Learning Collaborative Pushing and Grasping Policies in Dense Clutter

Bingjie Tang, Matthew Corsaro, George Konidaris, Stefanos Nikolaidis, Stefanie Tellex

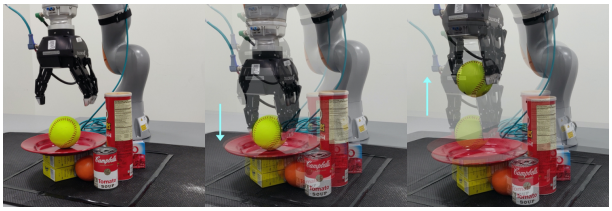
***Abstract*— Robots must reason about pushing and grasping in order to engage in flexible manipulation in cluttered environments. Earlier works on learning pushing and grasping only consider each operation in isolation or are limited to top-down grasping and bin-picking. We train a robot to learn joint planar pushing and 6-degree-of-freedom (6-DoF) grasping policies by self-supervision. Two separate deep neural networks are trained to map from 3D visual observations to actions with a Q-learning framework. With collaborative pushes and expanded grasping action space, our system can deal with cluttered scenes with a wide variety of objects (e.g. grasping a plate from the side after pushing away surrounding obstacles). We compare our system to the state-of-the-art baseline model VPG [1] in simulation and outperform it with 10% higher action efficiency and 20% higher grasp success rate. We then demonstrate our system on a KUKA LBR iiwa arm with a Robotiq 3-finger gripper.**

I. INTRODUCTION

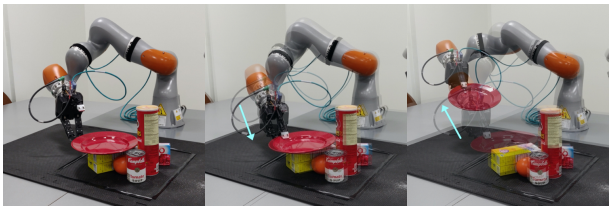
Imagine a robot trying to clean up a messy dinner table. Two main manipulation skills are required: grasping that enables the robot to pick up objects, and planar pushing that allows the robot to isolate objects in the dense clutter to find a good grasp pose. It is necessary to identify grasps within the full 6D space because top-down grasping is insufficient for objects with diverse shapes, e.g. a plate or a filled cup. Pushing operations are also essential because in real-world scenarios, the robot's workspace can contain many objects and a collision-free direct grasp may not exist. Pushing operations can singulate objects in clutter, enabling future grasping of these isolated objects. For example, in Fig. 1(c) the robot must push the sugar box in the middle to make a future grasp of the sugar box on the left accessible. Therefore, we explore learning joint planar pushing and 6-degree-of-freedom (6-DoF) grasping policies in a cluttered environment.

Many research studies have focused on learning to push or grasp in isolation instead of learning multi-action policies [2]–[4]. Fig. 1(c) shows that learning the synergies between the two actions is essential to decluttering tasks since pre-grasp pushing actions are often necessary in densely cluttered scenarios. Some only focus on manipulating a single object while in the real world target objects are often surrounded by other objects [5]. Other approaches to multiple-object tasks only consider top-down grasps or bin-picking scenarios [1], [6] which largely limit the diversity of grasps (See Fig. 1(b)).

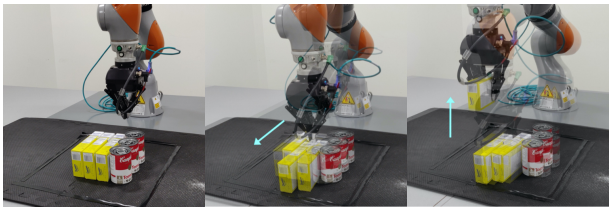
We propose an approach that learns multi-action policies in the full 6-DoF pose space of the robotic hand. We model the problem of learning joint policies for planar pushing and 6-DoF grasping as a Markov Decision Process (MDP) and use a deep reinforcement learning algorithm trained end-to-end with self-supervision. The main novel aspects of our system are twofold:



(a) Grasp a baseball from the top.



(b) Grasp a plate from the side.



(c) Grasp a sugar box after a push.

Fig. 1. Pushes and 6-DoF grasps are both essential in decluttering tasks. (a) the robot grasps a softball from the top. (b) the necessity of 6-DoF grasping; the robot can only grasp the red plate from the side. (c) the necessity of pushing; the sugar box on the left is only accessible after a push.

- In a Q-learning framework, we jointly train two separate neural networks with reinforcement learning to maximize a reward function. The reward function is defined as only encouraging successful grasps; we do not directly reward pushing actions, because such intermediate rewards often lead to undesired behavior [7].
- We tackle the problem of limited top-down grasping action space by integrating a 6-DoF grasping pose sampler [8] rather than using dense pixel-wise sampling from visual inputs and only considering hard-coded top-down grasping candidates [1].

We evaluate our approach by task completion rate, action efficiency, and grasp accuracy in simulation and demonstrate performance on a real robot implementation. Our system shows 10% higher action efficiency and 20% higher grasp success rate than VPG[1], the current state-of-the-art, indicating significantly better performance in terms of both higher prediction accuracy and quality of grasp pose selection.

