Active Pushing for Better Grasping in Dense Clutter with Deep Reinforcement Learning

Ning Lu[1,2], Tao Lu[1], Yinghao Cai[1], Shuo Wang[1]
1.Research Center on Intelligent Robotic Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China
2.School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
{luning2018, tao.lu, yinghao.cai, shuo.wang}@ia.ac.cn

2020 Chinese Automation Congress (CAC) | 978-1-7281-7687-1/20/$31.00 ©2020 IEEE | DOI: 10.1109/CAC51589.2020.9327270

***Abstract*—Robotic grasping in unstructured dense clutter remains a challenging task and has always been a key research direction in the field of robotics. In this paper, we propose a novel robotic grasping system that could use the synergies between pushing and grasping actions to automatically grasp the objects in dense clutter. Our method involves using fully convolutional action-value functions (FCAVF) to map from visual observations to two action-value tables in a Q-learning framework. These two value tables infer the utility of pushing and grasping actions, and the highest value with the corresponding location and orientation means the best place to execute action for the end effector. For better grasping, we introduce an active pushing mechanism based on a new metric, called Dispersion Degree, which describes how spread out the objects are in the environment. Then we design a coordination mechanism to apply the synergies of different actions based on the action-values and dispersion degree of the objects and make the grasps more effective. Experimental results show that our proposed robotic grasping system can greatly improve the robotic grasping success rate in dense clutter and also has the capability to be generalized to the new scenarios.**

Keywords—robotic grasp; active push; synergies; deep reinforcement learning

I. INTRODUCTION

Robotic grasping research has always been a key research direction in the field of robotics. And the related achievements have been widely used in industry, exploration, service, military, etc. [1]. While, as the complexity of environments increases, grasping different targets in a cluttered or even dense environment is more and more challenging.

A lot of initial work focuses on planning to grasp a known object and assume knowledge of object contact points, dynamics, poses and shapes [2][3] or maximizing the affordance metrics through the data-driven method to find the grasp locations [4][5]. However, in the unstructured environment, information is rarely known for novel objects, and clutter also poses great difficulty for reaching the planned grasps because of the occlusion between objects.

For human grasping objects in dense clutter, the list of primitive actions used is not limited to grasp, but also includes non-prehensile actions, such as pushing, pulling, and dumping. We pick up, push, slide and swipe with our hands and arms to reorganize the clutter surrounding our grasping tasks. Inspired by this, a branch of recent grasping research has been trying to change or explore the environment through other auxiliary actions, such as poking or pushing, so as to facilitate grasping objects in the environment [6][7][8][9]. These methods show great promise for improving the grasp success rate with the help of other actions. While there are still some problems when applying the synergies of different actions, such as useless pushing action in the area with no objects [9] or low efficiency of grasping [8] et al.

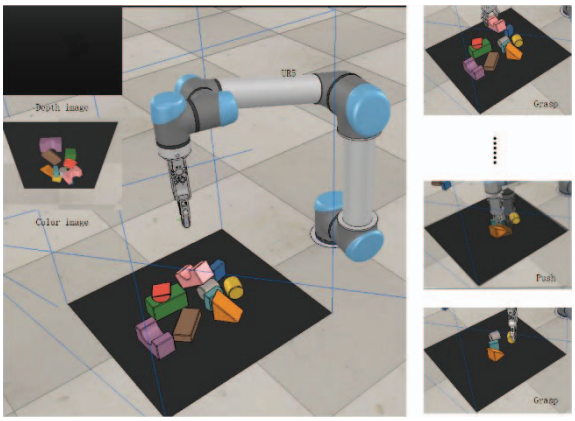


Fig. 1. Robotic grasping system. The system could automatically grasp the objects in dense clutter with the synergies between pushing and grasping actions.

In this work, we propose a novel robotic grasping system (as shown in Fig. 1) that could automatically grasp the objects in dense clutter with the synergies between pushing and grasping actions. The main contributions of our work are summarized as follows:

1) We propose an active pushing mechanism based on a new metric, named Dispersion Degree (DD). DD is designed to describe how spread out the objects are in the scene and is used to calculate the rewards for the pushing network. With the new metric, the pushing network could be trained to guide the robot to push the objects apart actively and purposefully and reduce the rate of useless pushes.

