Automating Planar Object Singulation by Linear Pushing with Single-point and Multi-point Contacts

Zisu Dong[1], Sanjay Krishnan[1], Sona Dolasia[1], Ashwin Balakrishna[1], Michael Danielczuk[1] and Ken Goldberg[1,2]

***Abstract*— Singulation is useful for manufacturing, logistics, and service applications; we consider the problem in a planar setting. We propose a novel $O(n(n+v))$ linear push policy (n denotes the number of objects, v denotes the maximum number of vertices per object), ClusterPush, that can be efficiently computed using clustering. To evaluate the policy, we define *singulation distance* as the average pairwise distance of polygon centroids given random arrangements of 2D polygonal objects on a surface, and seek pushing policies that can maximize singulation distance. When compared with a brute force evaluation of all candidate pushes in Box2D simulator using 50,000 pushing scenarios, ClusterPush achieves 70% of the singulation distance achieved using brute force and is 2000x faster. ClusterPush also improves on previous pushing policies and can be used for multi-point pushes with two-point and edge (infinite-point) contacts. Compared with pushes with single-point contacts using ClusterPush, pushes with two-point and edge contacts improve singulation by 7% and 13% respectively. In physical experiments conducted with an ABB YuMi robot on 40 sets of 3-7 blocks, ClusterPush increases singulation distance by 15-30%, outperforming the next best policy by 24% on average. Data and code are available at `https://github.com/Jekyll1021/MultiPointPushing`.**

I. Introduction

In manufacturing, logistics, and service applications, objects of interest may not be directly graspable due to reachability constraints introduced by the presence of other movable objects. For example, consider a robot trying to retrieve a pen from a messy desk. It may first have to move books and papers out of the way before there is a viable path to access and grasp the pen. Reasoning about object motion caused by incidental object contacts is a key challenge and has been studied extensively in prior work [1–5]. Recent work suggests that it is additionally valuable to apply singulation motions [6–8] or deliberate pushes to separate objects before grasping. Building on prior work, we study the problem of *planar singulation*, where, given a collection of 2D convex polygon objects on a planar surface, the objective is to maximize the average pairwise distance between polygons (singulation distance). Although previous work suggests that graspability is not always correlated with object separation [6], inaccurate perception may appear due to lack of object separation [9]. Figure 1 illustrates an initial configuration, a linear push, and the outcome.

[1]Department of Electrical Engineering and Computer Science
[2]Department of Industrial Engineering and Operations Research
[1,2]The AUTOLAB at UC Berkeley; Berkeley, CA 94720, USA

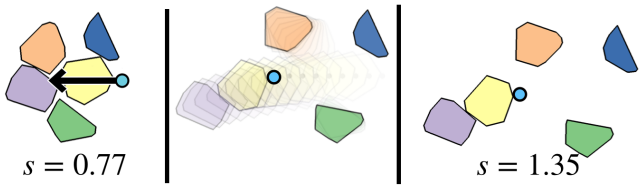

Fig. 1: We define singulation distance as the average pairwise distance between the center of mass of all objects. Illustrated above left to right: a single-point push that increases singulation distance by 75%.

Directly planning in a physics simulator can be computationally expensive. Even in a simplified Box2D [10] simulator, planning for a 3-body configuration takes 16 seconds on average on a 2.9GHz Intel Core I5-5287U processor, and this runtime scales linearly with the number of objects. This drawback motivates our study of geometric heuristics for singulation that can be computed several orders of magnitude faster than full dynamic simulation. On randomly generated planar singulation tasks, we compare 8 singulation heuristics against brute-force simulation. We use a Box2D simulation environment with simplified contact friction to ensure deterministic solutions. We ran 50,000 simulated trials of cluttered objects under 52 different configurations of object numbers and shapes for each of the heuristics studied. Based on the results of this study, we observed two key components contribute to empirically good pushes: moving the center object of a set without slipping for 3-body cases and moving more than one object away from a large object cluster for cases with more objects. These observations motivate the development of ClusterPush, an $O(n(n+v))$ pushing heuristic, where n denotes the number of objects and v denotes the maximum number of vertices per object.

ClusterPush plans a push by first finding the object closest to the center of the set of objects, and then uses geometric clustering to plan a push direction. It plans to push up to two objects away from the other objects without slipping. ClusterPush achieves 70% of the singulation distance found through brute-force evaluation of candidate pushes in 8ms, almost 2000x faster compared to 16s of brute-force evaluation.

We also consider pushes with a contact line (edge pushes) and pushes with two contact points (two-point pushes) to maximize contact. In complex configurations with 10 or more objects, edge pushing and two-point pushing can improve pushes planned by ClusterPush by 13% and 7% relative to single-point pushes respectively.

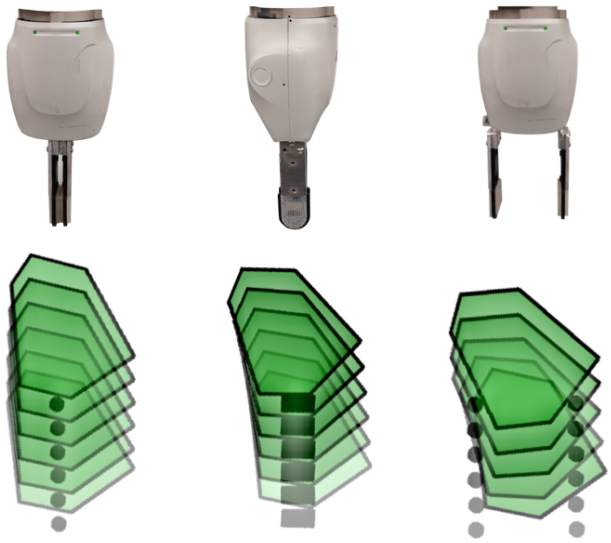

Fig. 2: (TOP) Single-point, edge and two-point pushes of an ABB Yumi robot and (BOTTOM) single-point, edge and two-point pushes in simulation.

