Push-MOG: Efficient Pushing to Consolidate Polygonal Objects for Multi-Object Grasping

Shrey Aeron[1], Edith Llontop[1], Aviv Adler[1], Wisdom C. Agboh[1,2], Mehmet Dogar[2], and Ken Goldberg[1]

Abstract— Recently, robots have seen rapidly increasing use in homes and warehouses to declutter by collecting objects from a planar surface and placing them into a container. While current techniques grasp objects individually, Multi-Object Grasping (MOG) can improve efficiency by increasing the average number of objects grasped per trip (OpT). However, grasping multiple objects requires the objects to be aligned and in close proximity. In this work, we propose *Push-MOG*, an algorithm that computes "fork pushing" actions using a parallel-jaw gripper to create graspable object clusters. In physical decluttering experiments, we find that Push-MOG enables multi-object grasps, increasing the average OpT by 34%. Code and videos are available at `https://sites.google.com/berkeley.edu/push-mog`.

I. INTRODUCTION

Efficient object manipulation is a key task in industrial, commercial, and domestic robotics, incorporating elements from motion planning, grasping, and task planning [1], [2], [3], [4]. In particular, many applications focus on the task of transferring a collection of objects from a surface into a bin or basket, for the purpose of either clearing the surface or preparing the objects for transportation to another location. This task is typically solved by picking one object at a time. However, an alternative approach of removing multiple objects at once with *multi-object grasping* (MOG) has received relatively little attention. MOG can provide better efficiency than traditional single-object grasping [5], [6], especially when the bin is relatively far away. Prior work on MOG mainly focused on developing techniques (such as MOG-Net [6]) for detecting and executing effective multi-object grasps in a scene. However, grasping multiple objects at once requires all of them to be close enough to fit within the gripper width, which may be rare when the objects are distributed randomly. Thus, to create graspable object clusters, pushing [7], [8] can be useful.

In this work, we propose *Push-MOG*, an algorithm that uses pushes to increase the average number of objects transported per trip to the bin. Push-MOG uses hierarchical clustering to identify clusters of objects which could potentially be grasped together. To execute stable pushes on polygonal objects with a parallel-jaw gripper, Push-MOG utilizes *fork pushing*, in which the jaw is opened to an appropriate width and the object is pushed perpendicular to the jaw, thus enabling a variety of stable pushes (see Fig. 3).

We also propose (mean) *objects per trip* (OpT) as a metric for evaluating the effectiveness of multi-object grasps. Thus,

[1]The AUTOLab at UC Berkeley (automation.berkeley.edu) `{aeron, goldberg} @berkeley.edu`
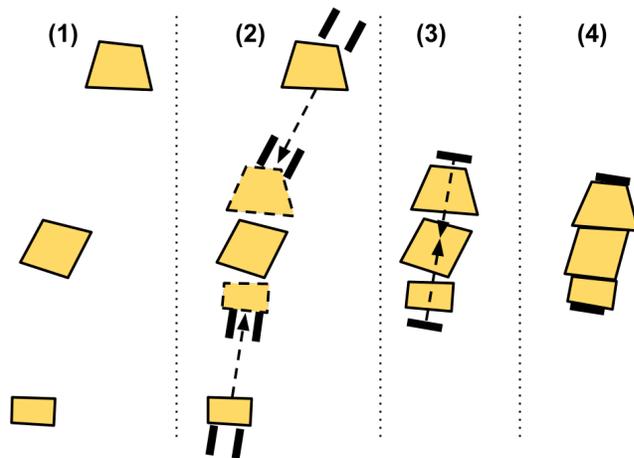[2]University of Leeds, UK

Figure 1: An example of Push-MOG pushing and grasping objects into a cluster. (1) A cluster is identified containing three objects; (2) two objects are moved toward the center object using fork pushing. (3) Once the cluster fits within the robot's gripper, (4) the gripper closes to create a tight fit and grasps the objects that will be transported to the bin.

