

A Deep Learning Approach to Grasping the Invisible

Yang Yang¹, Hengyue Liang² and Changhyun Choi²

Abstract—We study an emerging problem named “grasping the invisible” in robotic manipulation, in which a robot is tasked to grasp an initially invisible target object via a sequence of pushing and grasping actions. In this problem, pushes are needed to search for the target and rearrange cluttered objects around it to enable effective grasps. We propose to solve the problem by formulating a deep learning approach in a critic-policy format. The target-oriented motion critic, which maps both visual observations and target information to the expected future rewards of pushing and grasping motion primitives, is learned via deep Q-learning. We divide the problem into two subtasks, and two policies are proposed to tackle each of them, by combining the critic predictions and relevant domain knowledge. A Bayesian-based policy accounting for past action experience performs pushing to search for the target; once the target is found, a classifier-based policy coordinates target-oriented pushing and grasping to grasp the target in clutter. The motion critic and the classifier are trained in a self-supervised manner through robot-environment interactions. Our system achieves a 93% and 87% task success rate on each of the two subtasks in simulation and an 85% task success rate in real robot experiments on the whole problem, which outperforms several baselines by large margins. Supplementary material is available at <https://sites.google.com/umn.edu/grasping-invisible>.

Index Terms—Dexterous Manipulation, Deep Learning in Robotics and Automation, Computer Vision for Automation

I. INTRODUCTION

IMAGINE what happens when a young child is looking for a specific toy block buried in clutter, as shown in Fig. 1a. He/she may first push apart the pile of blocks and luckily spot the target block in clutter, then push around the target block to make space for the fingers (we refer to this type of motion as “singulation” [1]) and finally grasp it. We wonder if an intelligent agent can perform such a task. To grasp an invisible target, a robot is required to have the ability of explorational pushing, singulation by target-oriented pushing (i.e., push the target or surrounding objects to make some space), and target-oriented grasping.

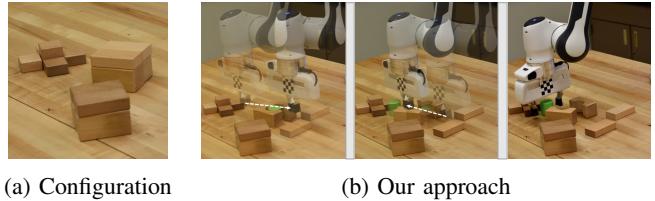
Manuscript received: September, 10, 2019; Revised December, 21, 2019; Accepted January, 14, 2020.

This paper was recommended for publication by Editor Hong Liu upon evaluation of the Associate Editor and Reviewers’ comments. This work was in part supported by the MnDRIVE Initiative on Robotics, Sensors, and Advanced Manufacturing.

¹Y. Yang is with the Department of Computer Science and Engineering, Univ. of Minnesota, Minneapolis, USA yang5276@umn.edu

²H. Liang and C. Choi are with the Department of Electrical and Computer Engineering, Univ. of Minnesota, Minneapolis, USA {liang656, cchoi}@umn.edu

Digital Object Identifier (DOI): see top of this page.



(a) Configuration

(b) Our approach

Fig. 1: **Example configuration of the “grasping the invisible” problem.** The target object is the green cylinder and initially invisible to the robot. We propose to solve the problem with an exploration-singulation-grasping procedure.

Robot pushing [1], grasping [2] or push-grasping [3] has been actively studied but mostly on relatively simple target-agnostic tasks. Without incorporating target information effectively, fast adaptations of these methods (e.g., by applying the target mask directly on the results of the methods) for target-oriented tasks are not successful. While some target-oriented manipulation approaches [4], [5] have been applied to the visible target in sparse environments (i.e., the target is well isolated), the scenarios to apply these methods are limited.

Moving forward from target-agnostic robotic manipulation to target-oriented manipulation challenges not only perception but also manipulation capabilities. Specifically, these challenges are 1) the target information is missing in the case of complete occlusion, and efficient explorations are thus required, 2) perception modules responsible for reasoning about the target region work poorly in dense clutter, 3) data labeling in target-oriented tasks is expensive and 4) the coordination between pushing and grasping (i.e., deciding when and how to push or grasp) is critical but tricky to accomplish.

In this paper, we propose to solve the “grasping the invisible” problem by formulating a deep learning approach in a critic-policy format. The key aspects of our system are

- A robust semantic segmentation module is used to annotate the objects of interest and detect the existence of the target.
- We learn a target-oriented motion critic through deep Q-learning. The critic takes as input visual observations and the target mask and predicts expected future rewards (i.e., Q values) for target-oriented pushing and grasping motion primitives.
- By incorporating Q predictions with domain knowledge, two policies are proposed to make pushing or grasping action decisions in different scenarios. Specifically, a Bayesian-based policy accounting for action experience performs efficient explorational pushing in the complete

occlusion. Once the target is visible, coordinated decision making in target-oriented pushing or grasping is achieved by a classifier-based policy that takes the clutteredness around the target as an input.

- Our learning models (the critic and the classifier) are fully self-supervised through robot-environment interactions.

Fig. 1 presents an example configuration of the “grasping the invisible” problem and how we propose to solve it.

Contributions This paper presents two core technical contributions: 1) a motion critic for target-oriented pushing and grasping motion primitives and 2) a Bayesian-based policy for target exploration and a classifier-based policy for coordinated decision making. Our system can perform target-oriented manipulation tasks with observations from an RGB-D camera.

II. RELATED WORK

Robotic grasping has been well and successfully studied. Classic model-driven approaches [6], [7] find stable force closures for known objects by utilizing prior knowledge such as 3D models of manipulators and objects and their physical properties. Recent data-driven approaches [8], [9], [2] harness learning algorithms and data (collected from humans or physical experiments) to directly map visual observations to grasp representations. Our approach is data-driven and model-agnostic, and the learning models are self-supervised.

To mitigate uncertainty and collision introduced by clutter, non-prehensile manipulation [10], such as pushing, are investigated in both model-driven [11], [12], [13] and data-driven approaches [1], [14], [15]. With the addition of pushing, push-grasping systems [16], [3] are advanced. Analogous to these methods, our approach utilizes non-prehensile pushing to facilitate grasping but further considers both target exploration and singulation.

In comparison to target-agnostic grasping discussed above, target-oriented grasping has been much less explored, except for [17], [4], [5], [15]. In [4], object representations are learned via autonomous robot interaction with the environment for target-oriented grasping. The work in [5] presents a multi-task domain adaptation framework to transfer the learned target-oriented grasping policy from simulation to the real world. The environments in these works, however, tend to be sparse, and thus the application scenarios are heavily limited. Moreover, visibility of the target is a strict precondition for representation computation in [4] or mask extraction in [5]. In contrast, our method does not assume initial visibility of the target and takes advantage of target-oriented pushing to grasp the target in challenging clutter. [15] employs a target-oriented pushing policy analogous to ours, in which a sole push policy is learned via Q-learning for the visible target in clutter. In contrast, we utilize both pushing and grasping for the (visible or invisible) target, and thus the application scenarios are enlarged. In [15], the singulation of the target is confirmed by a hand-crafted module which checks the minimum distance from the nearest object. However, we employ a neural network that learns to decide whether to push or grasp from self-experience. This implicit singulation scheme tends to improve

the action efficiency since complete isolation is not necessary for a successful grasping in clutter, as shown in Fig. 4.