II. Background & Related Work

Our work is in the intersection of pushing, grasping and learning joint policies for both actions.

a) Pushing: Pushing is one of many non-prehensile manipulation modalities [9], enabling complex manipulation tasks to be performed when objects are too large, too heavy, or too cluttered. Most work on robotic pushing focuses on generating accurate predictions for the outcome of a push. Many use pure analytical methods [10]–[14], relying heavily on strong assumptions (e.g. quasi-static assumption, [15]) and do not consider uncertainty. More studies use data-driven approaches, estimating the parameters of the analytical model based on observations [16]–[20]. While these data-driven methods can generalize the dynamics of pushing, they require explicit object modeling and are highly sensitive to parameters, unlike our approach which is learned from data. In terms of vision-based methods, Salganicoff, Metta, Oddera, *et al.* [21] introduced a learning method to push an unknown object with a single rotational point contact to an arbitrary goal point. Lau, Mitani, and Igarashi [22] expanded their method to handle objects with more complex shapes. Besides these, many studies leverage deep learning techniques to model pushing dynamics [23]–[25]. More recent work used deep reinforcement learning methods to train pushing control policies directly [2], [26]–[28]. However, these works only focused on stable and accurate pushing in isolation without considering potential subsequent manipulations, or have not explored pushing in a dense-cluttered workspace. Our work combines both pushing and grasping, allowing the system to learn synergistic behavior between pushing and grasping to autonomously clear a table.

b) Grasping: Grasping is widely explored so here we only discuss the branch of grasp pose detection most related to our work. Grasp pose detection finds an end-effector configuration that maximizes a grasp quality metric. Previous methods can be broadly divided into two main categories: model-based and model-free [29]. Model-based methods use mathematical and physical models of geometry, kinematics, and dynamics to calculate stable grasps [30]–[33]. These approaches typically use a pre-built grasp database of common 3D object models labelled with grasps and their quality and to transfer to similar objects at runtime [34]. However, these methods struggle to generalize to novel objects and objects placed in dense clutter due to the limited database. Other studies proposed data-driven model-free methods that leverage vision-based deep learning techniques to learn grasping policies for unknown objects. Most of them focus on grasping isolated objects [35]–[39] while our approach can grasp in dense clutter. Some studies explored grasping in cluttered scenes but are limited to top-down grasping or bin-picking scenarios (which are both 3-DoF grasping due to the grasping pose encoding or limited workspace) [34], [40]–[44]. We used a 6-DoF grasping pose generator from Pas, Gualtieri, Saenko, *et al.* [8] as grasp sampler in our framework to make more flexible, stable grasping poses accessible and increase the sampling efficiency at the same time. However, the spatial grasping pose generator alone cannot perform planning for heavily cluttered situation where pre-grasping manipulation (e.g. pushing) is required to reposition objects for future grasping. This limitation also holds for other works on 6-DoF grasping planning in clutter that do not consider pre-grasping manipulation skills [3], [4]. Our work, in contrast, combines pre-grasping manipulation skills with 6-DoF grasp planning.

c) Grasping Combined with Pushing: Mason [15] presented a theoretical investigation of the use of pushing in manipulation. Lynch and Mason [11] proposed one of the first planners that integrated collaborative pushing operations. King, Klingensmith, Dellin, *et al.* [5] presented the pre-grasp manipulation problem as trajectory optimization, including two modes of interactions: pushing and pick-and-place, and completed multi-step planning for several manipulation tasks. However, their approach only focused on a single object. Dogar and Srinivasa [45] and Boularias, Bagnell, and Stentz [46] both presented methods for robust grasping planning under uncertainty in dense clutter by exploiting pre-grasping actions. The policies in the former remain largely hand-crafted and in the latter relied on hand-crafted representations for pushing and grasping selection during training. Our work in contrast, uses learned representations for deciding where to push and grasp to clear the table.

Most related to our work, Zeng, Song, Welker, *et al.* [1] and Deng, Guo, Wei, *et al.* [44] proposed to leverage deep Q-Network (DQN) to learn pushing and grasping operations jointly, and Kalashnikov, Irpan, Pastor, *et al.* [6] introduced QT-Opt, a scalable self-supervised vision-based reinforcement learning framework that performs closed-loop grasping with non-prehensile pre-grasp manipulations. Although [6] achieved an impressive 96% grasping successful rate in bin-picking scenarios, their model was trained at high cost (580k grasp attempts over 7 robot arms). Also, these works are restricted to top-down grasping or bin-picking, which limits grasp diversity and may fail for objects with non-planar surface (e.g. a standing bottle). Also, in [1] they directly reward pushing primitives to encourage pushing. This might cause problems such as pushing objects out of workspace or performing unnecessary pushing actions when there are only a few objects scattered on the table and potential successful grasps are available [7].

We followed the Q-learning framework and Markov Decision Process (MDP) formulation illustrated in Zeng, Song, Welker, *et al.* [1]. We reconstructed the grasping planning module to enable 6-DoF grasp prediction and removed the direct reward for pushing actions. These changes to the reward function allow the model to operate more effectively when there are only a few objects on the table. Note that because we introduced 6-DoF grasping in our system so we expanded the grasping action space in the MDP formulation.