in single-object grasping, the OpT is 1 (or slightly less than 1 due to failed grasp attempts), whereas MOG improves efficiency by increasing OpT. We evaluate Push-MOG in a physical environment consisting of randomly-distributed polygonal objects on a flat surface, in which the robot executes both grasps and pushes using a parallel-jaw gripper. We consider two baselines: (i) Single-Object Grasping (SOG), which removes objects at random, one at a time; (ii) and MOG-Net, which searches for multi-object grasps but does not apply rearrange objects with pushes. In physical robot experiments, we find that prior work (MOG-Net) achieves an OpT of ≈ 1, suggesting that naturally occurring multi-object grasps are rare, while Push-MOG achieves an OpT of 1.339, though at a cost of extra time for executing pushes. Since increasing OpT is more valuable when the bin is further away, Push-MOG can improve picking efficiency when trips are long. We make the same assumptions as Agboh et al. [5]: that the objects are extruded convex polygons, and have roughly uniform mass so that their centers of mass coincide with their geometric centers. However, unlike [5], we also assume the objects are stable and do not topple when pushed.

In this work, we make the following contributions:

- We formulate a new variant of the decluttering problem, in which a robot uses pushing actions to increase the number of objects it can move simultaneously to the bin. To solve this problem, we propose and implement Push-MOG. We also propose objects per trip (OpT) as a metric for evaluating the effectiveness of MOG

algorithms.

- We propose and implement *fork pushing* as a way to quickly and stably move objects on a work surface using a parallel-jaw gripper.
- We evaluate the performance of Push-MOG and two baselines (one using only single-object grasping and one using multi-object grasping without pushes) and find that Push-MOG increases OpT by 34%.

II. RELATED WORK

A. Multi-object grasps

Prior work on multi-object grasping [9], [10], [11], [12] used multi-fingered robot hands and proposed conditions for a stable grasp of objects, focusing on numerical simulations without physical robot multi-object grasps or a focus on automation. Chen et al. [13] propose a method for a robot to dip its gripper inside a pile of identical spherical objects, close it, and estimate the number of grasped objects. Shenoy et al. [14] focus on the same problem but with the goal of transporting the picked spherical objects to another bin.

Another line of work has focused on designing appropriate grippers for picking up multiple objects at once. Jiang et. al. [15] proposed a multiple suction cup vacuum gripper, while Nguyen et. al. [16] proposed a soft gripper based on elastic wires for multi-object grasping.

Our work is focused on efficiently rearranging scenes with pushes [17], [18], [19], [20] before multi-object grasping. Sakamoto et al. [21] proposed such a picking system where the robot first uses pushing [22], [23], [24], to move one cuboid to the other and thereafter grasp both cuboids. Our approach handles any multi-sided polygonal object and focuses on large-scale problems where clustering is required to decide which objects to push and where.

B. Pushing

Pushing is a fundamental primitive in robot manipulation [25]. Robotic pushing has been used to move a target object to a goal location [8], [26], rearrange a working surface [27], [28], and retrieve items from cluttered shelves [19], [20], [29].

During robotic pushing, uncertainty in the state, control, and model can result in failures [30], [31]. Prior work [32], [7] has trained networks to generate robust pushing actions. Others have taken an open-loop approach to generate these robust pushes [33], [34]. For example, Johnson et. al. [35] propose robustness metrics and use them to generate robust open-loop pushes. They exploit gripper and object geometries to generate these robust pushes. We take a similar open-loop pushing approach and leverage object and gripper geometries to generate robust pushes.

C. Point Clustering

Prior work in point clustering is well-established [36], [37], specifically in the hierarchical variant [38], [39], which aims to cluster data in an unsupervised fashion. This method constructs a distance-based tree that forms clusters of points from the bottom up which is then split at a 'height' to determine the final clusters. Additional work by Dao et al. [40] explores clustering with human-defined constraints. It introduces the idea of simplifying and solving NP-Hard problems by constraining them through a limit on inter-cluster distance metrics, such as maximum cluster diameter, or within-cluster variation [40]. In this work, we use a distance-based hierarchical clustering algorithm but split object clusters to satisfy gripper width constraints.