We also implemented ClusterPush on a physical robot system (ABB YuMi), where edge pushes and two-point pushes improved the increase in singulation distance by 18.4% and 28.4%, respectively over single-point pushes.

This paper makes the following contributions:

1) A Box2D simulation environment for singulation with experimental data that includes over 50,000 planar scenarios and 8 singulation policies.
2) A *singulation distance* metric and 5 novel pushing policies.
3) A $O(n(n+v))$ policy, ClusterPush, which leverages sticking pushes, the collision between objects, and a disk-based approximation.
4) Heuristics for edge pushes and two-point pushes.

II. Related Work

Pushing is a core manipulation primitive and the mechanics of planar pushing has been studied in various forms for the last 30 years [11–13]. The seminal result is Mason's rule, which specifies the push result as a function of the right boundary of the friction cone, the left boundary of the friction cone, and the velocity vector of the push. When two of the three lines are left of the Center of mass (CoM), the object rotates clockwise, and vice versa. The object purely translates when the CoM is within the friction cone and the push direction is through the CoM.

The principle behind ClusterPush is that push directions that maximize translation tend to be effective at singulation because such directions can maximally displace target objects. Note that pure translational motions are actually quite hard to achieve as they require careful alignment of a gripper and the CoM of an object. Thus, object-to-object contacts are almost always rotationally dominant. Finding a single object and effectively pushing it with a strong translational motion naturally rotates other objects away from collision. This intuition is a heuristic and has limitations in cases where collisions are inevitable with an increasing number of objects, or where rotational motion is inevitable because push objects slip away.

This principle is supported by results in prior singulation work [6, 9, 14]. Hermans et al. proposed a guided linear push policy [14] which identifies the two closest objects and pushes one of them along the boundary between the two objects. All pushes are through the CoMs. These results were reproduced by [6], which additionally showed that the boundary shear approach improves downstream grasp quality metrics in addition to object separation.

Some recent advances in planar pushing include a high-fidelity single-object planar pushing dataset [15, 16], a heuristic-guided search for surface decluttering [17] and an end-to-end push proposal network for singulation [18]. Both latter works consider objects in clutter, and Cogsun et al. showed that physical simulation can effectively capture interactions in object contacts [17].

Researchers have investigated data-driven approaches for planar pushing and surface decluttering. The existing physical simulators rely on analytical models to coarsely approximate contact outcomes [10, 19–21]. The viability of large-scale datasets enables accurate learning for planar pushing to a precise pose [22]. However, most prior work focuses on single-object planar pushing. Recently, Agboh and Dogar investigated the effect of pushing velocity [23]; Xie and Chakraborty proposed a principled dynamic model for pushing [24]; Ajay et al. designed a physics engine with an RNN-based residual model to resemble the stochastic behavior of real-life planar-pushing dataset benchmarks [25]; Jiang et al. extended the data-augmentation technique to rigid body simulation [26]. We hope to explore these directions in future work.

Another very relevant line of work [1–5, 27] is grasping or manipulation in clutter. We differentiate this work from the problem of singulation as it largely studies the effects of *specific* object contacts in the process of generating a trajectory to a goal position. In contrast, we focus on singulation (maximizing the average pairwise distance) itself.

III. Problem Statement

Let o_i denote a 2D oriented polygon in the plane, $c_i \in \mathbb{R}^2$ denote its center of mass, and $\theta_i \in [0, 2\pi)$ denote its orientation. Let $\mathcal{O} = \{o_1, ..., o_n\}$ be a set of such polygons and $\mathcal{A} = \{(o_1, c_1, \theta_1), ..., (o_n, c_n, \theta_n)\}$ be an arrangement of the set. We assume that all objects are convex with known uniform density, geometry, and pose.

A push $p \in \mathbb{R}^2 \times \mathbb{R}^2$ is a planar linear motion defined by a directed line-segment $(x_0, y_0) \rightarrow (x_1, y_1)$. After applying a push to the arrangement of objects, the arrangement changes due to rotations and translations of each polygon:

$$\mathcal{A}' = p(\mathcal{A}) = \{(o_1, c_1', \theta_1'), ..., (o_n, c_n', \theta_n')\}$$

Definition 1 (Singulation Distance): We define the *singulation distance* as the average log pairwise distance between the CoM of all objects. We take the log distance to discourage dense clutters:

$$s(\mathcal{A}) = \frac{\sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1} \ln(||c_i - c_j||)_2}{\binom{n}{2}}$$

The objective is to find the linear push that maximizes the increase in singulation distance:

$$p^* = \max_p \frac{s(p(\mathcal{A})) - s(\mathcal{A})}{s(\mathcal{A})}$$

A. Assumptions

We make the following assumptions:

1) The pushing surface is planar with a homogenous friction coefficient.
2) Object-to-object and gripper-to-object contacts have the same friction coefficient with equal static and dynamic coefficients.[1]
3) Objects have the same density.
4) A quasi-static model of push dynamics.