III. Learning Joint Pushing & Grasping Policies

A. Problem Formulation

a) MDP: Following Zeng, Song, Welker, *et al.* [1], we formulate this sequential decision making process as a Markov Decision Process (MDP). An MDP is formally

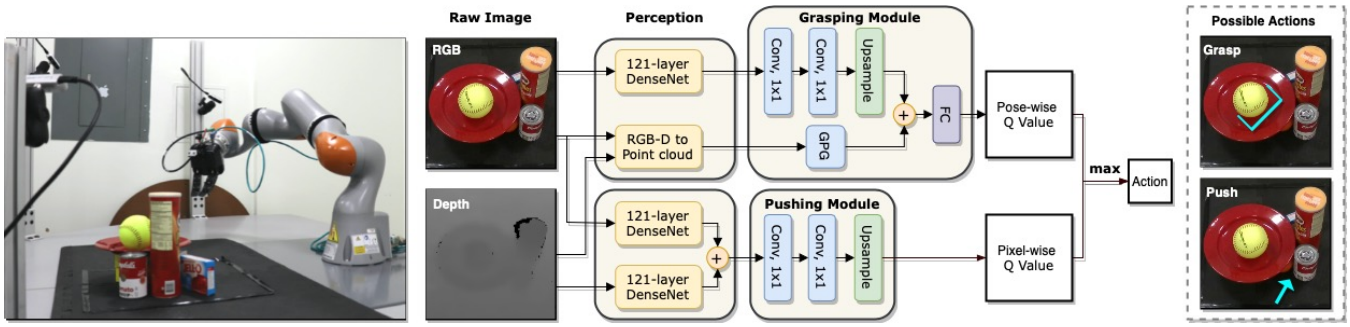


Fig. 2. **System Overview** Our system takes in RGB-D visual observation of the scene and predicts the next action primitive (i.e. pushing or grasping), as well as the position and orientation for executing that action. The learning objective is to find a policy that always chooses the action that maximizes the expected success of current/future grasps.

defined by a tuple $M = (\mathscr{S}, \mathscr{A}, \mathscr{T}, r, \gamma)$ in which $\mathscr{S}$, $\mathscr{A}$, $\mathscr{T}$, r and γ denote the state space, the action space, the transition function, the reward function and the discount factor respectively. Our goal is to find an optimal policy π^* that maximizes the expected discounted sum of future rewards: $R_t = \Sigma_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1})$. We define the state space $s \in \mathscr{S}$ as the current visual observation captured by cameras, an RGB-D image covers the robot's workspace. The actions $a \in \mathscr{A}$ consist of the choice of action primitive, the end-effector position and orientation:

$$a = (\psi, x, \theta), \psi \in \{push, grasp\}, x, \theta \in \mathbb{R}^3,$$

where x and θ denote the end-effector position and orientation respectively. The reward is defined as a binary, sparse reward function: the reward $R(s_t, s_{t+1})$ is 1 for a successful grasp and 0 otherwise. We detect grasp success by checking whether the antipodal distance between parallel-jaw gripper fingers after a grasp attempt is lower than a pre-defined threshold. No direct reward for pushing actions is included in our reward formulation. Our model learns collaborative pushes that can benefit future grasps only through the reward for successful grasps.

b) Learning Process: We use an off-policy Q-learning algorithm to learn a deterministic policy for collaborative pushing and grasping. Given any state s_t at time t, the robot takes an action a_t based on a policy $\pi(s_t)$ and transitions to a new state s_{t+1} with a reward $r(s_t, a_t, s_{t+1})$. Every timestep t the robot picks an action that has the maximum Q-value for a given state, in other words, our policy picks an action by maximizing the Q-function which measures the expected reward for taking action a_t at state s_t. And we used temporal difference (TD) learning in which the learning objective is to minimize the temporal difference of the current Q-value function $Q_\pi(s_t, a_t)$ to the label y_t:

$$y_t = r(s_t, a_t, s_{t+1}) + \gamma Q(s_{t+1}, \arg\max_{a'}(Q(s_{t+1}, a')),$$

where a' is the set of all available actions.

c) Loss Function: We use different loss functions for pushing and grasping skills. We only compute the loss for the selected pixel/pose (where the robot will take the next action), all other pixels/poses backpropagate with loss 0. For pushing, we generate y_i by calculating the scene image difference after the push, if it is higher than a threshold we consider the push successful. This method may be sensitive to the manually selected threshold, hence we use the Huber loss for pushing because it is less sensitive to inaccurate labels [47]:

$$\delta_t^{push} = |Q^{\theta_i} - y_i^{\theta_i^-}|$$
$$\mathscr{L}_{push} = \begin{cases} \frac{1}{2}(\delta_t^{push})^2, |\delta_t^{push}| < 1, \\ |\delta_t^{push}| - \frac{1}{2}, \text{ otherwise.} \end{cases}$$

For grasping, we can get the ground truth label y_i via the feedback signal from the gripper. Therefore, we can assume all labels are accurate and use the binary cross-entropy loss. Let $\hat{y}_i^k$ and y_i^k denote the estimate of Q-value and true label for grasp pose k at iteration i,

$$\delta_t^{grasp} = -\sum_k y_i^k \log \hat{y}_i^k$$
$$\mathscr{L}_{grasp} = \delta_t^{grasp} = -(y_i \log(Q^\phi) + (1 - y_i) \log(1 - Q^\phi)).$$

$\delta_t^{push}/\delta_t^{grasp}$ denotes the temporal difference for pushing/grasping, θ_i/ϕ_i are the parameters of pushing/grasping prediction network at iteration i, and θ_i^- are the target push network parameters, which are fixed between individual updates.