III. PROBLEM STATEMENT

Agboh et al [5] studied the problem of using an overhead camera and a parallel-jaw gripper to transport a collection O of extruded polygonal convex objects (prisms) from a flat work surface to a bin or box adjacent to the workspace by taking advantage of *multi-object grasping*, in which multiple objects are grasped simultaneously and transported together to the bin.

In this work, we extend this problem by adding the *fork push* action, in which the gripper pushes an object. By doing so, the robot can arrange the objects to create more efficient grasps. Since pushing is faster to execute than grasping and the objects are typically closer to each other than to the bin, using pushes to facilitate multi-object grasps can reduce the number of trips needed to clear the objects. As in Agboh et al. [6], we consider a frictional model of planar grasping, which not only corresponds better to real grasping problems but also permits a larger set of stable grasps, thus allowing the algorithm to consider a wider range of potential solutions.

A. State, Action, and Objective

Let the set of objects on the work surface be $O = \{o_0, o_1, \ldots, o_{N_o-1}\}$ where each o_i is a convex polygonal object and N_o is the number of objects on the work surface. The robot executes a sequence of *pushes* and transports (which we call *trips*); each push translates and/or rotates a single object in the workspace without colliding with others, while each trip grasps a set of closely clustered objects and transfers them to the bin.

We divide this problem into three sub-problems:

1) *Clustering:* Divide the objects into clusters. No cluster should contain more objects than can be stably grasped by the gripper.
2) *Pushing:* Push the objects in each cluster close together.
3) *Trip:* Perform grasps (containing as many objects as possible each time) and transport the grasped objects to the bin until no clusters remain.

Due to the inherent uncertainty of working with real objects, a grasp might not get every object in a cluster, so a cluster may need multiple grasps to clear completely. The clustering step is purely computational and does not involve any physical action.

Since we don't optimize an explicit objective in the clustering or pushing problems, we instead aim to guide the choice of algorithm to maximize overall OpT.

B. The Clustering Problem

In the clustering stage, the goal is to partition the set of objects O into clusters in such a way that (i) the objects in any cluster are close together and (ii) each cluster consists of objects that can be stably grasped together (provided they are pushed into an appropriate configuration). Finally, since the goal of clustering the objects is to reduce the number of grasps necessary to clear the workspace, a good overall clustering will partition the objects into as few clusters as possible, as each cluster roughly corresponds to one grasp. We denote the output of the clustering algorithm as a list of clusters C, and each individual cluster as $c \subseteq O$ (since a cluster is a set of objects).

In this problem, each object o's position is represented by its geometric centroid, which we denote M_o. This allows the application of an appropriately modified version of the hierarchical clustering algorithm (which clusters points).

An important characteristic of each object o is its *minimum final grasp diameter* d_o^*, corresponding to the minimum width of a stable single-object grasp on o. Then, given a cluster c, we denote the *minimum grasp diameter of c* as $d_c^* = \sum_{o \in c} d_o^*$. The gripper also has a width $d^{(\text{gr})}$, and if $d_c^* > d^{(\text{gr})}$ we regard cluster c as ungraspable (thus requiring division into smaller clusters)[1].

Also, it is important for the clusters to not interfere with each other during the pushing step, which might happen if clusters are spatially intermingled. While point clustering usually avoids this (such clusters are locally non-optimal), the grasp diameter constraint may make such a solution to be 'optimal' unless it is specifically excluded. We thus add a constraint that for any $c_1, c_2 \in C$, the centroids of the objects in c_1 (i.e. the set $\{M_o : o \in c_1\}$) are linearly separable from the centroids of the objects in c_2.

The clustering problem is then defined to partition O into a set of clusters C such that: (1) $d_c^* \leq d^{(\text{gr})}$ for all $c \in C$; (2) any $c_1, c_2 \in C$ have a line which separates the centers of mass of the objects they contain; (3) each cluster has small pairwise distances between its objects; and (4) there are few clusters overall. It is solved using a modified hierarchical clustering algorithm.