One method that motivates our study is visual pushing for grasping (VPG) by Zeng et al. [3]. VPG proposes a Q-learning framework to learn complementary pushing and grasping policies for robot picking tasks. In VPG, the robot performs target-agnostic tasks and removes all objects from the workspace. In contrast, our approach learns the critic for target-oriented manipulation and proposes subtask policies to solve a more general and complex problem, “grasping the invisible”.

A recent work by Danielczuk et al. [18] proposes action heuristics to choose between grasping, suction, and pushing actions to retrieve a target occluded in clutter. (See Appendix IX-A for more details.) The main differences between our work and [18] are two-fold: 1) Problem formulation. [18] formulates the problem as a POMDP but solves it by hard-coded heuristics. In contrast, we learn a target-oriented motion critic with an MDP formulation when the target is visible. The critic is further utilized in both visible and invisible cases by two policies. 2) Utilization of pushing. In [18], pushing is given a lower priority by the heuristics, and makes up only 5% of executed actions. Instead, we use a learned classifier to coordinate pushing and grasping for solving more challenging arrangements.

III. PROBLEM FORMULATION

The “grasping the invisible” problem in this paper is formulated as follows:

Definition 1. *Given a description (e.g., the class name) of the target object, the goal is to grasp the target via a finite sequence of pushes and grasps. The target can be placed with arbitrary pose and occlusions in dense clutter.*

To tackle the challenge from various poses and occlusions, we divide the problem into two subtasks:

Subtask 1. *If the target is completely buried in clutter, then the robot searches for the target and breaks the structure to make it visible. We name this the **exploration** task.*

Subtask 2. *Though clearly visible, the target might be closely surrounded by other objects, leaving no space for grasping. Thus sole target-oriented grasping is impossible or inefficient without breaking the structured clutter by target-oriented pushing. In the **coordination** task, the target-oriented pushing and grasping need to be sequentially coordinated to grasp the target with the most action efficiency.*

IV. MAKING TARGET-ORIENTED ACTION DECISION

A. System Overview

As shown in Fig. 2, a fixed-mount RGB-D camera captures the predefined workspace. The RGB image is first passed into a pre-trained semantic segmentation module to predict a target mask. The segmentation module robustly detects the target mask, even in heavy occlusions. (See Appendix IX-B for more details.) Then RGB, depth, and mask images are orthographically projected in the gravity direction with a

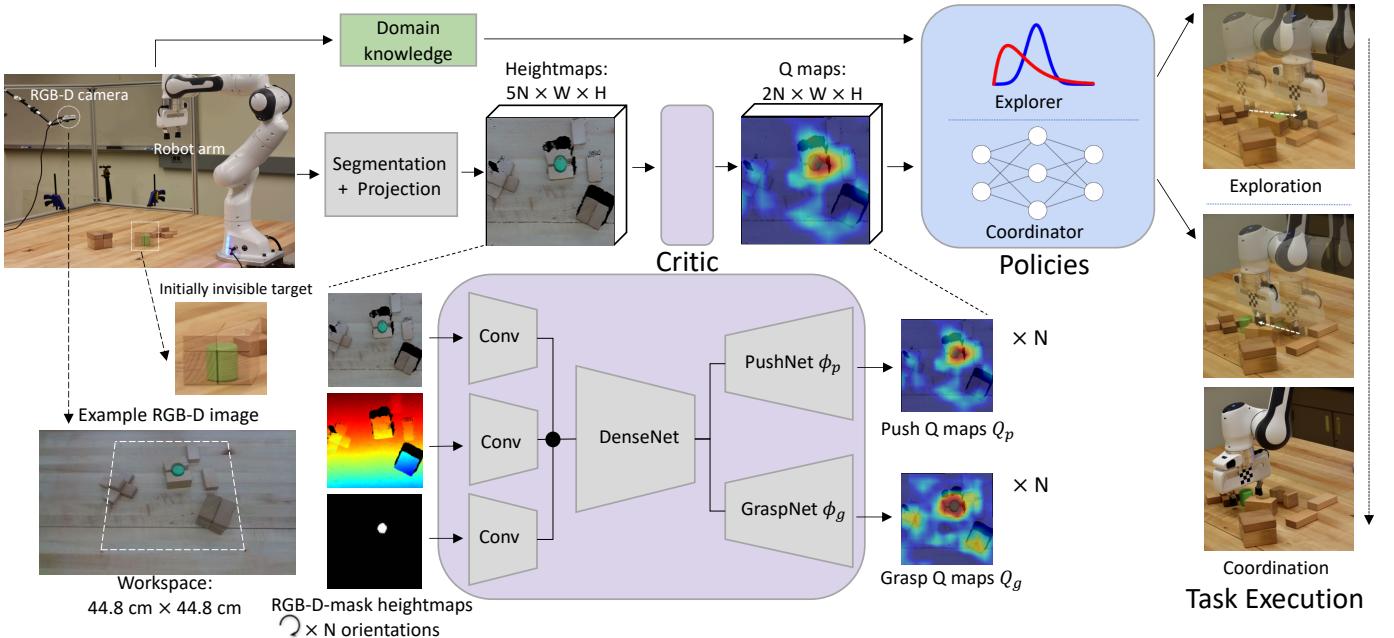


Fig. 2: **Overview.** The visual observations of the scene (we use an example image containing a visible target for illustration) and the target mask from semantic segmentation are orthographically projected to construct heightmaps. The heightmaps are rotated by N orientations for different motion angles and then fed into the motion critic. The critic predicts pixel-wise push and grasp Q values. The two policies, explorer and coordinator, take as input Q predictions and specific domain knowledge to search for the invisible target (exploration) and coordinate pushing and grasping (coordination).

known extrinsic parameter of the camera to construct color heightmap c_t , depth heightmap d_t , and target mask heightmap m_t .

The motion critic takes as input the heightmaps, and the mask heightmap specifies the target. Under the assumption of a visible target, the critic predicts pixel-wise critic scores (i.e., Q values) for target-oriented pushing and grasping motion primitives. And we find suitable action execution location and rotation based on the critic predictions. We discuss the motion critic in detail in Sec. V-A.

Every pixel in the Q maps parameterizes a primitive pushing or grasping action, so there is a direct mapping from Q maps to primitive motions. Every 2D pixel is mapped to a 3D action execution position through the depth heightmap with a heuristic distance. Different motion angles are achieved by rotating the input heightmaps by N orientations before the heightmaps are fed into the networks. There are N corresponding Q maps for pushing and grasping respectively [3]. We choose $N = 16$ in our system, and the angle discretion is thus 22.5° .

To grasp an initially invisible target, top-level policies combine Q predictions and domain knowledge for task execution. First, an exploration policy (explorer) decides a pushing location to search for the invisible target. The explorer uses the height distribution of the workspace and the history of previous actions as the domain knowledge. Once the target is found, a coordination policy (coordinator) coordinates target-oriented pushing and grasping by considering the clutteredness around the target as domain knowledge. We discuss the formulation of the two policies in the following Sec. IV-B.