B. Perception Module

Given the state s_t at time t, which is the RGB-D image provided by cameras, perception module generates feature representations and point clouds and feed them into push/grasp modules, structural details of the perception module is shown in Fig.2. All three 121-layer DenseNet [48] are pretrained on ImageNet [49], we use ReLU [50] activation for all convolutional layers and the fully-connected layer, and we apply spatial batch normalization for all convolutional layers.

C. Pushing Module

The pushing module aims to predict the most beneficial pushing action a_t^{push} for future grasps among a pixel-wise sampling of end-effector locations and orientations given s_t.

The pushing action in our system is defined similar to Zeng, Song, Welker, *et al.* [1]. Given s_t at time t, a planar

pushing action a_t^{push} is parameterized as (x, θ) in which x and θ denote the location (x_x, x_y) and the orientation of the end-effector when execute the push. Since all the pushes are planar, θ is the rotated angle of the end-effector in xy-plane.

The detailed push prediction network structure can be found in Fig. 2. It is a feed-forward fully convolutional network which maps the visual observation of the scene at time t to a dense pixel-wise Q-value estimate of the workspace. When the robot executes a push, it closes its gripper, moves the gripper to the selected location and pushes for 10cm in the selected direction. During the execution, the gripper holds the same height from the tabletop in order to keep the push planar and stable. We keep length fixed for pushing since our goal is exploring the synergies between actions. However, direction is not fixed; we rotated the original visual observation in 16 orientations before prediction so that every pixel gets 16 Q-value estimates for executing a push in each direction.

D. Grasping Module

This module contains a grasp pose generation algorithm, a convolutional network for color features, and a fully-connected layer that learns to find a grasp candidate a_t^{grasp} at time t that can maximize the probability of future grasp success. Enabling the robot to select a grasp from the full 6-DoF space of poses is the key advance of our model, compared to several previous works that are limited to top-down grasping or bin-picking [1], [6].

We define the 6-DoF grasping action a_t^{grasp} in our system as (x, θ), in which x and θ denote the 3D location (x_x, x_y, x_z) and the orientation $(\theta_\alpha, \theta_\beta, \theta_\gamma)$ of the end-effector when executing the grasp. These 6-DoF poses are produced by the grasp candidate generation algorithm defined in Pas and Platt [51] and used in Pas, Gualtieri, Saenko, *et al.* [8]. Given s_t, a set of candidates is generated by sampling points from the point cloud to use as grasp centroids, with grasp orientations determined by each centroid's local geometry. We filter grasping samples by the end-effector position and their approach direction before feeding into the fully-connected layer; if either the pre-grasp or target end-effector position is out of robot's workspace, we consider this sample as invalid.

As shown in Fig. 2, grasp prediction network takes color feature extraction from 121-layer DenseNet [48] and point clouds from raw RGB-D camera data as input. ϕ_g^{color} are two convolutional layers and a bilinearly upsampling layer for color features. The point cloud is passed into ϕ_g^{gpg}, a grasping pose generator (GPG) [8]. Q-values for every grasping candidate sampled from GPG are calculated by concatenating the output from ϕ_g^{color} and ϕ_g^{gpg}, passing through a fully-connected layer.

IV. Evaluation

A. Evaluation Metrics

To assess our model's performance, we assess its speed and accuracy at picking up objects. Each trial is a new clutter of N objects in the workspace and the robot must remove all objects by grasping. For each test, we execute 10 trials and compute the average evaluation scores. We quantitatively evaluate the performance of the learned pushing and grasping policies with three evaluation metrics:

- **Completion**: If the robot successfully picked up every object without more than 10 consecutive failed grasping attempts in one test trial, we consider this trial a completion. This shows the ability of the system to finish the task.
- **Grasp Success (GS)**: the ratio of successful grasps in all grasp attempts per completion, measures the accuracy of the grasping policy. This matrix indicates how quickly the policy can reach a completion.
- **Action Efficiency (AE)**: the ratio of number of objects N to the number of actions executed before completion.

B. Simulation Experiments

In this section, we present system evaluation with CoppeliaSim [52] simulator. In simulation, we compare the quantitative results with VPG from [1] on a collection of basic geometric shapes and provide quantitative results of our model in simulation on the Yale-CMU-Berkeley (YCB) dataset [53] to show its performance on decluttering more challenging objects, that are representative of real-life tasks.