C. The Pushing and Grasping Problems

After the clusters have been defined, the next step is to push each cluster closer together to make multi-object grasping more effective (see Fig. 1 for an illustration). We assume that (due to the separating line constraint in the clustering problem) each cluster can be treated as a separate instance of the pushing problem.

Finally, once the pushing is done, we move the objects to the bin via multi-object grasping. As in [6], the objective is to move the objects to the bin with as few grasps as possible (without pushing, as that has already been done).

---

[1]It is possible that even if $d_c^* \leq d^{(\text{gr})}$ the cluster does not have a stable grasp. For example, a cluster of three identical equilateral triangles cannot be stably grasped at all.
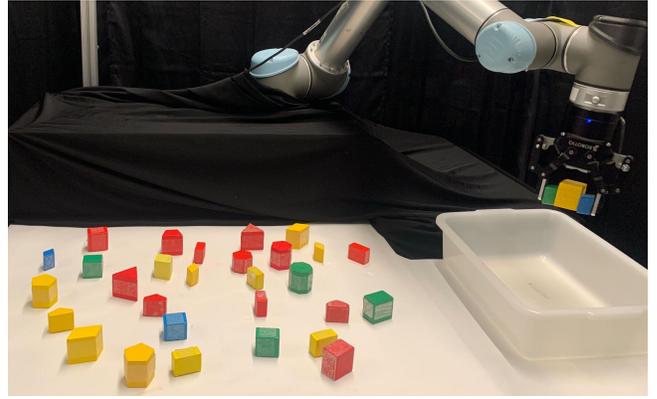

Figure 2: As illustrated in Fig. 1, Push-MOG uses "fork pushes" to consolidate objects for multi-object grasping as shown on the right.

Algorithm 1: Push-MOG

---

1 do
2 | $I \leftarrow$ Image of the workspace
3 | Parse I into a 2D object map of all objects O
4 | Partition O into clusters C (Alg. 2, see IV-A)
5 | Get a cluster c from C at random
6 | Plan the pushes, P (Alg. 3, see IV-B)
7 | **for** *push* $p \in P$ **do**
8 | | Predict post-push location of pushed object
9 | Run MOG-Net (IV-C), using predicted post-push locations, to identify grasp g
10 | Execute g to remove c
11 while *objects remain on the workspace*;

IV. Push-MOG Algorithm

We present the Push-MOG algorithm in Alg. 1. After perceiving an initial scene, it computes a cluster of objects. We plan and execute a series of pushing actions to bring together the clustered objects so that they are touching each other. Finally, we query the MOG-Net algorithm [6] to transport that cluster through Multi-Object grasps. Below, we outline how the Push-MOG algorithm plans and executes these steps.

A. Clustering Details (Alg 2)

As described in Section III-B, the goal of this step is to cluster objects into graspable subsets. In order to accomplish this we compute the centroids of each object and use a bottom-up hierarchical clustering. We find the smallest bounding box that encloses each object and cluster them based on their proximity to each other. Once these clusters are formed, we check that the sum of the widths of all the objects in the cluster does not exceed the width of the gripper. If we encounter a non-valid cluster c (i.e. such that $d_c^* > d^{(\text{gr})}$), then we split c into two clusters c_1, c_2 using a separating line (objects are divided based on their centroids), minimizing the difference between the cluster grasp diameters $|d_{c_1}^* - d_{c_2}^*|$.