B. Policy

Exploration policy (explorer). To effectively search for the invisible target in the workspace, we propose a Bayesian-based policy π_e to push¹. As shown in Fig. 3, we use the product of target-agnostic push maps A_p and clutter prior C_p as the prior probability for searching. All-ones mask [5] is used as an input to the target-oriented push network ϕ_p , representing all objects in the workspace as the potential target. Thus the corresponding push Q maps A_p give target-agnostic pushing skills. Besides, the prior knowledge of clutter is also considered—the surfaces of clutter are usually not flat. C_p is generated from the depth heightmap by detecting varying heights. It gives the prior knowledge about the edges of clutter along the intended pushing direction, in the form of probability

¹Pushing is empirically more effective in our setting, while grasping is inefficient or often impossible in challenging arrangements.

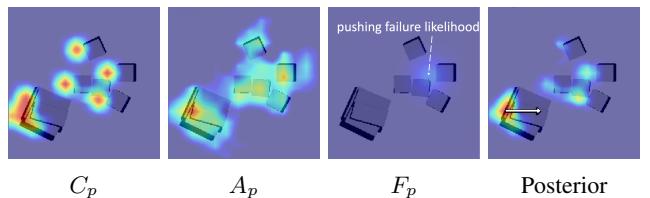


Fig. 3: **Example of exploration probability maps.** Clutter prior C_p , target-agnostic push maps A_p and pushing failure likelihood F_p are multiplied to construct posterior probability maps, according to which the explorational pushing is executed. Only the map representing the intended pushing orientation is visualized here.

maps. A_p and C_p together constitute our prior for exploration. More details are delineated in Sec. VII-A.

To avoid the robot getting stuck at local areas, we account for the past failing experience and construct pushing failure likelihood F_p , which is a multimodal Gaussian likelihood function with low peaks centered at the three most recent locations of failed pushes. In Appendix IX-D we discuss how to construct F_p in detail. In general, F_p reflects the fact that repeated pushes on the previously failed locations are less likely to find the target object. The explorer then makes a pushing decision based on the posterior probability maps

$$\pi_e : \arg \max_a F_p \circ (C_p \circ A_p) \quad (1)$$

where \circ is the Hadamard product, also known as the entrywise product.

Coordination policy (coordinator). To coordinate pushing and grasping, we propose a classifier-based policy, denoted as π_c . The binary classifier takes as input domain knowledge, as well as maximum push Q value q_p and maximum grasp Q value q_g . Specifically, the domain knowledge is target border occupancy ratio $r_b = \frac{o_b}{\sum m_t}$, target border occupancy norm $n_b = \frac{o_b}{\sum m_t}$ (border occupancy value o_b , target border m_b , target mask m_t defined in Sec. V-B) and the number of consecutive grasping failures c_g . This knowledge provides direct information for the classifier to make action decisions: 1) r_b and n_b are indicators of the clutteredness around the target but hard for the networks to learn directly and 2) c_g records the history of failed grasps.

We train the classifier to determine whether to push or grasp under the given state. The action with maximum corresponding Q value is executed. We name the classifier as action classifier f_a (i.e., classify the state into grasp-favored or not) and the coordinator is formulated as

$$y = f_a(q_p, q_g, r_b, n_b, c_g) \quad (2)$$

$$\pi_c : \arg \max_a Q_y \quad (3)$$

where f_a is a function approximator composed of three fully connected layers with batch normalization [19] and ReLU [20]. It learns to signify the influential variables through its weights and drop unimportant factors by ReLU.

C. Policy Execution

Algorithm 1 summarizes the details of policy execution to grasping the invisible target. The algorithm is repeated until the robot grasps the target or exceeds the maximum number of motions. For each iteration, one of the two policies (explorer or coordinator) is effective upon the existence of target mask M .

V. LEARNING TARGET-ORIENTED MOTION CRITIC

With the setup of visible targets, our approach learns the target-oriented motion critic, which is used in both explorer π_e and coordinator π_c . In what follows, we discuss the target-oriented motion critic and the reward scheme for training.

Algorithm 1 Critic-Policy to Grasping the Invisible

Input: RGB-D image I

Output: action decision a_t at time t

```

1:  $M \leftarrow \text{ObjectSegmentation}(I)$ 
2: if  $M = \emptyset$  then ▷ exploration subtask
3:    $M \leftarrow \text{AllOnesMask}()$ 
4:    $s_t \leftarrow \text{HeightmapProjection}(I, M)$ 
5:    $C_p \leftarrow \text{ClutterPrior}(s_t)$ 
6:    $A_p \leftarrow \phi_p(s_t)$ 
7:    $F_p \leftarrow \text{FailureLikelihood}(\cdot)$ 
8:    $a_t \leftarrow \arg \max_a F_p \circ (C_p \circ A_p)$  ▷ explorer
9: else ▷ coordination subtask
10:   $s_t \leftarrow \text{HeightmapProjection}(I, M)$ 
11:   $Q_p \leftarrow \phi_p(s_t), Q_g \leftarrow \phi_g(s_t)$ 
12:   $y \leftarrow f_a(\max Q_p, \max Q_g, r_b, n_b, c_g)$ 
13:   $a_t \leftarrow \arg \max_a Q_y$  ▷ coordinator
14: end if

```

A. Motion critic

As discussed in Sec. I, when the visible target is within clutter, a sequence of target-oriented pushing and grasping actions should be applied to free the space around the target, and finally grasp it. With the RGB-D-mask heightmaps of the scene, it is sufficient to estimate the state of the current workspace. More specifically, a reasonable estimate of the target pose can be derived. Thus, we model the target-oriented manipulation problem as a discrete Markov Decision Process (MDP). For convenience of notations, we treat the heightmaps equivalent to the state, i.e., $s_t = (c_t, d_t, m_t)$.

In MDP, the robot performs an action a_t in state s_t , then transitions to state s_{t+1} , and receives the corresponding reward $R(s_t, a_t, s_{t+1})$. The goal of our critic is to learn the action-value function $Q_{\{p,g\}}^{\pi}(s, a)$ which predicts the expected return for pushing or grasping action a in state s under a policy π .

The motion critic is represented with a fully convolutional encoder-decoder network [21], and it outputs pixel-wise Q maps. The RGB, depth, and mask heightmaps are fed into the corresponding feature extractor (2-layer residual [22] network) for feature extraction. The extracted features are concatenated as the input to a DenseNet-121 [23] pre-trained on ImageNet [24], to produce the motion-agnostic features [25]. Then push network ϕ_p and grasp network ϕ_g take the features as input to predict push maps Q_p and grasp maps Q_g , respectively. Network ϕ_p and ϕ_g have the same architecture, a 3-layer residual network followed by bilinear upsampling.

B. Reward Function

We divide our reward scheme for the critic into two stages: the pre-action stage and post-action stage. For one action, either zero reward or maximum stage reward is assigned.