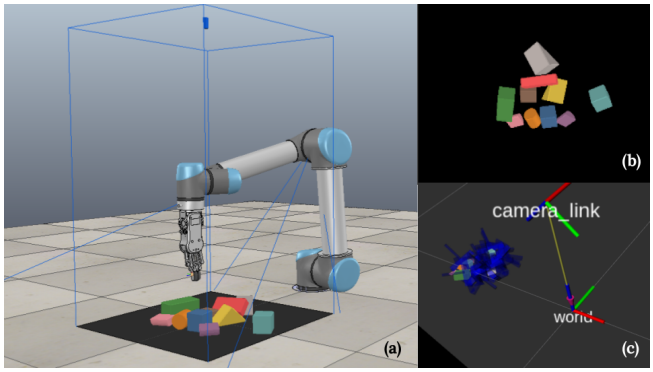


Fig. 3. **Simulation Configuration** (a) shows our environment configuration in CoppeliaSim [52]. (b) shows the view captured by th RGB-D camera mounted on robot's shoulder. (c) shows the point cloud we build from the raw RGB-D images, and generated grasping samples (blue) by Pas, Gualtieri, Saenko, *et al.* [8], the camera frame (camera_link) and the world frame (world).

a) Simulation Configuration: We use CoppeliaSim [52] simulator with Bullet Physics 2.83 for dynamics and CoppeliaSim's built-in inverse kinematics for motion planning. We test our system on a simulated UR5 robot with an RG2 gripper. A RGB-D camera is placed to provide visual inputs from an ego-centric view of the workspace for the robot. The camera resolution is 640×480, with a near/far clipping plane of $0.01/10.0m$, a perspective angle of $54.7°$. Fig.3 shows the simulated CoppeliaSim [52] scene configuration we use for experiments. We did not add any noise to images captured by the vision sensor and we removed the simulated floor from the point cloud data for better grasping pose detection.

b) Baseline Model: We compare our system to the Visual Pushing for Grasping (VPG) model [1]. VPG is the state-of-the-art model that also learns joint pushing and grasping polices for clutter clean-up. However, it has

three main constraints. First, the grasping action space of VPG is restricted to top-down grasping. Second, VPG uses computationally-consuming dense pixel-wise sampling for both actions. Last, it includes direct reward for pushing actions that made changes to the scene. This can be problematic, it may encourage the robot to push objects in clutter, but risks rewarding the robot for redundant pushes.

c) General Tasks: We first verify that our model can pick up basic geometric shaped objects with learned policies. Our training shape set includes 5 different shapes and the testing shape set includes the training shape set and 5 novel shapes. For each trial, we randomly select shapes from corresponding shape set to generate a new arrangement of N objects with randomly selected shape/color/position/orientation, so clutters in testing have never appeared in training even their shape sets overlap. We train our model to convergence with 600 randomly generated 10-object scenes and test it with both 10-object scenes and more cluttered 30-object scenes. Since 5 novel shapes are added during testing so we can also evaluate how our model generalizes to novel objects.

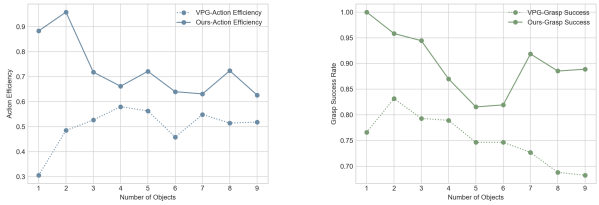
The testing results for general tasks are shown in Table I, which indicates that though both methods have 100% completion rate, our method outperforms VPG in both action efficiency and grasp success rate. Also, as the number of objects increases, our model, benefiting from more flexible 6-DoF grasps, has a higher grasp success rate while VPG's grasp success decreases by 8.3%, which implies our model's ability to adapt to more cluttered scenarios. To verify that removing the pushing reward contributes to our improved performance, we also tested our model with the pushing reward used in Zeng, Song, Welker, *et al.* [1] included, for both 10-object and 30-object scenarios. As shown in Table I, the added pushing reward results in a mostly comparable grasp success rate, but leads to a significant decrease in action efficiency compared to our method. These results show that including a direct pushing reward leads to redundant pushes, thereby decreasing performance.

Two methods are also tested in scenarios with fewer objects (< 10), the results can be found in Fig.4. It shows that our system has a better performance over all fewer-object scenarios. The action efficiency is significantly higher than the baseline which indicates that our model has fewer redundant pushes, which comes from removing the pushing reward. However, with more objects on the table, we find the action efficiency of our system dropped. Since the action efficiency is defined as the ratio of number of objects N to the number of actions executed before completion, with more objects in the workspace, more pushing operations are expected to singulate objects from the clutter for future grasp. With at least N grasp attempts to finish the task and more expected pushing operations, the action efficiency may decrease in scenarios with more objects.

Interestingly we find VPG has a lower action efficiency ($\leq 50\%$) for sparse scenes (with ≤ 2 objects) which are normally considered easier for grasping. We hypothesize that this is due to the direct reward for pushing actions that encourage the robot to execute unnecessary pushes, hence reducing the robot's action efficiency. Our model, on the other hand, learned to pick up objects in the workspace with significantly higher ($\geq 60\%$) action efficiency.