2) We introduce a grasping system framework for the robot to perform the synergies between grasping and pushing actions with deep reinforcement learning method. Based on designed coordination mechanism, the robot could properly select pushes or grasps and avoid the influence between the different kinds of actions. That greatly improves the grasp efficiency, and experiments also show the excellent performance of the developed system when grasping objects in dense clutter.

The rest of the article is organized as follows. Section II introduces related work. Section III briefly presents the preliminaries relative to our work. Section IV overviews our robotic grasping system. Section V addresses the proposed method in detail. Section VI describes the training process and extensive experiments, which prove the excellent performance of our method. Section VII summarizes conclusions and future work.

Corresponding author: Tao Lu{tao.lu@ia.ac.cn}

II. RELATED WORK

Grasping is an important research direction in robotics. Most of the initial work in this field focused on 3D reasoning and analytical methods to predict grasp configurations and locations [10][11]. Ponce et al. [12] present a set of linear inequalities based on grasp contact positions and convert the finding force-closure grasps problem to the projecting polyhedron problem on linear subspace. Mirtich et al. [13] simplify the 3D object grasp to a 2D one through the calculation of position relationship between the three contact points of three fingers and the object. Zeng et al. [14] present a 3D-Match model. It can learn a local volumetric patch descriptor to establish the correspondences between partial 3D data. Zeng et al. [15] segment and label objects with a fully convolutional neural network in multiple views of the scene. In order to get the 6D object pose for grasping, they fit the segmentation results to the pre-scanned 3D object model. The analytical methods make grasping feasible, while it requires physical modeling of the grasping objects, which has high computational complexity and is difficult to be applied in unstructured real-world environments.

More recent data-driven methods make it possible to train model-agnostic grasping policies [16][17]. These methods utilize learned visual features to detect grasps and do not use object-specific knowledge (i.e., pose, shape, contact points). Jiang et al. [16] design the convolutional neural network for learning 6D grasping positions of objects and also give out of a series of object dataset and label methods which could be reused for other researchers. Pinto et al. [17] present a self-supervised algorithm instead of using manually labeled grasp datasets. And the network is trained to predict grasp locations with a large-scale datasets of 50K tries over 700 robot hours. Jang et al. [18] combine spatial and semantic reasoning into a single neural network policy and propose a learning-based system for visual robotic grasping to complete the task of semantic robotic grasping. Gualtieri et al. [19] abstract the robotic pick and place task as a deep RL problem, in which the target reach poses for the hand are actions and the history of such reaches are states. The authors show that this method can solve a class of challenging pick-place and regrasping problems of the objects to be processed, which are unknown in geometry. Compared with the analytical methods, the data-driven methods can be applied in unstructured environments without the consideration of physical modeling of the objects. This kind of methods commonly needs large datasets or trial-and-error for satisfied grasping.

Combining prehensile and non-prehensile manipulation policies is a new way to effectively grasp the objects in a cluttered environment. Dogar et al. [20] present a robust planning framework to reduce grasp uncertainty with the help of an additional motion primitive, which is defined to sweep around obstacles in dense clutter. Gupta et al.[21] used the Euclidean clustering algorithm to divide the scene into different regions, and assigned states to the regions to select appropriate manipulation primitives to sort Duplo bricks accurately. Boularias et al. [6] first extract hand-crafted features of pushing and grasping actions from the images and then use reinforcement learning to train control strategies to select among the feature proposals. Kalashnikov et al. [7] introduce an extensible vision-based reinforcement learning framework called QT-Opt to learn to pick up objects and execute non-prehensile pre-grasp actions. Deng et al. [9] utilize reinforcement learning to train the action pushing motions to explore and change the environment actively on the basis of the affordance map [1].

More closely related to our work is that of Zeng et al. [8], which propose deep end-to-end networks to learn synergies between pushing and grasping actions with self-supervised deep reinforcement learning. They train the networks of pushing and grasping in a mutually supportive way and then execute the action with the highest Q value. While their method only considers the changes of environment to define the success of pushes and this results in the useless pushes. And the synergies between pushes and grasps in a mutual way also sometimes low the grasp efficiency. Inspired by them, we introduce a new metric to describe the physical properties of the scene and guide the robot to perform pushes actively and purposefully and achieve better grasps. Compared with the previous work [21], our new metric is continuous and used as a reward for training pushing network. We also propose a grasping framework with deep Q learning method that after the objects in the environment are properly pushed apart, the robot begins to grasp. This further improves the grasp efficiency. We demonstrate that our system increases the success rate of grasping and also can be quickly generalized to novel objects and scenarios.