We implement this model in a Box2D simulator with the following parameters: a global friction coefficient $\mu \sim \mathcal{N}(0.5, 0.1)$ for both gripper-object contact and object-object contact, and a global density $\rho \sim \mathcal{N}(1.0, 0.2)$. Line contacts are calibrated in the simulator based on the size of the grippers relative to objects on the ABB YuMi robot. We check the reachability of a push by ensuring free space for the gripper at the push starting point.

B. Types of Pushes

Assuming constant push velocity, we investigate pushing under three contact modes: single-point, edge, and two-point (Figure 2). For point pushing, a line segment defines the trajectory of the contact point while for two-point pushing and edge pushing, a line segment defines the trajectory of the midpoint of the two contact points or contact line.

Single-Point Push: Let $u_{sp} = (P, Q) \in \mathbb{R}^2 \times \mathbb{R}^2$ be a point linear pushing action in 2D space defined by start point $P \in \mathbb{R}^2$ and end point $Q \in \mathbb{R}^2$, such that the pushing jaw, starting at point P, moves a fixed distance along line segment PQ.

Two-Point Push: Let $u_{tp} = (P, Q) \in \mathbb{R}^2 \times \mathbb{R}^2$ be a two-point linear pushing action in 2D space defined by start point $P \in \mathbb{R}^2$ and end point $Q \in \mathbb{R}^2$. P is the midpoint of the line segment connecting the two gripper jaws f_1, f_2 with a predefined length l such that $\overline{PQ} \perp \overline{f_1 f_2}$. The two jaws move a fixed distance along $\overrightarrow{PQ}$ at the same rate.

Edge Push: Let $u_e = (P, Q) \in \mathbb{R}^2 \times \mathbb{R}^2$ be an edge linear pushing action in 2D space defined by start point $P \in \mathbb{R}^2$ and end point $Q \in \mathbb{R}^2$. P is the midpoint of the pushing edge with a predefined length l such that the pushing edge is perpendicular to $\overline{PQ}$. The pushing edge moves a fixed distance along $\overrightarrow{PQ}$.

[1]Box2D does not use a Coulomb model for friction. It calculates the contact dynamics with an idealized infinite-friction edge contact, then damps the resulting impulse by the amount of friction between the two bodies. This model removes indeterminate solutions that might arise due to discrete contact modes.

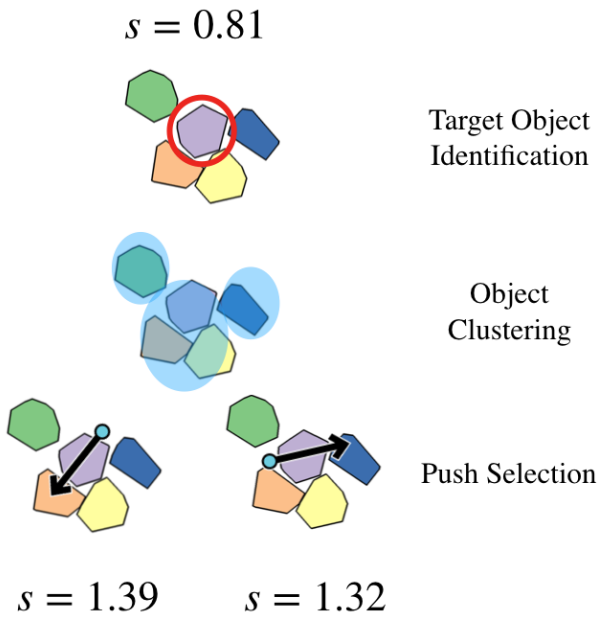

Fig. 3: The three steps of ClusterPush. The first step identifies a center object to push, the second step groups objects into clusters, and the third step chooses between a direction to push two objects (Bottom left) and a direction to separate the push object (Bottom right).

IV. ClusterPush

Based on the intuition given earlier, we focus on sticking pushes with a single target object. Although ClusterPush is motivated by 3-object problems, we find that ClusterPush also performs well on larger problems. We observe that, in object clusters with more than three objects, it is often desirable to separate more than one object from the large cluster. To address this limitation, we plan a push pointing towards another object's center of mass, creating a sticking contact for moving both objects. There are a few important components in the policy: (1) identifying a target object, (2) grouping objects based on geometric proximity, and (3) identifying a sticking push that pushes an object away from the rest of a cluster. Figure 3 steps through one push planned by ClusterPush.

Step 1. Target Object Identification

For all pairs of objects, we calculate the Euclidean distance between their CoMs:

$$\mathcal{D}(o_i, o_j) = \|c_i - c_j\|$$

Each object is then assigned a connectivity score, which is the sum of its distances to all other objects:

$$\mathcal{C}(o_i) = \sum^{N} \mathcal{D}(o_i, o_j)$$

The **push object** o^* is the object is the object with the lowest connectivity score:

$$o^* = \underset{i}{\operatorname{argmin}}\ \mathcal{C}(o_i)$$

Thus, o^* is the object that if removed (without affecting the others), would most increase the sum of pairwise distances.