TABLE I
GENERAL TASK RESULTS (MEAN %)

Model	Num. of Obj.	Completion	AE	GS
VPG [1]	10	100.0	51.9	68.3
Ours	10	100.0	**62.7**	**88.9**
Ours w. Pushing Rwd	10	100.0	50.2	87.7
VPG [1]	30	100.0	67.7	60.9
Ours	30	100.0	**87.9**	**95.6**
Ours w. Pushing Rwd	30	100.0	62.2	89.3



(a) Action Efficiency (b) Grasp Success

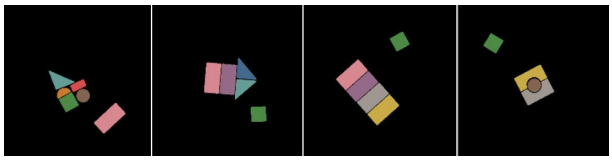
Fig. 4. Fewer-object Clutters

d) Challenging Tasks: We also tested our system with more challenging scenarios: adversarial clutters and YCB-object clutters. Example object arrangements of adversarial clutters and YCB-object clutters can be found in Fig.5.

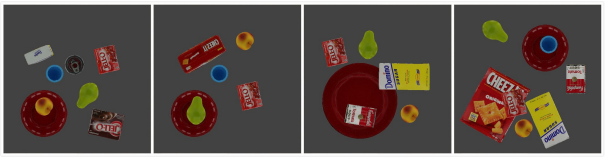
Like shown in Fig.5(a), each adversarial clutter has 3-6 basic geometric shaped objects that are intentionally placed tightly enough that policies with only grasping actions will struggle to finish the task. An isolated object is placed as a sanity check, since a policy should at least be able to pick up the isolated object.

To create YCB-object clutters, we select shapes from Yale-CMU-Berkeley (YCB) dataset [53] instead of formerly used basic geometric shapes so that high-precision, more flexible 6-DoF grasps are essential to finish the task. We include the following objects from the YCB dataset for testing: 001_chips_can, 003_cracker_box ,004_sugar_box, 005_tomato_soup_can, 008_pudding_box, 009_gelatin_box, 015_peach, 016_pear, 024_bowl, 025_mug, 029_plate, 065_a_cups. When we generate random YCB-object clutters, we make two restrictions on 003_cracker_box and 029_plate. We let the object 003_cracker_box can only be placed as "standing" on the table otherwise it cannot be picked up due to its width, and we set the object 029_plate always on top of other objects to give it an offset height so that the gripper will not scratch table surface when try to grasp it. Note that we never train our model on these challenging tasks, we use the same model trained for general tasks (i.e. randomly generated 10-object clutter which only contains basic geometric shapes).

We have 10 test trials for both challenging tasks and calculate the mean completion rate, action efficiency, and



(a) Adversarial Clutters



(b) YCB Dataset Objects

Fig. 5. **(a)** Adversarial clutters used by Zeng, Song, Welker, *et al.* [1] for fair comparison. **(b)** Randomly generated YCB-object clutters.

TABLE II

CHALLENGING TASK RESULTS (MEAN %)

Model	Task	Completion	AE	GS
VPG [1]	Adversarial	82.7	**77.2**	60.1
Ours	Adversarial	**100.0**	65.2	**87.0**
Ours	YCB-object	77.8	49.8	58.4

grasp success rate for each task. The quantitative results for these challenging tasks are shown in Table II.

In the adversarial clutter setting, we see a higher completion rate and grasp success rate compared to VPG which indicates a better performance in terms of finishing the task. However, we observe a 12% lower action efficiency. Our model is able to tackle difficult adversarial clutters that are challenging for VPG given a 17.7% higher completion rate, which we believe is due to additional effective pushes. By the definition of action efficiency, more pushing actions will lower the evaluation score. Since we do not consider the extra pushing actions are redundant in this case, we do not think the lower action efficiency indicates a poorer performance of our model. For YCB-object clutters, the result shows our model is capable of decluttering a wide variety of objects. Though we did see decreased action efficiency and grasp success in YCB-object scenarios due to more challenging object shapes, our completion rate remains relatively high at 77.8%. We did not include the results for applying VPG to YCB-object scenarios because VPG only allows top-down grasps so that it would constantly fail with objects which require side grasps (e.g. plates) and have a 0% completion.

C. Robot Demonstration

The aim of our real robot demonstration is to show that our framework can work with real-life objects. We use a KUKA LBR iiwa arm with the Robotiq 3-finger adaptive gripper for demonstration as shown in Fig.6. We set the gripper to its pinch mode, where two fingers on the same side move close together, to perform parallel-jaw-like grasps. Two Intel RealSense D435 cameras are mounted above the tabletop scene; one captures RGB-D images for the top-down view for color feature extraction and the other provides a side-view point cloud data. Both cameras are calibrated with respect to the robot base. We place objects from the Yale-CMU-Berkeley dataset [53] within a $0.224m^2$ workspace in front of the robot to create a cluttered tabletop scene.