We give an example of the reward scheme in Fig. 4. We assign pre-action reward $R_p(s_t, s_{t+1}) = 0.25$ if the intended pushing vector passes mask m_t . Then post-action reward $R_p(s_t, s_{t+1}) = 0.5$ is given for pushes that make more space around the target for future grasping. To detect the space

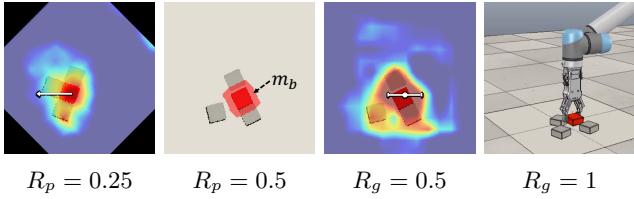


Fig. 4: **Example of our reward scheme.** One push and grasp are executed consecutively for the target (the red cuboid), and we assign the pre-action and post-action rewards to the corresponding actions.

increase, we first dilate around m_t to construct the mask of target border m_b (shown as the mask of light red color). And the space increase is confirmed if border occupancy value o_b (defined as the number of pixels in m_b with height above the ground) decreased by some threshold. In the example, the pushing action frees spaces around the target, and thus the reward of 0.5 is given.

Similarly, we assign pre-action reward $R_g(s_t, s_{t+1}) = 0.5$ for those grasps with an intended grasping position in m_t , and post-action reward $R_g(s_t, s_{t+1}) = 1$ if the target is successfully grasped.

VI. TRAINING IN SELF-SUPERVISION

We train the motion critic and action classifier f_a of policy π_c by self-supervision in simulation.

A. Loss Function

The motion critic is trained by minimizing temporal difference error δ_t as

$$\delta_t = Q(\theta_t; s_t, a_t) - (R_{at}(s_t, s_{t+1}) + \gamma \max_a Q(\theta_t^-; s_{t+1}, a)) \quad (4)$$

via the Huber loss

$$\mathcal{L}_\delta = \begin{cases} \frac{1}{2}\delta_t^2, & \text{if } |\delta_t| \leq 1 \\ |\delta_t| - \frac{1}{2}, & \text{otherwise} \end{cases} \quad (5)$$

where θ_t are the parameters of the critic networks at time t , and the target network parameters θ_t^- are held fixed between iterations. At time t , we pass gradients only through the single pixel on which the motion primitive was executed while all other pixels backpropagate with 0 loss.

The coordinator is trained using the binary cross-entropy loss

$$\mathcal{L}_y = -(\bar{y} \log y + (1 - \bar{y}) \log(1 - y)) \quad (6)$$

where y is the predication from action classifier f_a and \bar{y} is the ground-truth label.

B. Data Collection and Training

We collect the data with the following procedure: n target candidates (i.e., detectable by the semantic segmentation module) and m basic objects are randomly selected and dropped into the workspace in front of the robot. The robot needs to grasp one randomly appointed target via a sequence of

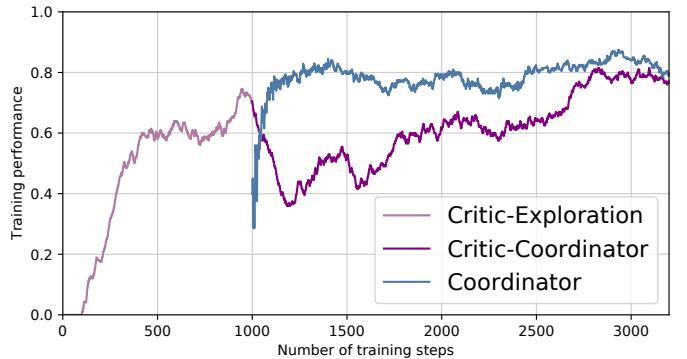


Fig. 5: **Training performance.** The purple lines indicate the target-oriented grasping success rate of the critic-policy, and the blue line indicates the training accuracy of action classifier f_a in policy π_c over training steps.

pushing and grasping. Once the target is successfully grasped, the new target is appointed for the next trial. The objects are again randomly dropped if the workspace is void of target candidates. We save heightmaps, executed actions, and execution results for training the critic. In addition, labels are automatically generated for grasping actions to train policy π_c (see equation 6). The label \bar{y} is assigned as 1 if the target is grasped or 0 if the grasping position is within mask m_t but results in a grasping failure (this might indicate dense clutter around the target).

Multi-stage training is utilized. At the first stage, we only train the critic to reach a good initialization; the robot follows under an ϵ -greedy policy π_ϵ . We set $m = 3$ to ease the learning of target-oriented pushing and grasping. Then m increases to be 8, and the policy switches to be randomly initialized coordinator π_c to learn coordinated decision making in structured dense clutter. In the meantime, the critic is still under training and expected to accommodate π_c , i.e., finetuned from Q^{π_ϵ} to Q^{π_c} . We summarize the training process in Appendix IX-C.

As shown in Fig. 5, only the critic is trained at the first stage (first 1000 iterations in our experiments) under π_ϵ exploration and reaches a high target-oriented grasping success rate (defined as $\frac{\# \text{successful target-oriented grasps}}{\# \text{total motions (pushes and grasps)}}$). Then we replace the policy to be coordinator π_c and start to train the coordinator to increase its prediction accuracy gradually. Note that critic-coordinator finally achieves a higher target-oriented grasping success rate in even more cluttered scenes.

VII. EXPERIMENTS

We train the system in simulation where all states are known. We executed a set of ablation studies for explorer π_e and comparative experiments for coordinator π_c . The goals of the experiments are 1) to show the importance of the extra domain knowledge in the policies, 2) to understand the limitation of each submodule in explorer π_e and the advantages and robustness of π_e and 3) to demonstrate that our coordinator π_c can coordinate target-oriented pushing and grasping in structured clutter. The simulation environment and the robot are kept the same with [3] for a fair comparison. We also run experiments on a real robot to show the performance of our

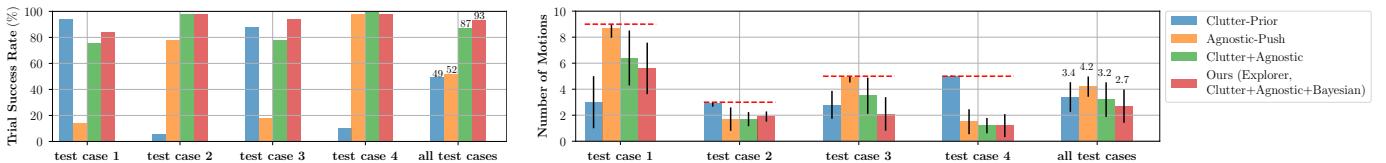


Fig. 6: **Performance in exploration subtask.** The task success rate (left) and the number of motions (right) of four approaches on the four test cases of complete occlusion. The plot shows the effectiveness of our approach, **Explorer**, which achieves a task success rate of 93% with 2.7 motions on average. The red dotted line is the number limitation of motions.