Algorithm 1: Find Pushing Direction

find_push_direction($\mathcal{G}, o^*$);
Input : a list of clusters $\mathcal{G} = \{g_1, ..., g_m\}$, a push object o^*
Output: push trajectory defined by start point and end point $u_s = (p, q) \in \mathbb{R}^2 \times \mathbb{R}^2$
g^* = the cluster g_i contains o^*;
if g^* has the most objects **and** $m \neq 1$ **then**
    $furthest$ = Furthest pushing direction to the centroid of each cluster $g \in G \setminus \{g*\}$;
    for o in g^* **do**
        find $vector$ connecting c_{o^*} and c_o that is closest to $furthest$;
    end
    return (p, q) along the direction of $vector$;
else
    for 16 directions v spans 0 to 2π **do**
        for $cluster \in G \setminus \{g*\}$ **do**
            find the centroid of the *cluster*;
            compute the distance from the centroid of the *cluster* to v;
        end
    end
    find v with the minimum sum of the distance;
    return (p, q) along the direction of v and across c_{o^*};
end

Step 2. Object Clustering

We model an object o_i with its smallest bounding circle (c_i, r_i), where c_i denotes the object's CoM and r_i denotes the radius of the circle.

Using the CoM of the designated push object o^* found by *Step 1* as the center of a cluster g_j, we incrementally grow the cluster to encompass the other neighboring objects until no other object satisfies the criteria:

$$||g_j - c_i|| < r_j + r_i$$

All objects within that cluster are removed from consideration, and the clustering algorithm recurses. To seed the next cluster, we find the object with the maximum sum of CoM distances to the other cluster centers:

$$\underset{i}{\operatorname{argmax}} \sum_{j=1}^{m} ||g_j - c_i||$$

The result is a list of clusters, one of which is centered on the target push object.

Step 3. Push Direction

A linear push can make sticking contact with more than one object: one via pushing contact, the others via an indirect push from the push object, as shown in Figure 1. However, ClusterPush only plans sticking contact for two objects: a push that connects the CoM of both objects.

In scenarios where more than three objects are densely cluttered, it is often desirable to push two objects away instead of one. In ClusterPush, if there are at least two clusters and the cluster centered around the push object contains the most objects, a two-object push is planned. Otherwise, a one-object push is planned. This step is detailed in Algorithm 1.

The selecting criteria are: if the cluster centered around the push object contains the most objects, and there is more than one cluster, the policy plans to move two objects; otherwise, it plans to move only the push object. The intuition behind this criteria is that if the cluster with the most objects does not contain the push object, the push object is considered sufficiently separated from all other objects.

Case 1. Two-object Push: If the cluster has more than 3 objects, we plan a two-object push by choosing a direction that will create a sticking contact between the two objects, and move them in the furthest direction from the other objects.

To create a sticking contact, we only consider pushing directions connecting the CoM of the push object and the CoM of another object in the same group as the push object. For all the objects in the group, we find the object that creates the push direction furthest away from the CoMs of other objects in the cluster, formulated as:

$$o_s = \arg\min_i \sum_{o_j \neq o_i, o_j \neq o*}^{j} proj(o_i, v)$$

where $proj$ denotes a standard orthogonal projection. Therefore, the selected vector would move objects away from the rest to the greatest extent.

Case 2. Single-object Push: If the cluster has no more than 3 objects, we seek to separate the push object from the rest without slipping. We model each object with its bounding circle and find the push direction that maximizes the sum of distances from disk borders of the other objects to pushing trajectory.

V. Experimental Setup

We compare ClusterPush to eight policy baselines, four of which are new.

A. Policy Inputs

All of the evaluated policies require full state information (geometries, positions, and orientations of all polygons). The state information is retrieved from Box2D and collision checks are handled within Box2D. We create cluttered sets with 3 to 15 randomly generated objects and initialize the simulation by sampling over $\mathcal{N}(0.5, 0.1)$ for friction coefficients and $\mathcal{N}(1.0, 0.2)$ for object densities.

To evaluate the difficulty posed by specific object sets, we measure the "eccentricity" [28] of objects, or how similar the objects are to a disk. Less-eccentric objects were empirically easier to singulate. Eccentricity is quantified by the ratio of an object's area and the area of its minimum bounding circle, defined as $O_c = (c, r)$, where $c \in \mathbb{R}^2 \times \mathbb{R}^2$ is the CoM of the object O and r is the maximum Euclidean distance between c and any of the object's vertices. Figure 4 illustrates 4 sample objects from each group. We experiment with 4 object sets of increasing eccentricity.

B. Baseline policies

We consider the following baseline policies:

Brute Force: This singulation policy performs a brute-force evaluation of a large set of candidate pushes via forward simulation. For each of the objects in the cluster, the policy

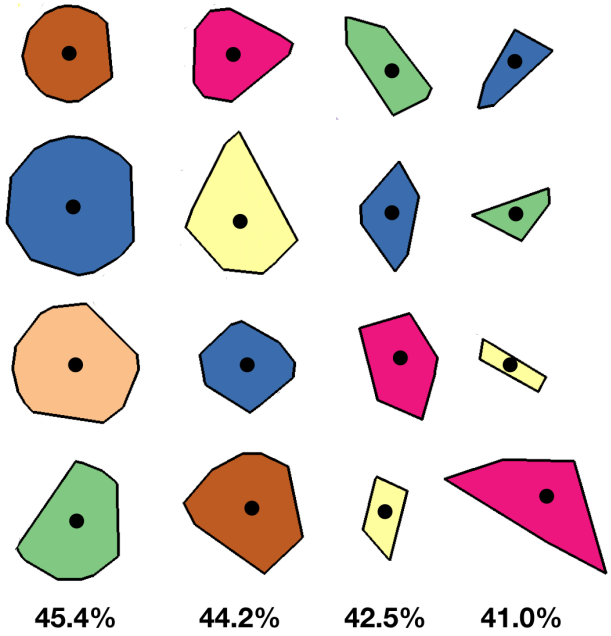

Fig. 4: From left to right: Group 0, 1, 2 and 3 polygon objects. Center of mass of each object is marked by a black dot. The numbers summarize the average increase in singulation distance using quasi-random policy normalized by brute force. Empirically, objects with higher eccentricity (see text) are hard to singulate.

searches over 16 uniformly sampled push directions and 16 uniform perturbations of the push vector passing through the object centroid.