Algorithm 2: Clustering Algorithm

Input	: O: List of objects
Output	: C: List of clusters

1 Compute centroids M_o for all $o \in O$

2 Hierarchical clustering on $\{M_o : o \in O\}$ to generate clusters C

3 Check the validity of each cluster $c \in C$ based on max gripper width

4 for *non-valid cluster $c \in C$* **do**

5   Minimize weight W over θ using the following:
         for $\theta \in [-\frac{\pi}{4}, \frac{\pi}{4}]$ **do**

6 $v = \langle \cos\theta, \sin\theta \rangle$

7   Split the objects into two clusters c_1, c_2 on an infinite line defined by v, through the centroid of c

8   For this θ, calculate weight difference $W = |\sum_{o \in c_1} d_o^* - \sum_{o \in c_2} d_o^*|$

9   Split the cluster on this θ

B. Push Planning (Alg 3)

Push planning considers how to push objects in a cluster closer together in order to facilitate grasping. This can be broken into two steps: (i) determining the desired locations of the cluster's objects; (ii) computing the parameters of the pushes to get them there. For object o, we denote the desired location of its centroid as M_o', which is chosen both to make pushing easier and to position the objects for grasping. For simplicity, we only specify a desired location for the centroid (and not a desired orientation). All pushes are executed in one motion in a straight line; the direction and distance that o is pushed is given by the vector $M_o' - M_o$ (to move its center of mass from M_o to M_o').

For executing a push with a parallel-jaw gripper, we propose the technique of *fork pushing*: we angle the gripper so the line between the jaws is perpendicular to the desired push direction. This allows the gripper to push both along an edge or around a vertex of an object (see Fig. 3), whereas a flat-edged pusher could not push against a vertex without the object undergoing a major rotation, deviating from the desired path.

C. MOG-Net Integration

Following the pushing step, we use MOG-Net [6] on the clusters to clear the workspace. To improve efficiency, we use the simulated coordinates of the push action to plan grasps instead of taking an image of the workspace again.

However, this process is not fully robust, because pushes may cause other blocks in the way to deviate from their estimated location, leading to a faulty grasp.

To remedy this, whenever a trip is performed (thus moving the gripper out of the camera's view of the workspace), a new image of the workspace is taken, allowing these errors to be corrected; the algorithm also uses this image to re-plan the clustering (IV-A) and pushing (IV-B) on the remaining blocks (which may have been pushed aside or missed during an earlier grasp).

Algorithm 3: Push Planning Algorithm

Input	: C: List of Clusters
Output	: v_{start}, v_{end}: Start and end push points for each object

1 for *valid cluster $c \in C$* **do**

2 $M_c = \frac{\sum_{\text{object } o \in c} M_o}{|c|}$

3 $m = \arg\min_{o \in c} \|M_o - M_c\|$, the central object in the cluster

4   **for** *object $o \in c \backslash m$* **do**

5 $v = M_o - M_m$, v_l line with endpoints M_o, M_m

6   Calculate center of furthest edge e_o and corner d_o of object o from m, which intersects with v_l

7   Calculate closest edge e_m on o to o_m on v_l

8   Scale $v \to v_p$ to capture the length of the push vector $\|v\|$

9   Ensure no collisions by shortening v_p, such that $v_p + o$ does not collide with o_m
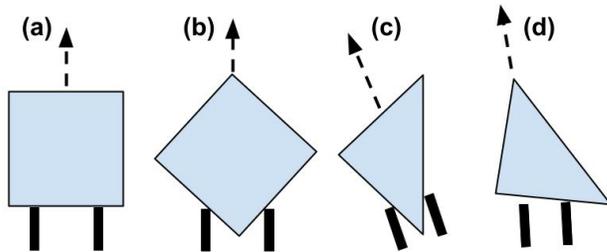

Figure 3: Examples of fork pushing with a parallel-jaw gripper, both against an edge ((a) and (d)) and around a vertex ((b) and (c)) noticing that the direction the object will move to is roughly perpendicular to the parallel-jaw gripper.

V. EXPERIMENTS AND RESULTS

To evaluate the performance of the policies developed in the paper, we run experiments in a real workspace.

A. Experimental Setup

An example input is shown in Fig. 4. We use a Universal Robotics (UR) 5 Robot with a Robotiq 2F-85 gripper mounted to its wrist. We perform experiments on 34 convex objects of various sizes, ranging from 3-sided to 8-sided. All experiments assume a random placement of the objects in the workspace, and one such configuration is shown in Fig. 4. To generate a random placement, we uniformly sample the workspace and create circles that encompass the max radius of any object we are using. Before creating each circle we ensure that they do not overlap, and continue sampling the object set without replacement until all objects are placed. Then a human sets up the configuration in real, mirroring the randomly generated image. We generated 5 random scenes, on which we will perform all experiments. We add friction on the graspable faces of the objects by wrapping them in frictional tape.