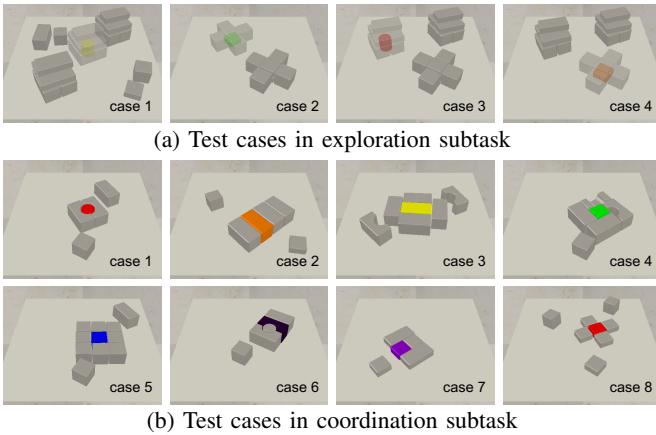


Fig. 7: **Test cases in simulation.** (a) shows four test cases in the exploration subtask, where the target is invisible, and (b) shows eight challenging arrangements where the coordination between pushing and grasping is required. The target is the colored object.

system on the “grasping the invisible” problem. The testing objects are either toy blocks or daily objects of a similar shape to the training objects. Although it would be of interest to see how our system generalizes to quite different testing objects, the main goal of the experiments in this paper is to show the generalization of our approach to challenging object arrangements never seen during training.

A. Exploration Subtask

To validate our approach, we run an ablation study for which our explorer is compared with the following methods in the exploration subtask: 1) **Clutter-Prior** builds the probability maps C_p for potential pushing actions based on the depth heightmap [26]. The heightmap is first translated along one fixed axis for 25 pixels (approximately twice the width of the closed gripper), then the pixel with enough depth difference between original and translated heightmap is recorded as 1 otherwise 0. This binary map is filtered with a 25×25 all-ones kernel to get a pixel-wise probability map. Like Q maps, the heightmap is also rotated by N orientations to construct N probability maps. By detecting varying heights, C_p encodes prior knowledge of edges of clutter. 2) **Agnostic-Push** utilizes target-agnostic pushing A_p from ϕ_p taking in RGB-D heightmaps c_t , d_t and all-ones mask heightmap. **Agnostic-Push** lacks the sense of clutter and could push a well-isolated object, as ϕ_p is trained to make changes around the visible target by pushing. 3) **Clutter+Agnostic** is the Hadamard product of **Clutter-Prior** and **Agnostic-Push**, i.e., $C_p \circ A_p$. By balancing between

TABLE I: Average Performance in the Exploration Subtask

Method	Task Success Rate (%)	Number of Motions
Clutter-Prior	49.5	3.43 ± 1.14
Agnostic-Push	52	4.20 ± 0.78
Clutter+Agnostic	87.5	3.20 ± 1.33
Our Explorer	93.5	2.70 ± 1.28

clutter prior and target-agnostic pushing, the robot achieves significant performance gain. 4) **Clutter+Agnostic+Bayesian** is our explorer. It adds pushing failure likelihood F_p on **Clutter+Agnostic** to reduce the probability around the three most recent locations of failed pushes, which improves the robustness of the system by avoiding getting stuck.

The test cases are shown in Fig. 7a. In each test case, the maximum number of pushes is $n_{\text{pushes}} = 2 * n_{\text{cluster}} - 1$ where n_{cluster} is the number of object clusters in the workspace. The robot is tasked to find the target via explorational pushes and allowed to stop if the target is found. We execute 50 runs on each test case and report the task success rate and the number of motions. Fig. 6 presents the performance of the above methods in the exploration subtask. Both **Clutter-Prior** and **Agnostic-Push** perform poorly in specific cases. For instance, **Clutter-Prior** is discouraged in either the equal height case (case 2) or the taller pyramid-like clutter case (case 4). **Agnostic-Push** is prone to push every object in the workspace until it finds the target buried under the pyramid-like shape in the test case 1 and 3, and hence it necessitates a large number of motions. By balancing between clutter prior and target-agnostic pushing, **Clutter+Agnostic** works consistently well across all test cases. Our explorer **Clutter+Agnostic+Bayesian** further improves the performance of **Clutter+Agnostic** in terms of the task success rate with a fewer number of motions. As shown in Table I, our explorer **Clutter+Agnostic+Bayesian** outperforms **Clutter+Agnostic** by 6% in terms of the task success rate and requires about 0.5 fewer average number of motions compared to **Clutter+Agnostic**.

B. Coordination Subtask

We compare the picking performance of our coordinator with the following baseline approaches: 1) **RAND** randomly chooses one of motion primitives and samples motion angle from the N angles and motion position in target mask m_t . 2) **Mask-VPG** is an extension of VPG [3] by incorporating the mask m_t as post-processing. VPG produces Q maps for target-agnostic tasks, and we filter the push maps by a dilated target mask and the grasp maps by the target mask. The action with the highest Q value is executed. 3) **Border-Heuristic**

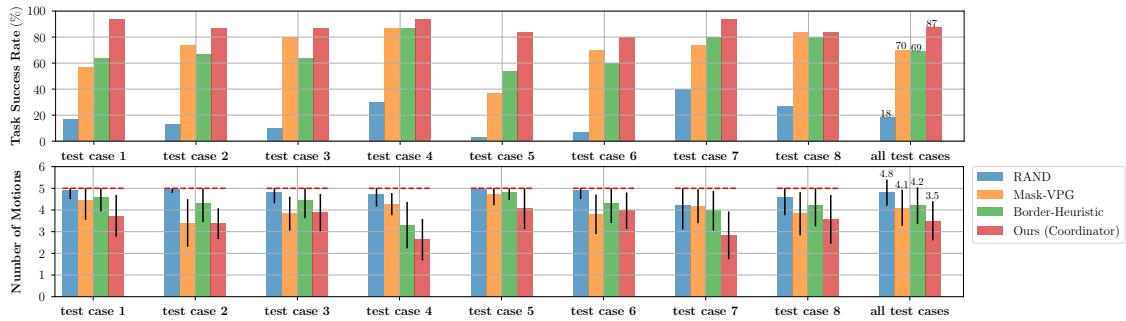


Fig. 8: **Performance in coordination subtask.** The task success rate (top) and the number of motions (bottom) of four approaches on the 8 test cases of challenging arrangements. The plot shows the effectiveness of our approach, **Coordinator**, which achieves a task success rate of 87% with 3.3 motions on average.

TABLE II: Average Performance in the Coordination Subtask

Method	Task Success Rate (%)	Number of Motions
RAND	18.3	4.77 ± 0.60
Mask-VPG	70.0	4.06 ± 0.83
Border-Heuristic	69.2	4.25 ± 0.85
Our Coordinator	87.5	3.51 ± 0.90

TABLE III: Real-robot Results on “Grasping the Invisible”

Method	Task Success Rate (%)	Number of Total Motions
VPG	32.5	14.4
Mask-VPG	67.5	11.6
Ours	85.0	9.8

determines whether to push or grasp by a heuristic policy and asks the motion critic to execute the action. The policy is ϵ -greedy, and higher ϵ indicates a higher pushing exploration rate. The base rate is $\epsilon_0 = 0.5$, and we adjust its value to accommodate maximum Q values q_p , q_g and the domain knowledge r_b , n_b , c_g defined in Sec. IV-B. In brief ϵ is advanced with the decrease of $q_g - q_p$, the increase of r_b and n_b , and the growth of c_g . Note that the method takes the same input with our coordinator, but the difference is the subtask policy. Please refer to Appendix IX-E for more details.