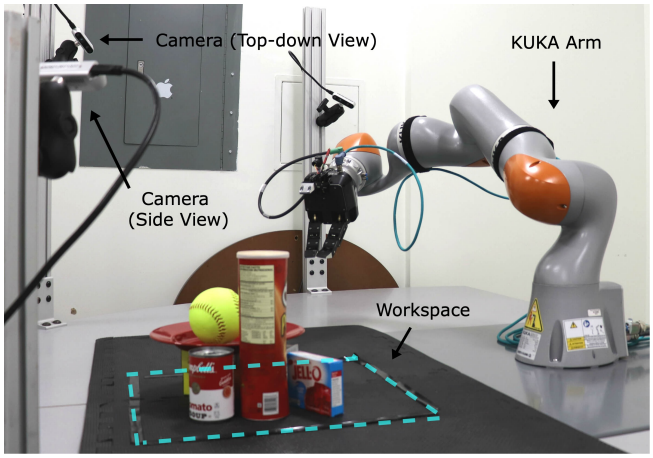


Fig. 6. **Robot Experiment Configuration**

Using the models trained in simulation with synthetic data, our system can successfully pick objects from the scene using flexible 6-DoF grasp poses. We show the necessity of collaborative pushing and 6-DoF grasping in Fig.1. VPG [1] may be able to finish the task in Fig.1(a) but it would fail to pick up the plate in Fig.1(b) due to its limitations on grasping action space. A link to our real robot demonstration video can be found in the supplemental materials.

V. CONCLUSION AND FUTURE WORK

We present a system that jointly learns pushing and 6-DoF grasping policies. Our system outperforms the state-of-the-art learning method [1] in terms of action efficiency, grasp success rate and completion rate in simulated evaluation. Also, we tackle the problem of grasping pose inaccessibility for objects with more challenging shapes, by expanding the grasping action space to 6-DoF.

In our real robot demonstrations, we observe several failure cases that suggest the limitations of our current system. The quality of the point cloud data will dominate the model's performance since the grasp candidates are sampled from it. Also, the surface direction of the point cloud provided by the camera will affect the generated grasp pose orientations which may cause motion planning failures if the camera is not mounted near the origin of the robot's workspace.

In future work, we hope to integrate tactile sensor data in our system for non-rigid objects manipulation since in real-life the robot might encounter fabrics, glasses, and other materials that requires tactile sensor information to manipulate. Also, it would be interesting to replace the sparse reward function with a human feedback signal in our system to help with reward shaping or use natural language for grasping target specification. Investigating a better formulation for the reward function for this framework to accelerate learning and to improve the system performance is worth exploring.

References

[1] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," 2018.

[2] A. Ghadirzadeh, A. Maki, D. Kragic, and M. Björkman, "Deep predictive policy training using reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2351–2358.

[3] Y. Qin, R. Chen, H. Zhu, M. Song, J. Xu, and H. Su, "S4g: Amodal single-view single-shot se(3) grasp detection in cluttered scenes," in *Proceedings of the Conference on Robot Learning*, pp. 53–65.

[4] A. Murali, A. Mousavian, C. Eppner, C. Paxton, and D. Fox, "6-DOF grasping for target-driven object manipulation in clutter," in *2020 IEEE International Conference on Robotics and Automation*.

[5] J. King, M. Klingensmith, C. Dellin, M. Dogar, P. Velagapudi, N. Pollard, and S. Srinivasa, "Pregrasp manipulation as trajectory optimization," in *Robotics: Science and Systems IX*.

[6] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Proceedings of The 2nd Conference on Robot Learning*, pp. 651–673.

[7] M. J. Mataric, "Reward functions for accelerated learning," in *Machine Learning Proceedings 1994*, Elsevier, pp. 181–189.

[8] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp pose detection in point clouds," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1455–1473,

[9] K. Lynch and M. Mason, "Dynamic nonprehensile manipulation: Controllability, planning, and experiments," *International Journal of Robotics Research*, no. 1, pp. 64–92,

[10] M. T. Mason, "Compliant sliding of a block along a wall," in *Experimental Robotics I*, pp. 568–578.

[11] K. M. Lynch and M. T. Mason, "Stable pushing: Mechanics, controllability, and planning," *The International Journal of Robotics Research*, no. 6, pp. 533–556,

[12] S. Akella and M. Mason, "Posing polygonal objects in the plane by pushing," in *Proceedings 1992 IEEE International Conference on Robotics and Automation*.

[13] K. Miyazawa, Y. Maeda, and T. Arai, "Planning of graspless manipulation based on rapidly-exploring random trees," in *The 6th IEEE International Symposium on Assembly and Task Planning: From Nano to Macro Assembly and Manufacturing, 2005.*, pp. 7–12.

[14] M. Dogar and S. Srinivasa, "A framework for push-grasping in clutter," in *Proceedings of Robotics: Science and Systems*.

[15] M. T. Mason, "Mechanics and planning of manipulator pushing operations," *The International Journal of Robotics Research*, no. 3, pp. 53–71,

[16] K. M. Lynch, "Estimating the friction parameters of pushed objects," in *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, 1993, 186–193 vol.1.

[17] S. Walker and J. Kenneth Salisbury, "Pushing using learned manipulation maps," in *2008 IEEE International Conference on Robotics and Automation*, pp. 3808–3813.