To perceive the environment, we use an Intel RealSense L515 Camera, which outputs RGB-D images, mounted di-

TABLE I: Physical decluttering experimental results for random scenes, comparing baselines **Frictional SOG, MOG-Net**, and the method **Push-MOG**. Experiments were conducted with a Robotiq 2F-85 parallel-jaw gripper mounted on a UR5 robot arm, on 5 random scenes of 34 objects, with the goal placed directly adjacent (see Figs 2 and 4).

Methods	Objects per Trip (OpT)
Frictional SOG	0.993 ± 0.014
MOG-Net	0.986 ± 0.038
Push-MOG	1.339 ± 0.145

rectly above the workspace. To calibrate the Camera-Robot transform, use an ArUco marker [41] which is manually calibrated with an accuracy of $\sim 1cm$.

The pushing action consists of 5 repeated steps:

1) Open the jaws to 30 percent width
2) Move the gripper above the desired push location
3) Move down to the object's Z-height
4) Move the gripper and push it to its planned location
5) Move the gripper above its planned location

The Multi-Object Grasping action is executed with the MOG-Net algorithm [6]. Here we repeat the following:

1) Plan a grasp for each cluster
2) Move the gripper above the desired grasp location
3) Open the jaws fully
4) Move down to the work surface
5) Close the jaws to complete the grasp
6) Move the gripper outside the workspace and open the jaws fully to drop the objects into the basket

B. Baselines

We compare Push-MOG against two baselines: Frictional SOG, which uses single-object grasps, and MOG-Net [6], which uses multi-object grasps but does not use additional actions to help group the objects together. See Section I for detailed information on baselines.

C. Results

Results are given in Table I. We find that randomly-placed objects are not generally well-positioned for multi-object grasps, as OpT is almost identical between MOG-Net and Frictional SOG, thus requiring the use of additional actions such as pushes to assist MOG. Indeed, the similarity in the performance of Frictional SOG and MOG-Net suggests that there are few if any, 'naturally occurring' multi-object grasps in a typical scene of randomly-scattered objects. Push-MOG successfully uses the pushing action to generate graspable object clusters, increasing OpT by roughly 34%. This comparatively modest increase in OpT (as compared to the experimental results on the original MOG-Net [6]) is because objects that are stable under pushing tend to have larger minimum grasp diameters and the gripper we used has a relatively small max opening width, so it can grasp at most 2 or 3 objects at a time.
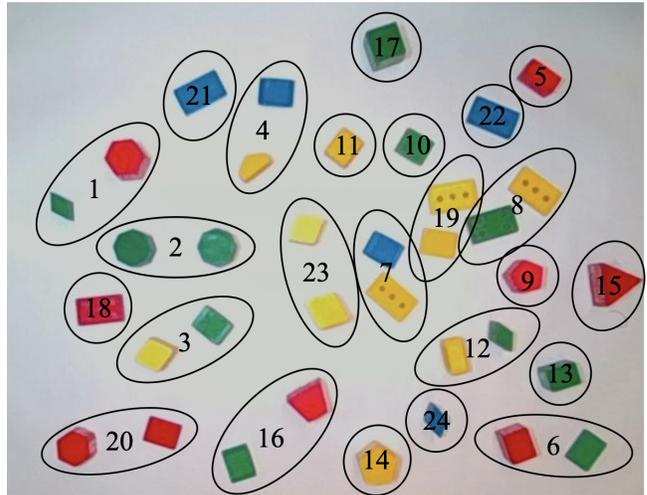

Figure 4: This figure shows the clustering formed by Push-MOG of an initial scene. The numbers on each cluster identify the order that the algorithm will perform the pushing and grasping actions.