We evaluate the methods on eight challenging test cases with adversarial structures shown in Fig. 7b. In each test case, the maximum number of motions is 5, and the robot is tasked to grasp the target in clutter. We execute 30 runs on each test case and report the task success rate and the number of motions in Fig. 8. Our approach **Coordinator** outperforms the other approaches in terms of both the task success rate and the number of motions. Overall **Coordinator** achieves 87.5% task success rate in the 8 challenging arrangements; **RAND** shows about 18% chance of task success rate; the performance of **Mask-VPG** and **Border-Heuristic** are similar, while **Mask-VPG** shows slightly superior performance. **Mask-VPG** is originally designed for target-agnostic tasks, and thus it lacks the reasoning about the target and surrounding objects. Though **Border-Heuristic** has the same input with **Coordinator**, the hard-coded heuristic limits the coordination of target-oriented pushing and grasping. The comparison between **Border-Heuristic** and **Coordinator** shows the effectiveness of π_c for coordination. As shown in Table II, **Coordinator** increases by more than 17% in terms of the task success rate and requires about 0.6 fewer average number of motions compared with **Mask-VPG** and **Border-Heuristic**.

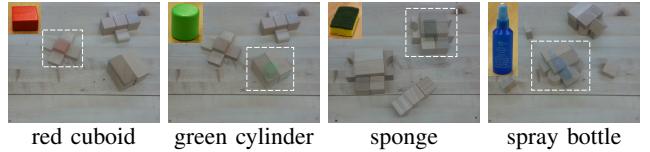


Fig. 9: **Test cases on the real robot.** The invisible target is either a toy block or a novel object never seen in training.

C. Real-robot Experiments

We evaluate our system on the “grasping the invisible” problem with a Franka EMIKA Panda robot using the model trained in simulation². For the comparison baselines, we use the available model trained with a robot system in the real world [3]. Over 4 test cases, as shown in Fig. 9, our approach and two baselines, **VPG** and **Mask-VPG**, are tested. Successful target grasping is manually checked for **VPG** as it is target-agnostic. When searching for the target, **Mask-VPG** works as **VPG** (i.e., no mask post-processing) except that only pushing is enabled for a fair comparison.

We run on each test case for 10 runs, and the maximum number of motions for each run is 15. The robot is tasked to find the initially invisible target as well as grasp it in challenging clutter. Table III reports the task success rate and the number of total motions of the three methods. Overall, our approach trained in simulation outperforms the domain-adapted baselines and achieves a task success rate of 85%. The results show our system is capable of generalizing to new environments, sets of objects of a similar shape to training objects, and adversarial arrangements. **VPG** shows a task success rate of only 32.5% with a high average number of motions. Target-agnostic **VPG** tends to prioritize grasping easily graspable objects while the invisible target is buried in heavy clutter. **Mask-VPG** is advanced by reducing the action field by the target mask, and its task success rate improves to be 67.5% with 11.6 average number of motions. In contrast to **Mask-VPG**, our approach outperforms by 17.5% in terms of the task success rate and requires about 1.8 fewer average number of motions. Through experiments, we find that **Mask-VPG** tends to be less robust to noise in real settings and

²We don’t fine-tune the model on the real robot. To collect data in the real world, the robot could hold a grasped object up to the camera and use an additional classifier to check grasping results [5].

lacks the capability of superior coordination that our approach demonstrates.

VIII. CONCLUSIONS

In this work, we presented the “grasping the invisible” problem and proposed a deep learning approach in a critic-policy format. The learning models of our approach were trained by self-supervision in simulation. We evaluate the system in both simulated and real settings. Our approach shows a 93% and 87% task success rate on the two subtasks in simulation and an 85% task success rate in the real robot experiments, which outperforms the other compared approaches by large margins. The evaluation results with the real robot show the generalization capability of our approach. The learned model in simulation was reliably transferred in the real setting and even generalizes to novel objects.

REFERENCES

- [1] A. Eitel, N. Hauff, and W. Burgard, “Learning to singulate objects using a push proposal network,” in *Proc. of the International Symposium on Robotics Research (ISRR)*, 2017.
- [2] C. Choi, W. Schwarting, J. DelPreto, and D. Rus, “Learning object grasping for soft robot hands,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2370–2377, 2018.
- [3] 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,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4238–4245.
- [4] E. Jang, C. Devin, V. Vanhoucke, and S. Levine, “Grasp2vec: Learning object representations from self-supervised grasping,” in *Conference on Robot Learning*, 2018, pp. 99–112.
- [5] K. Fang, Y. Bai, S. Hinterstoisser, S. Savarese, and M. Kalakrishnan, “Multi-task domain adaptation for deep learning of instance grasping from simulation,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3516–3523.
- [6] J. Bohg, A. Morales, T. Asfour, and D. Kragic, “Data-driven grasp synthesis survey,” *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2013.
- [7] A. Sahbani, S. El-Khoury, and P. Bidaud, “An overview of 3d object grasp synthesis algorithms,” *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 326–336, 2012.
- [8] 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,” *Robotics: Science and Systems (RSS)*, 2017.
- [9] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke *et al.*, “Scalable deep reinforcement learning for vision-based robotic manipulation,” in *Conference on Robot Learning*, 2018, pp. 651–673.
- [10] M. R. Dogar and S. S. Srinivasa, “A planning framework for non-prehensile manipulation under clutter and uncertainty,” *Autonomous Robots*, vol. 33, no. 3, pp. 217–236, 2012.
- [11] A. Cosgun, T. Hermans, V. Emeli, and M. Stilman, “Push planning for object placement on cluttered table surfaces,” in *2011 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2011, pp. 4627–4632.
- [12] T. Hermans, J. M. Rehg, and A. Bobick, “Guided pushing for object singulation,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4783–4790.
- [13] L. Chang, J. R. Smith, and D. Fox, “Interactive singulation of objects from a pile,” in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 3875–3882.
- [14] M. Danielczuk, J. Mahler, C. Correa, and K. Goldberg, “Linear push policies to increase grasp access for robot bin picking,” in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2018, pp. 1249–1256.
- [15] K. Marios and M. Sotiris, “Robust object grasping in clutter via singulation,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 1596–1600.
- [16] A. Boularias, J. A. Bagnell, and A. Stentz, “Learning to manipulate unknown objects in clutter by reinforcement,” in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.
- [17] E. Jang, S. Vijayanarasimhan, P. Pastor, J. Ibarz, and S. Levine, “End-to-end learning of semantic grasping,” in *Conference on Robot Learning*, 2017, pp. 119–132.
- [18] M. Danielczuk, A. Kurenkov, A. Balakrishna, M. Matl, D. Wang, R. Martn-Martn, A. Garg, S. Savarese, and K. Goldberg, “Mechanical search: Multi-step retrieval of a target object occluded by clutter,” in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 1614–1621.
- [19] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International Conference on Machine Learning*, 2015, pp. 448–456.
- [20] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.
- [21] V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.
- [22] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [23] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.
- [24] 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*. Ieee, 2009, pp. 248–255.
- [25] L. Pinto and A. Gupta, “Learning to push by grasping: Using multiple tasks for effective learning,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2161–2168.
- [26] A. Zeng, “visual-pushing-grasping,” 2018. [Online]. Available: <https://github.com/andyzeng/visual-pushing-grasping>
- [27] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, “Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes,” *Robotics: Science and Systems (RSS)*, 2018.
- [28] A. S. Periyasamy, M. Schwarz, and S. Behnke, “Robust 6d object pose estimation in cluttered scenes using semantic segmentation and pose regression networks,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6660–6666.
- [29] C. Wang, D. Xu, Y. Zhu, R. Martn-Martn, C. Lu, L. Fei-Fei, and S. Savarese, “Densefusion: 6d object pose estimation by iterative dense fusion,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3343–3352.
- [30] V. Nekrasov, C. Shen, and I. Reid, “Light-weight refinenet for real-time semantic segmentation,” *arXiv preprint arXiv:1810.03272*, 2018.
- [31] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized experience replay,” *arXiv preprint arXiv:1511.05952*, 2015.
- [32] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, “Hindsight experience replay,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5048–5058.