[18] M. Kopicki, S. Zurek, R. Stolkin, T. Mörwald, and J. Wyatt, "Learning to predict how rigid objects behave under simple manipulation," in *2011 IEEE International Conference on Robotics and Automation*, pp. 5722–5729.

[19] Jiaji Zhou, R. Paolini, J. A. Bagnell, and M. T. Mason, "A convex polynomial force-motion model for planar sliding: Identification and application," in *2016 IEEE International Conference on Robotics and Automation*, pp. 372–377.

[20] M. Bauza and A. Rodriguez, "A probabilistic data-driven model for planar pushing," in *2017 IEEE International Conference on Robotics and Automation*, pp. 3008–3015.

[21] M. Salganicoff, G. Metta, A. Oddera, and G. Sandini, *A vision-based learning method for pushing manipulation*. University of Pennsylvania.

[22] M. Lau, J. Mitani, and T. Igarashi, "Automatic learning of pushing strategy for delivery of irregular-shaped objects," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3733–3738.

[23] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 64–72.

[24] A. Byravan and D. Fox, "SE3-nets: Learning rigid body motion using deep neural networks," in *2017 IEEE International Conference on Robotics and Automation*, pp. 173–180.

[25] N. Watters, D. Zoran, T. Weber, P. Battaglia, R. Pascanu, and A. Tacchetti, "Visual interaction networks: Learning a physics simulator from video," in *Advances in Neural Information Processing Systems*, vol. 30.

[26] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search," in *2015 IEEE International Conference on Robotics and Automation*.

[27] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[28] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *2017 IEEE International Conference on Robotics and Automation*, pp. 2786–2793.

[29] H. Liang, X. Ma, S. Li, M. Gorner, S. Tang, B. Fang, F. Sun, and J. Zhang, "PointNetGPD: Detecting grasp configurations from point sets," in *2019 International Conference on Robotics and Automation*.

[30] J. Weisz and P. K. Allen, "Pose error robust grasping from contact wrench space metrics," in *2012 IEEE International Conference on Robotics and Automation*.

[31] A. Rodriguez, M. Mason, and S. Ferry, "From caging to grasping," in *Robotics: Science and Systems VII*.

[32] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez, and J. Xiao, "Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge," in *2017 IEEE International Conference on Robotics and Automation*.

[33] A. Zeng, S. Song, M. Niessner, M. Fisher, J. Xiao, and T. Funkhouser, "3DMatch: Learning local geometric descriptors from RGB-d reconstructions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition*.

[34] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. Aparicio, and K. Goldberg, "Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Robotics: Science and Systems XIII*.

[35] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724,

[36] D. Kappler, J. Bohg, and S. Schaal, "Leveraging big data for grasp planning," in *2015 IEEE International Conference on Robotics and Automation*.

[37] X. Yan, J. Hsu, M. Khansari, Y. Bai, A. Pathak, A. Gupta, J. Davidson, and H. Lee, "Learning 6-DOF grasping interaction via deep geometry-aware 3d representations," in *2018 IEEE International Conference on Robotics and Automation*.

[38] Q. Lu, K. Chenna, B. Sundaralingam, and T. Hermans, "Planning multi-fingered grasps as probabilistic inference in a learned deep network," in *Springer Proceedings in Advanced Robotics*, pp. 455–472.

[39] A. Mousavian, C. Eppner, and D. Fox, "6-DOF GraspNet: Variational grasp generation for object manipulation," in *2019 IEEE/CVF International Conference on Computer Vision*.

[40] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *2016 IEEE International Conference on Robotics and Automation*, pp. 3406–3413.

[41] J. Mahler and K. Goldberg, "Learning deep policies for robot bin picking by simulating robust grasping sequences," in *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 515–524.

[42] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, N. Fazeli, F. Alet, N. C. Dafle, R. Holladay, I. Morona, P. Q. Nair, D. Green, I. Taylor, W. Liu, T. Funkhouser, and A. Rodriguez, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," *The International Journal of Robotics Research*,

[43] 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," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436,

[44] Y. Deng, X. Guo, Y. Wei, K. Lu, B. Fang, D. Guo, H. Liu, and F. Sun, "Deep reinforcement learning for robotic pushing and picking in cluttered environment," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 619–626.

[45] M. R. Dogar and S. S. Srinivasa, "A planning framework for non-prehensile manipulation under clutter and uncertainty," *Autonomous Robots*, no. 3, pp. 217–236,

[46] A. Boularias, J. A. Bagnell, and A. Stentz, "Learning to manipulate unknown objects in clutter by reinforcement," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, ser. AAAI'15, Austin, Texas, 1336–1342, ISBN: 0262511290.

[47] R. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision*, pp. 1440–1448.

[48] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[49] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*.

[50] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10, Haifa, Israel, 807–814, ISBN: 9781605589077.

[51] A. ten Pas and R. Platt, "Using geometry to detect grasp poses in 3d point clouds," in *Springer Proceedings in Advanced Robotics*, Springer International Publishing, pp. 307–324.

[52] E. Rohmer, S. P. N. Singh, and M. Freese, "Coppeliasim (formerly V-REP): A versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems*, www.coppeliarobotics.com.

[53] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Yale-CMU-berkeley dataset for robotic manipulation research," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 261–268,