Quasi-random: The Quasi-random policy generates a randomly sampled push vector through the centroid of a randomly sampled object. It does not optimize the push direction.

Boundary Shear Policy: The Boundary Shear policy is adapted from the policy introduced by Hermans *et al.* [14]. It aims to separate the two closest objects by pushing one in the direction of free space.

Cluster Diffusion Policy: The cluster diffusion policy is adapted from the policy introduced by Danielczuk *et al.* [6]. It aims to push one object away from the center of the clutter towards free space.

C. New Baselines

Observing some of the failure modes in these baselines, we proposed a few variations and improvements. These new variations incrementally build up to ClusterPush.

Center Object Removal Policy: This planner finds the object which, if removed from the clutter, results in maximum separation. It pushes the object away from the line segment connecting the center of mass of the two objects closest to the selected object.

Minimum Contact Range Policy: This policy pushes the selected object in the direction that minimizes its contact range.

Minimum Overlap Policy: This policy pushes the selected object in the direction that maximizes the distance between the CoMs of other objects and the push line segment.

Two Cluster Separation Policy: Since the Minimum Overlap Policy often performs poorly in dense object clusters, we propose a new policy that separates clusters into two clusters and pushes the selected object to the center of a cluster in which it is not contained.

VI. Simulation Results

A. Summary of Point Pushing Results

Figure 6 provides a summary of the performance of each policy as the number of objects increases. It shows that the three new policies (Center Object Removal, Minimum Contact Range, Minimum Overlap) aiming to move the center object while minimizing contact between objects perform well in scenarios with fewer and less-eccentric objects, but are inferior to other policies in denser clutter with more-eccentric objects. On the other hand, Cluster Diffusion, which seeks to push one object away from clutter, does not perform well in scenarios with fewer and less eccentric objects but performs better in denser clutter with more eccentric objects. Table 1 provides a summary of the runtime and performance of each policy. It shows that ClusterPush performs comparably to the other policies in easy scenarios and also remains robust in hard scenarios. In the easiest 3-body scenarios with less-eccentric objects (Groups 0 and 1), ClusterPush achieves 88.7% of the increase in separation distance compared to brute-force evaluation of candidate pushes, comparable to other policies (71.0% for Cluster Diffusion, the best previously proposed baseline, 88.7% for Minimum Overlap, the best new baseline). In hard 15-body scenarios with 15 objects of higher eccentricity (Groups 2 and 3), ClusterPush outperforms the Cluster Diffusion policy, the best policy among baselines, by 19.3% on average. However, we observe that all policies perform worse in hard scenarios (more objects and more-eccentric objects); ClusterPush only achieves 49.4% of the increase in separation distance in these scenarios compared to brute-force search. ClusterPush consistently achieves 70.2% of the increase in separation distance compared to brute-force search across all the scenarios, whereas Cluster Diffusion achieves 53.9% and Minimum Overlap achieves 64.8%.

Table 1 also compares the runtime (including simulation) for the pushing policies. We observe that ClusterPush achieves a 1905x speedup compared to brute-force search. Furthermore, none of the other policies except Quasi-random run significantly faster than ClusterPush.

B. Effect of Multi-point Contact Modes

We observe that most failure modes for single-point pushes occur when the push object slips away. To address this issue, we evaluate edge pushing with the length of the contact edge set to approximately the jaw width of the ABB YuMi robot, and two-point pushing using 60% of the radius of the push object's smallest bounding circle as the distance between jaws. We choose this heuristic because it performs empirically well. Table 2 provides a summary of the improvement in push performance for each of the