IX. APPENDIX

A. Action Policy of Mechanical Search

We summarize the action policy of Mechanical Search [18] for comparison purposes. The action policy of [18] takes as input the masks of objects in the bin and selects the action to execute. The policy is composed of an action selection method and an action execution criterion. The action selection method (e.g., largest visible object first) is responsible for keeping a priority list. And the action execution criterion iterates the objects in the list to make the action decision. The action execution criterion can be summarized as Algorithm 2. For every object in the priority list, Algorithm 2 selects an action or reports a failure until the target is retrieved.

The formulation of a POMDP is given in [18] for the target retrieval problem. However, due to the computation difficulty, Algorithm 2 is proposed to apply action decision making heuristically instead of solving the POMDP problem in the reinforcement learning framework.

In addition, pushing is of low priority in Algorithm 2, i.e., pushing is only considered without a satisfactory grasping being found. In the experiments, [18] shows pushing makes up only 5% of executed actions. More effective push primitives are listed as one of the future work in [18].

Algorithm 2 Action Execution Criterion

Input: object mask o
Output: selected action a or termination
Notations: threshold t , action a , action quality q

- 1: **if** o is the search target or pushing is disabled **then**
- 2: $t_{\text{grasp}} \leftarrow t_{\text{low}}$
- 3: **else**
- 4: $t_{\text{grasp}} \leftarrow t_{\text{high}}$
- 5: **end if**
- 6: $a, q \leftarrow \text{GraspingModules}(o)$
- 7: **if** $q > t_{\text{grasp}}$ **then**
- 8: execute grasp action a
- 9: **else**
- 10: **if** pushing enabled **then**
- 11: $a, q \leftarrow \text{PushingModule}(o)$
- 12: **if** valid a not found or pushed three times **then**
- 13: termination with a failure
- 14: **end if**
- 15: **else**
- 16: termination with a failure
- 17: **end if**
- 18: **end if**

B. Pre-trained Perception Module

Semantic segmentation is widely used in 6D object pose estimation in cluttered scenes [27], [28], [29] and proves to be very robust to occlusions between objects. We choose Light-Weight RefineNet [30] as our perception module and pre-train it on the augmented data. Inspired by [28], we synthetically generated the training dataset covering all target candidates, rich pose variations of the objects, and occlusions



Fig. 10: **Example of semantic image segmentation.** The background is visualized in white color for best visualization.

with a handful of labeled data. The pre-trained model robustly segments the cluttered scenes, as shown in Fig 10.

C. Additional Training Details

We save states, actions and rewards into experience replay buffer B_c and train the critic with prioritized experience replay [31]. To deal with sparse rewards for target-oriented grasping (especially at early stages), the hindsight experience replay technique [32] is used. If a non-target object is grasped at time t , we save executed action a_t , states, mask of the grasped object m'_t and posthoc labeled reward $R'_{a_t}((c_t, d_t, m'_t), s_{t+1}) = 1$ for further experience replay training. To train action classifier f_a , we save Q values and domain knowledge, and the generated label into bounded buffer B_π (a cyclic buffer of bounded size). The training process is summarized in Algorithm 3.

D. Additional Details of Pushing Failure Likelihood

We first construct a 2D Gaussian function G with the peak at the center of the workspace. The standard deviation of G is set to cover the pushing length. Then we apply a linear transform on G to get function G'

$$G' = 1 - \frac{(1-s)G}{\max(G)} \quad (7)$$

where s is the scale parameter and is set to be 0.75 in our experiments. We shift G' to the location of a failed action to reduce the probability around the failed locations. The shifting is repeated for the three most recent locations of failed pushes, and we multiply the shifted functions to construct pushing failure likelihood F_p . In the implementation, we use a deque to store the most recent shifted functions for performance.

E. Additional Details of Border-Heuristic

Border-Heuristic determines whether to push or grasp by a heuristic. Once the decision is made, pushing or grasping is executed by target-oriented motion critic. We include Border-Heuristic as a baseline to show how much a learned classifier (in our coordination policy) outperforms a hand-engineered heuristic. **Border-Heuristic** is an exploration-based heuristic, and the pushing exploration rate is defined as

$$f_e(\cdot) = \min(1.0, \max(0.0, 0.5 - \alpha \frac{q_g - q_p}{(c_g + 1)(r_b + n_b)})) \quad (8)$$

where $\alpha = 1e^{-2}$ is the hand-engineered parameter in our experiments. The algorithm of **Border-Heuristic** is summarized in Algorithm 4.

Algorithm 3 Training Algorithm

Objects: critic C , policy π_c (with classifier f_a)
Initialize experience replay buffer B_c for C
Initialize bounded buffer B_π for π_c
Notations: iteration i , ϵ -greedy policy π_ϵ , target T , image I , mask M , state s , push or grasp network ϕ , maximum Q value q , domain knowledge $D = (r_b, n_b, c_g)$

```

1:  $i \leftarrow 0$ 
2: while True do
3:   reset the simulation and randomly drop objects
4:   randomly select a visible target  $T$ 
5:   while  $T$  not grasped do
6:      $i \leftarrow i + 1$ 
7:      $M \leftarrow \text{ObjectSegmentation}(I, T)$ 
8:      $s_t \leftarrow \text{HeightmapProjection}(I, M)$ 
9:      $Q_p \leftarrow \phi_p(s_t), Q_g \leftarrow \phi_g(s_t)$ 
10:    if  $i < 1000$  then
11:      execute action  $a_t \leftarrow \pi_\epsilon(Q_p, Q_g)$ 
12:    else
13:      execute action  $a_t \leftarrow \pi_c(Q_p, Q_g, D)$ 
14:      label  $\bar{y}$  according to grasping results
15:      save  $(q_p, q_g, D)$  and  $\bar{y}$  into  $B_\pi$ 
16:      sample a batch from  $B_\pi$  to train  $f_a$ 
17:    end if
18:     $R_t \leftarrow \text{RewardFunction}(\cdot)$ 
19:    save  $s_t, a_t, R_t$  into  $B_c$ 
20:    if non-target object grasped then
21:      save extra mask  $m'_t$  and reward  $R'_t$  into  $B_c$ 
22:    end if
23:    sample data from  $B_c$  to train  $C$ 
24:  end while
25: end while