VI. LIMITATIONS AND FUTURE WORK

This work proposes Push-MOG, a novel algorithm that consolidates polygonal objects into optimal clusters which increase the number of objects per grasp. This work has the following limitations: 1) pushing can cause misalignments where grasps fail 2) Push-MOG does not always avoid collisions 3) total time is slow, leading to fewer picks per hour, even though Push-MOG gets more objects per trip. In future work, we can incorporate *vertical stacking* to create larger multi-object clusters.

REFERENCES

[1] J. Ichnowski, M. Danielczuk, J. Xu, V. Satish, and K. Goldberg, "Gomp: Grasp-optimized motion planning for bin picking," in *ICRA*, 2020.
[2] J. Ichnowski, Y. Avigal, V. Satish, and K. Goldberg, "Deep learning can accelerate grasp-optimized motion planning," *Science Robotics*, vol. 5, no. 48, 2020.
[3] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Fastrack: A modular framework for fast and guaranteed safe motion planning," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 1517–1522.
[4] J. Ichnowski, Y. Avigal, Y. Liu, and K. Goldberg, "Gomp-fit: Grasp-optimized motion planning for fast inertial transport," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 5255–5261.
[5] W. C. Agboh, J. Ichnowski, K. Goldberg, and M. R. Dogar, "Multi-object grasping in the plane," 2022. [Online]. Available: https://arxiv.org/abs/2206.00229
[6] W. C. Agboh, S. Sharma, K. Srinivas, M. Parulekar, G. Datta, T. Qiu, J. Ichnowski, E. Solowjow, M. Dogar, and K. Goldberg, "Learning to efficiently plan robust frictional multi-object grasps," 2022. [Online]. Available: https://arxiv.org/abs/2210.07420

[7] E. Arruda, M. J. Mathew, M. Kopicki, M. N. Mistry, M. Azad, and J. L. Wyatt, "Uncertainty averse pushing with model predictive path integral control," *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pp. 497–502, 2017.

[8] F. R. Hogan and A. Rodriguez, *Feedback Control of the Pusher-Slider System: A Story of Hybrid and Underactuated Contact Dynamics*. Cham: Springer International Publishing, 2020, pp. 800–815. [Online]. Available: https://doi.org/10.1007/978-3-030-43089-4_51

[9] K. Harada and M. Kaneko, "Enveloping grasp for multiple objects," in *ICRA*, 1998.

[10] K. Harada, M. Kaneko, and T. Tsujii, "Rolling-based manipulation for multiple objects," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 5, pp. 457–468, 2000.

[11] T. Yamada, T. Ooba, T. Yamamoto, N. Mimura, and Y. Funahashi, "Grasp stability analysis of two objects in two dimensions," in *ICRA*, 2005.

[12] T. Yamada and H. Yamamoto, "Static grasp stability analysis of multiple spatial objects," *Journal of Control Science and Engineering*, vol. 3, 2015.

[13] T. Chen, A. Shenoy, A. Kolinko, S. Shah, and Y. Sun, "Multi-object grasping - estimating the number of objects in a robotic grasp," in *IROS*, 2021.

[14] A. Shenoy, T. Chen, and Y. Sun, "Multi-object grasping - efficient robotic picking and transferring policy for batch picking," *IEEE IROS*, 2022.

[15] P. Jiang, J. Oaki, Y. Ishihara, and J. Ooga, "Multiple-object grasping using a multiple-suction-cup vacuum gripper in cluttered scenes," 2023.

[16] V. P. Nguyen and W. T. Chow, "Wiring-claw gripper for soft-stable picking up multiple objects," *IEEE Robotics and Automation Letters*, vol. 8, no. 7, pp. 3972–3979, 2023.

[17] M. Dogar and S. S. Srinivasa, "A framework for push-grasping in clutter," in *RSS*, 2011.

[18] M. Danielczuk, J. Mahler, C. Correa, and K. Goldberg, "Linear Push Policies to Increase Grasp Access for Robot Bin Picking," in *CASE*, 2018.