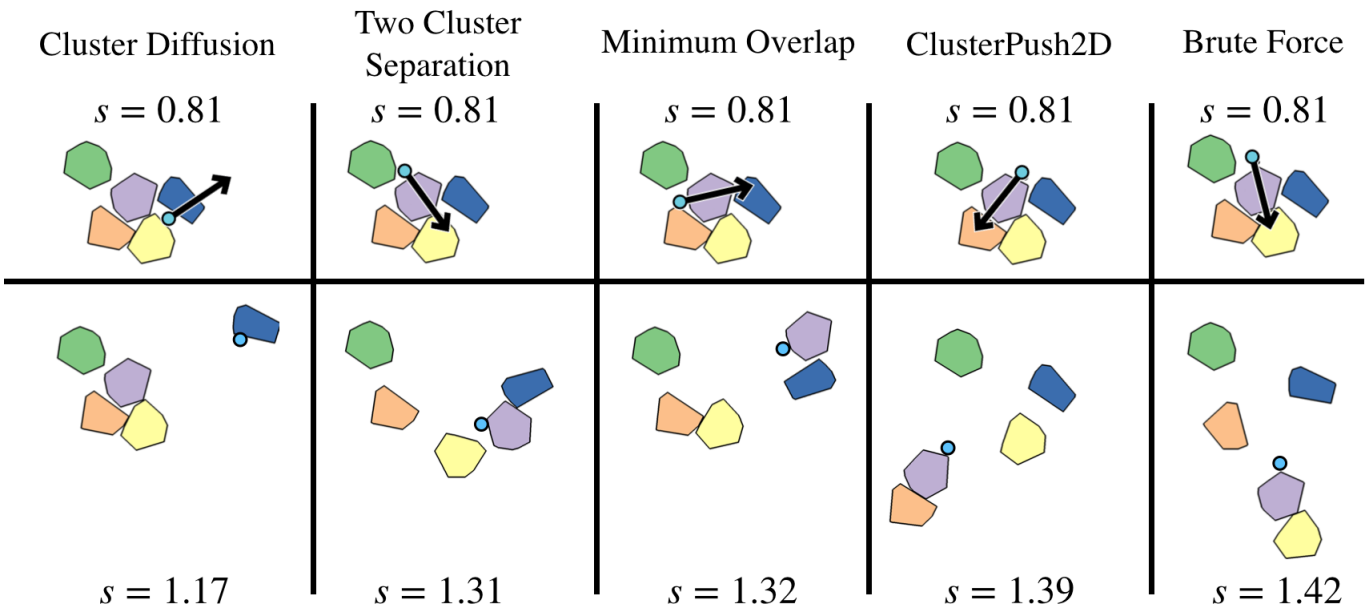

Fig. 5: Effects of proposed pushing policies. The top image is the planned push and the bottom image is the resulting state, each labeled with the state's singulation distance.

	Average Runtime (ms) on 2.9GHz Intel I5-5287U	Performance: Easy Scenarios (3-body Group 0/1)	Performance: Hard Scenarios (15-body Group 2/3)	Average Performance
Brute Force	16150.023	0.751 (100.0%)	0.078 (100.0%)	0.244 (100.0%)
Quasi-random	7.850	0.429 (57.2%)	0.018 (23.4%)	0.104 (42.8%)
Boundary Shear	12.495	0.510 (67.9%)	0.020 (26.2%)	0.124 (51.1%)
Cluster Diffusion	10.474	0.534 (71.0%)	0.032 (41.4%)	0.131 (53.9%)
Center Object Removal	**8.437**	0.647 (86.2%)	0.030 (38.3%)	0.160 (65.8%)
Min Contact Range	12.381	0.547 (72.8%)	0.033 (42.6%)	0.153 (62.9%)
Min Overlap	11.493	**0.666 (88.7%)**	0.026 (34.6%)	0.157 (64.8%)
Two Cluster Separation	10.609	0.512 (68.2%)	0.025 (31.5%)	0.149 (61.4%)
ClusterPush	**8.475**	**0.666 (88.7%)**	**0.038 (49.4%)**	**0.171 (70.2%)**

TABLE I: Results for single-point pushing in 52,000 scenarios. Performance is defined as the ratio of the change of singulation distance after and before a push (normalized to Brute Force = 100%). Average performance is computed across 52,000 scenarios of all object numbers and shapes. ClusterPush outperforms other policies while maintaining a fast runtime.

	Single-point Pushing	Edge Pushing	Two-point Pushing
Quasi-Random	0.027 (100.0%)	0.031 (114.8%)	0.029 (107.4%)
Boundary Shear	0.032 (100.0%)	0.036 (112.5%)	0.035 (109.4%)
Cluster Diffusion	0.043 (100.0%)	0.062 (144.2%)	**0.065 (151.2%)**
Center Object Removal	0.046 (100.0%)	0.050 (108.7%)	0.047 (102.2%)
Min Contact Range	0.050 (100.0%)	0.055 (110.6%)	0.052 (104.1%)
Min Overlap	0.042 (100.0%)	0.044 (105.9%)	0.043 (101.4%)
Two Cluster Separation	0.044 (100.0%)	0.050 (113.6%)	0.047 (106.8%)
ClusterPush	**0.056 (100.0%)**	**0.063 (112.5%)**	0.060 (107.1%)

TABLE II: Results for Multi-point Pushing over 24,000 scenarios with 10 to 15 objects. Contact edge length is approximated by the jaw width of the ABB YuMi robot, and the distance between two contact points is 60% of the radius of the push objects smallest bounding circle. Performance is defined as the ratio of the change of singulation distance after and before a push (normalized to single-point push = 100%). Multi-point contact improves pushes for all policies, but has the largest effect on Cluster Diffusion. This effect may occur due to Cluster Diffusion not planning pushes that avoid contact with other objects, and thus benefiting from sticking pushes.

linear policies in scenarios with 10 to 15 objects. Multi-point pushing can consistently improve push performance for linear push policies in complex scenarios with 10-15 objects. Furthermore, edge pushing improves singulation of ClusterPush by 12.5% while two-point pushing with the above heuristic improves singulation by 7.1% compared to a single-point push.

Since ClusterPush performs well on hard cases and thus has less room to improve, we compare edge pushes and two-point pushes against single-point pushes for other policies. An edge push can improve the worst-performing policy, Quasi-random, by 14.8%; a two-point push can improve Quasi-random by 7.4%. As multi-point pushes serve to prevent objects from slipping away, the policies that seek to minimize contact between objects (Center Object Removal, Minimum Contact Range, Minimum Overlap) benefit less than Cluster Diffusion, which seeks to push one object away from the clutter. The Cluster Diffusion Policy benefits the most from multi-point pushing, achieving a 44.2% improvement from edge pushes and 51.2% improvement from two-point pushes.

