Kazemi, J. A. Bagnell, and A. Stentz, "Perceiving, learning, and exploiting object affordances for autonomous pile manipulation", *Autonomous Robots*, vol. 37, no. 4, pp. 369–382, 2014.

[21] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey", *The Int'l Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[22] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition", in *ICML Deep Learning Workshop*, vol. 2, 2015.

[23] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation", in *Proc. Advances in Neural Information Processing Systems*, 2016, pp. 3675–3683.

[24] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps", *Int. Journal of Robotics Research (IJRR)*, vol. 34, no. 4-5, pp. 705–724, 2015.

[25] J. Mahler and K. Goldberg, "Learning deep policies for robot bin picking by simulating robust grasping sequences", in *Conf. on Robot Learning (CoRL)*, 2017, pp. 515–524.

[26] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics", *Proc. Robotics: Science and Systems (RSS)*, 2017.

[27] ——, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics", *CoRR*, vol. abs/1703.09312, 2017. arXiv: `1703.09312`.

[28] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg, "Dex-net 3.0: Computing robust robot suction grasp targets in point clouds using a new analytic model and deep learning", *arXiv preprint arXiv:1709.06670*, 2017.

[29] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg, "Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards", in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, IEEE, 2016, pp. 1957–1964.

[30] C. Michaelis, M. Bethge, and A. Ecker, "One-shot segmentation in clutter", in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, Stockholmsmässan, Stockholm Sweden: PMLR, 2018, pp. 3549–3558.

[31] C. Michaelis, M. Bethge, and A. S. Ecker, "One-shot segmentation in clutter", *arXiv preprint arXiv:1803.09597*, 2018.

[32] M. Moll, L. Kavraki, J. Rosell, *et al.*, "Randomized physics-based motion planning for grasping in cluttered and uncertain environments", *IEEE Robotics & Automation Letters*, vol. 3, no. 2, pp. 712–719, 2018.

[33] D. Morrison, P. Corke, and J. Leitner, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach", *arXiv preprint arXiv:1804.05172*, 2018.

[34] A. Mousavian, A. Toshev, M. Fiser, J. Kosecka, and J. Davidson, "Visual representations for semantic target driven navigation", *arXiv preprint arXiv:1805.06066*, 2018.

[35] R. Parr and S. J. Russell, "Reinforcement learning with hierarchies of machines", in *NIPS*, 1998.

[36] A. ten Pas and R. Platt, "Using geometry to detect grasp poses in 3d point clouds", in *Robotics Research*, Springer, 2018, pp. 307–324.

[37] C. Paxton, F. Jonathan, M. Kobilarov, and G. D. Hager, "Do what i want, not what i did: Imitation of skills by planning sequences of actions", in *IROS*, 2016.

[38] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár, "Learning to refine object segments", in *European Conference on Computer Vision*, Springer, 2016, pp. 75–91.

[39] D. Prattichizzo and J. C. Trinkle, "Grasping", in *Springer handbook of robotics*, Springer, 2008, pp. 671–700.

[40] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection", in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[41] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks", in *Advances in neural information processing systems*, 2015, pp. 91–99.

[42] A. Saxena, L. L. Wong, and A. Y. Ng, "Learning grasp strategies with partial shape information.", in *AAAI*, vol. 3, 2008, pp. 1491–1494.

[43] A. Shaban, S. Bansal, Z. Liu, I. Essa, and B. Boots, "One-shot learning for semantic segmentation", *arXiv preprint arXiv:1709.03410*, 2017.

[44] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer", in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, IEEE, 2014, pp. 639–646.

[45] J. Sung, B. Selman, and A. Saxena, "Learning sequences of controllers for complex manipulation tasks", in *International Conference on Machine Learning*, Citeseer, 2013.

[46] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning", *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[47] K. E. Van de Sande, J. R. Uijlings, T. Gevers, and A. W. Smeulders, "Segmentation as selective search for object recognition", in *Computer Vision (ICCV), 2011 IEEE International Conference on*, IEEE, 2011, pp. 1879–1886.

[48] U. Viereck, A. t. Pas, K. Saenko, and R. Platt, "Learning a visuomotor controller for real world robotic grasping using simulated depth images", *arXiv preprint arXiv:1706.04652*, 2017.

[49] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.*, "Matching networks for one shot learning", in *Advances in Neural Information Processing Systems*, 2016, pp. 3630–3638.

[50] J. Wolfe, B. Marthi, and S. J. Russell, "Combined task and motion planning for mobile manipulation", in *ICAPS*, 2010.

[51] D. Xu, S. Nair, Y. Zhu, J. Gao, A. Garg, L. Fei-Fei, and S. Savarese, "Neural task programming: Learning to generalize across hierarchical tasks", in *ICRA*, 2018.

[52] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, *et al.*, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching", *arXiv preprint arXiv:1710.01330*, 2017.

[53] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning", in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, IEEE, 2017, pp. 3357–3364.