```

Algorithm 4 Border-Heuristic

Input: push and grasp Q maps, target border occupancy ratio r_b , target border occupancy norm n_b , the number of consecutive grasping failures c_g

Output: motion type $y \in \{p, g\}$ and action decision a_t

```

1:  $q_p \leftarrow \max Q_p, q_g \leftarrow \max Q_g$ 
2:  $\epsilon_p = f_e(q_p, q_g, r_b, n_b, c_g)$ 
3: if random()  $< \epsilon_p$  then
4:    $y \leftarrow p$ 
5: end if
6: if  $y = g$  and  $c_g \neq 0$  then
7:   if  $c_g < 2$  then
8:     if random()  $< r_b$  then
9:        $y \leftarrow p$ 
10:    end if
11:  else
12:     $y \leftarrow p$ 
13:  end if
14: end if
15:  $a_t \leftarrow \max_a Q_y$ 

```

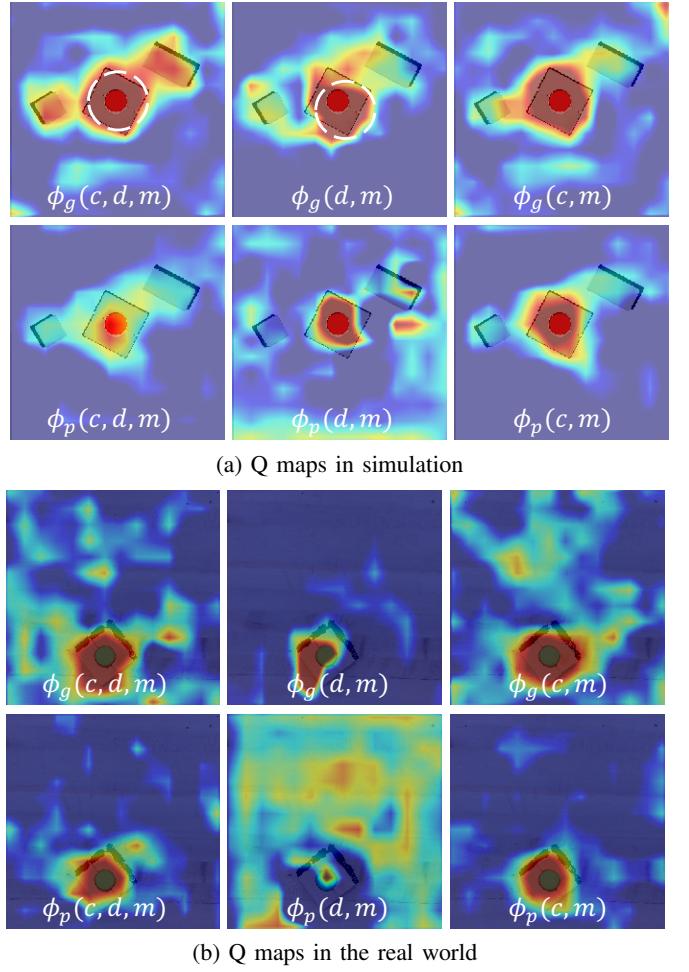


Fig. 11: **Testing with different inputs and in different domains.** The maps are generated by our pre-trained model.

F. Additional Testing

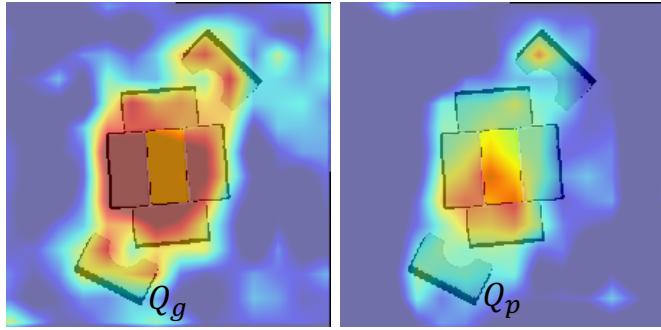
We test our model with different inputs and in different domains. We visualize the output of the motion critic (composed of push network ϕ_p and grasp network ϕ_g) with different combinations of color, depth, and mask input (denoted as c , d and m) in Fig. 11. The Q maps are produced by one pre-trained model, and the corresponding CNN parameters are set to zero for the disabled channel.

We observe that both color and depth channel affect the output. There are nontrivial changes if we disable either one of the channels. For example, we notice that, in simulation, the Q map shifts away from the target if we disable the color channel, as shown with the white dashed circle in Fig. 11.

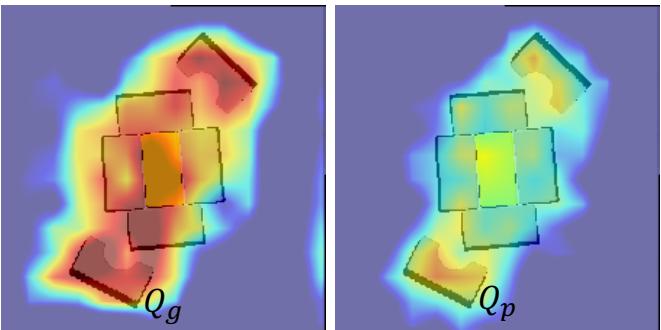
Our model generalizes to the real world since $\phi(c, d, m)$ shares some similar patterns when transferring from simulation to the real world. For example, $\phi_g(c, d, m)$ and $\phi_p(c, d, m)$ have higher Q values around the target in both simulation and the real world.

G. Additional Training

To further investigate the importance of the color channel in our approach, we train a depth+mask variant of our approach. We make minimal modifications on the motion critic to accept



(a) Q maps of Ours (color+depth+mask)



(b) Q maps of Depth+Mask

Fig. 12: **Comparison of Q maps.** The push and grasp maps are for one test case.

only the depth and the target mask while keeping all other parts unchanged. We evaluate **Depth+Mask** and report the results in Table IV. Our approach outperforms **Depth+Mask**.

TABLE IV: Average Performance of depth+mask

Method	Success Rate (%)	Number of Motions
Depth+Mask in exploration	87.0	3.30 ± 1.34
Ours in exploration	93.5	2.70 ± 1.28
Depth+Mask in coordination	78.7	4.06 ± 0.29
Ours in coordination	87.5	3.51 ± 0.90

We conjecture that the color input helps the networks to distinguish the target from surrounding objects, especially in generalization. As shown in Fig. 12, when we test **Depth+Mask** on the challenging arrangements never seen during training, the Q maps shift from the target compared to the Q maps of our approach. This observation is consistent with Appendix IX-F.