[19] H. Huang, M. Dominguez-Kuhne, V. Satish, M. Danielczuk, K. Sanders, J. Ichnowski, A. Lee, A. Angelova, V. Vanhoucke, and K. Goldberg, "Mechanical search on shelves using lax-ray: Lateral access x-ray," in *IEEE IROS*, 2021.

[20] W. Agboh and M. Dogar, "Real-time online re-planning for grasping under clutter and uncertainty," in *IEEE Humanoids*, 2018.

[21] T. Sakamoto, W. Wan, T. Nishi, and K. Harada, "Efficient picking by considering simultaneous two-object grasping," 2021.

[22] E. Huang, Z. Jia, and M. T. Mason, "Large-scale multi-object rearrangement," in *ICRA*, 2019.

[23] W. Bejjani, W. Agboh, M. Dogar, and M. Leonetti, "Occlusion-aware search for object retrieval in clutter," in *IEEE IROS*, 2021.

[24] M. Hasan, M. Warburton, W. Agboh, M. Dogar, M. Leonetti, H. Wang, F. Mushtaq, M. Mon-Williams, and A. Cohn, "Human-like planning for reaching in cluttered environments," in *ICRA*, 2020.

[25] M. Mason, *Mechanics of Robotic Manipulation*, ser. Intelligent Robotics and Autonomous Agents series. MIT Press, 2001. [Online]. Available: https://books.google.com/books?id=Ngdeu3go014C

[26] W. C. Agboh, D. Ruprecht, and M. R. Dogar, "Combining coarse and fine physics for manipulation using parallel-in-time integration," *ISRR*, 2019.

[27] J. E. King, M. Cognetti, and S. S. Srinivasa, "Rearrangement planning using object-centric and robot-centric action spaces," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 3940–3947.

[28] H. Song, J. A. Haustein, W. Yuan, K. Hang, M. Y. Wang, D. Kragic, and J. A. Stork, "Multi-object rearrangement with monte carlo tree search: A case study on planar nonprehensile sorting," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 9433–9440.

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

[30] W. C. Agboh and M. R. Dogar, "Pushing fast and slow: Task-adaptive planning for non-prehensile manipulation under uncertainty," in *WAFR*, 2018.

[31] W. Agboh and M. Dogar, "Robust physics-based manipulation by interleaving open and closed-loop execution," *CoRR*, vol. abs/2105.08325, 2021.

[32] W. Yuan, J. A. Stork, D. Kragic, M. Y. Wang, and K. Hang, "Rearrangement with nonprehensile manipulation using deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 270–277.

[33] K. Y. Goldberg, "Orienting polygonal parts without sensors," *Algorithmica*, vol. 10, pp. 201–225, 1993.

[34] M. Erdmann and M. Mason, "An exploration of sensorless manipulation," *IEEE Journal on Robotics and Automation*, vol. 4, no. 4, pp. 369–379, 1988.

[35] A. M. Johnson, J. King, and S. Srinivasa, "Convergent planning," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1044–1051, 2016.

[36] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. [Online]. Available: https://faculty.marshall.usc.edu/gareth-james/ISL/

[37] B. Mirkin and B. Mirkin, *Clustering for Data Mining: A Data Recovery Approach*, 1st ed. Chapman and Hall/CRC, 2005.

[38] D. Müllner, "Modern hierarchical, agglomerative clustering algorithms," 2011. [Online]. Available: https://arxiv.org/abs/1109.2378

[39] Z. Bar-Joseph, D. K. Gifford, and T. S. Jaakkola, "Fast optimal leaf ordering for hierarchical clustering ," pp. S22–S29, 06 2001. [Online]. Available: https://doi.org/10.1093/bioinformatics/17.suppl_1.S22

[40] T.-B.-H. Dao, K.-C. Duong, and C. Vrain, "Constrained clustering by constraint programming," pp. 70–94, 2017, combining Constraint Solving with Mining and Learning. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370215000806

[41] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," pp. 2280–2292, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320314000235