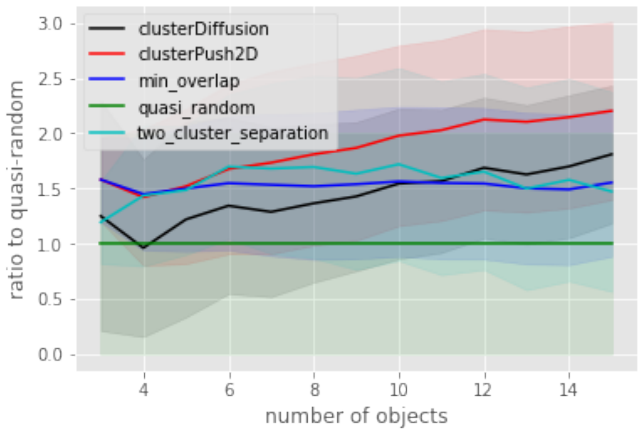

Fig. 6: Mean and standard deviation of the singulation distance increase after pushes, normalized by quasi-random performance. ClusterPush outperforms other policies as the number of object increases.

VII. Physical Experiments

We plan pushes on an ABB YuMi robot using the ClusterPush policy and Cluster Diffusion (the best previously proposed baseline) on 40 sets of three to seven wooden blocks sampled from square blocks, isosceles right triangle blocks, and rectangle blocks. For each set of cluttered objects, we use a Photoneo PhoXi depth sensor to acquire a point cloud of the bin. Then, we perform object segmentation using the Point Cloud Library implementation of Euclidean Cluster Extraction [29]. For a depth-image input, we smooth contours of segmented point clusters using OpenCV's implementation of the Douglas-Peucker algorithm [30] and import the resulting polygon into the Box2D simulator. We approximate object center of mass as the centroid of the segmented cluster. The robot then executes pushes by following the linear push trajectory planned by an object singulation policy. The single point pushes are executed by closing the parallel jaw gripper and using the jaw tips, edge pushes by closing the parallel jaw gripper and using the planar side of the jaws, and two-point pushes by opening the gripper based on the pre-computed distance as in Section VI: 60% of the radius of the push object's smallest bounding circle.

ClusterPush outperforms Cluster Diffusion by 18.8% in easy scenarios with three square blocks, 30.0% in hard scenarios with mixed shapes, and 23.6% on average over 5 sets in 8 different configurations. Edge pushing and two-point pushing improve push performance by 18.4% and 28.4% on average respectively.

VIII. Future Work

Although this study focuses on the planar case and uses simulation of frictional pushing that is deterministic and hence imprecise, insights from this planar study are a valuable step toward policies for real-world, more complex singulation policies. ClusterPush can be adapted for this purpose and we look forward to generalizing it to more realistic objects in physical settings. We also plan to investigate metrics that are more closely related to real-world graspability.

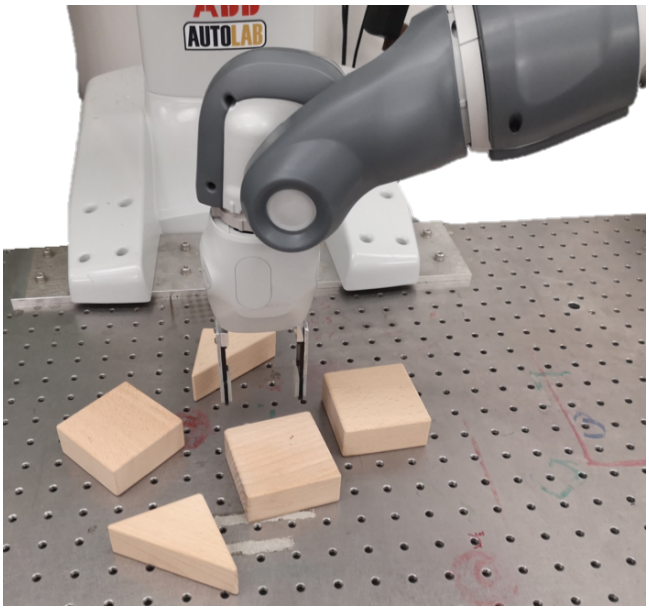

Fig. 7: Front view of the robot arm and the setup.

	Easy	Hard	Average
Single-point ClusterPush	0.366	0.194	0.290
Single-point Cluster Diffusion	0.308	0.149	0.235
Edge ClusterPush	0.411	0.227	0.344
Edge Cluster Diffusion	0.377	0.188	0.277
Two-point ClusterPush	0.445	**0.245**	**0.358**
Two-point Cluster Diffusion	**0.463**	0.169	0.316

TABLE III: Results of physical experiments on 40 sets of three to seven wooden blocks. Performance is defined as the ratio of the change of singulation distance after and before pushing. ClusterPush outperforms Cluster Diffusion in most scenarios, and multi-point contact can improve both policies by more than 10% in physical experiments.

IX. Acknowledgments

This research was performed at the AUTOLAB at UC Berkeley in affiliation with the Berkeley AI Research (BAIR) Lab, Berkeley Deep Drive (BDD), the Real-Time Intelligent Secure Execution (RISE) Lab, and the CITRIS "People and Robots" (CPAR) Initiative. The authors were supported in part by donations from Siemens, Google, Toyota Research Institute, Honda, Intel, Hewlett-Packard and by equipment grants from PhotoNeo, NVidia, and Intuitive Surgical. We thank our colleagues Carolyn Matl, Ajay Kumar Tanwani, Daniel Seita, Jeffery Ichnowski and Michelle Lu for providing helpful feedback and suggestions.

References

[1] M. Stilman, J.-U. Schamburek, J. Kuffner, and T. Asfour, "Manipulation planning among movable obstacles," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, IEEE, 2007, pp. 3327–3332.

[2] M. Dogar and S. Srinivasa, "A framework for push-grasping in clutter," *Proc. Robotics: Science and Systems (RSS)*, vol. 1, 2011.

[3] M. Laskey, J. Lee, C. Chuck, D. Gealy, W. Hsieh, F. T. Pokorny, A. D. Dragan, and K. Goldberg, "Robot grasping in clutter: Using a hierarchy of supervisors for learning from demonstrations," in *Proc. IEEE Conf. on Automation Science and Engineering (CASE)*, IEEE, 2016, pp. 827–834.

[4] M. Dogar, K. Hsiao, M. Ciocarlie, and S. Srinivasa, "Physics-based grasp planning through clutter," in *Proc. Robotics: Science and Systems (RSS)*, Jul. 2012.

[5] N. Kitaev, I. Mordatch, S. Patil, and P. Abbeel, "Physics-based trajectory optimization for grasping in cluttered environments," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, IEEE, 2015, pp. 3102–3109.

[6] 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, pp. 1249–1256.

[7] 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," in *Proc. Robotics: Science and Systems (RSS)*, 2017.

[8] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *Int. Journal of Robotics Research (IJRR)*, vol. 37, no. 4-5, pp. 421–436, 2018.

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

[10] *Box2D: A 2D physics engine for games*, www.box2d.org/, November 2010.

[11] K. M. Lynch and M. T. Mason, "Stable pushing: Mechanics, controllability, and planning," *Int. Journal of Robotics Research (IJRR)*, vol. 15, no. 6, pp. 533–556, 1996.

[12] S. Akella and M. T. Mason, "Posing polygonal objects in the plane by pushing," *Int. Journal of Robotics Research (IJRR)*, vol. 17, no. 1, pp. 70–88, 1998.

[13] K. Y. Goldberg, "Orienting polygonal parts without sensors," *Algorithmica*, vol. 10, no. 2-4, pp. 201–225, 1993.

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

[15] K.-T. Yu, M. Bauza, N. Fazeli, and A. Rodriguez, "More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, IEEE, 2016, pp. 30–37.

[16] D. Ma and A. Rodriguez, "Friction variability in auto-collected dataset of planar pushing experiments and anisotropic friction," *arXiv preprint arXiv:1802.10089*, 2018.

[17] A. Cosgun, T. Hermans, V. Emeli, and M. Stilman, "Push planning for object placement on cluttered table surfaces," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, IEEE, 2011, pp. 4627–4632.

[18] A. Eitel, N. Hauff, and W. Burgard, "Learning to singulate objects using a push proposal network," *arXiv preprint arXiv:1707.08101*, 2017.

[19] *Bullet physics engine*, bulletphysics.org, 2010.

[20] *Open dynamics engine (ode)*, www.ode.org, 2006.

[21] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, IEEE, 2012, pp. 5026–5033.

[22] A. Kloss, S. Schaal, and J. Bohg, "Combining learned and analytical models for predicting action effects," *CoRR*, vol. abs/1710.04102, 2017.

[23] W. Agboh and M. Dogar, "Pushing fast and slow: Task-adaptive planning for non-prehensile manipulation under uncertainty," in *Springer Proceedings in Advanced Robotics (SPAR) series on 2018 Workshop on the Algorithmic Foundations of Robotics (WAFR 2018)*, 2018.

[24] J. Xie and N. Chakraborty, "Dynamic models of planar sliding," in *Springer Proceedings in Advanced Robotics (SPAR) series on 2018 Workshop on the Algorithmic Foundations of Robotics (WAFR 2018)*, 2018.

[25] A. Ajay, J. Wu, N. Fazeli, M. Bauza, L. P. Kaelbling, J. B. Tenenbaum, and A. Rodriguez, "Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2018.

[26] Y. Jiang and C. K. Liu, "Data-augmented contact model for rigid body simulation," *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019.

[27] D. Seita, F. T. Pokorny, J. Mahler, D. Kragic, M. Franklin, J. Canny, and K. Goldberg, "Large-scale supervised learning of the grasp robustness of surface patch pairs," in *Proc. IEEE Int. Conf. on Simulation, Modeling, and Programming of Autonomous Robots (SIMPAR)*, IEEE, 2016, pp. 216–223.

[28] A. F. van der Stappen, K. Goldberg, and M. Overmars, "Geometric eccentricity and the complexity of manipulation plans," *Algorithmica*, vol. 26, no. 3-4, pp. 494–514, 2000.

[29] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 1–4, 2011.

[30] J. Hershberger and J. Snoeyink, "An o (n log n) implementation of the douglas-peucker algorithm for line simplification," in *Int. S. on Computational Geometry (SoCG)*, ACM, 1994, pp. 383–384.