III. PRELIMINARIES

In this section, we introduce the preliminaries, which include the basic concepts for reinforcement learning, deep Q network and fully convolutional action-value function.

A. Reinforcement Learning

We consider the standard Markov Decision Process (MDP) represented as <S, A, P, R, γ >, consisting of a set of states S, a set of actions A, a transition function $P(s'|s,a)$, a reward function $R(s,a)$ and a discount factor γ. At each time step t, the agent is in the state $s_t \in S$ and performs an action $a_t \in A$ according to the current policy $\pi(s_t)$, then transitions to a new state s_{t+1} according to the state transition probability $p(s_{t+1} \mid s_t, a_t) \in P$ and receives a corresponding reward $r_t = r(s_t, a_t) \in R$. The goal in reinforcement learning is to learn an optimal policy π^* to maximize the future rewards $R_t = \sum_{i=t}^{T} \gamma^{i-t} r_t$ over time horizon T and with the discount factor $\gamma \in [0,1]$.

One way to find the optimal policy π^* is not to directly calculate the policy, but to first calculate the state-action value function $Q(s,a)$,called Q-Learning[22]. According to the Bellman Equation, the state-action value function $Q(s,a)$ is defined as:

$$Q(s_t, a_t) = r_t + \gamma \sum_{s' \in S} p(s' \mid s_t, a_t) \cdot \max_{a \in A} Q(s', a) \tag{1}$$

And the optimal action to be executed by the agent in the

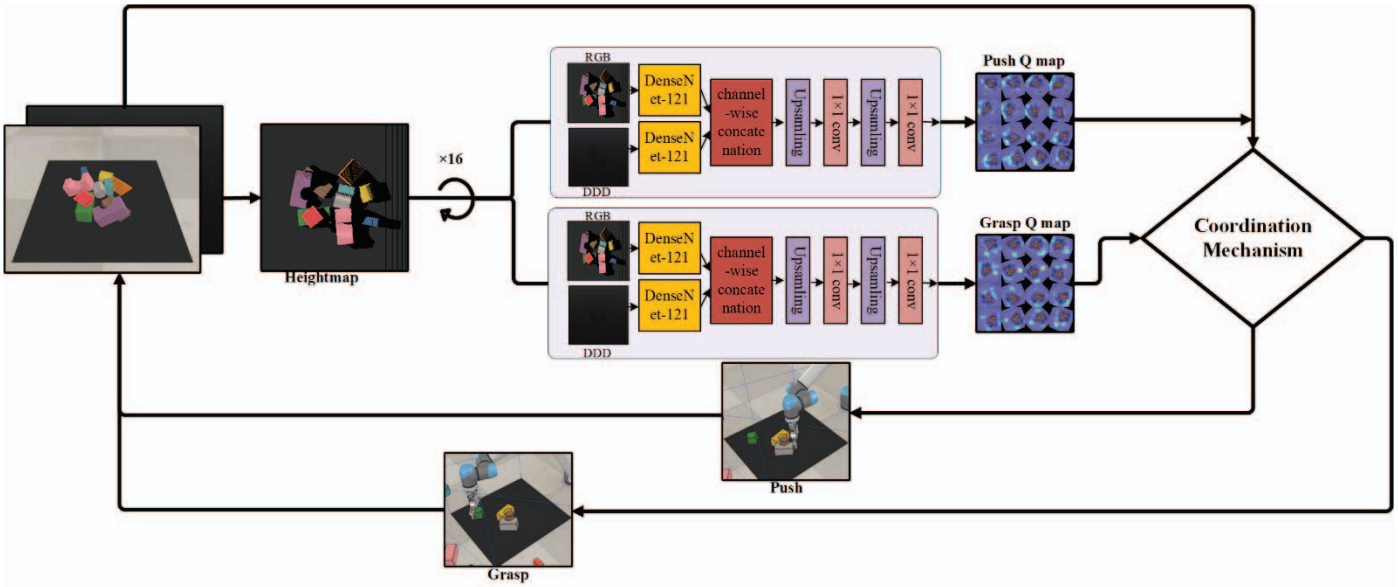


Fig. 2. System Pipeline. The system first obtains an RGB-D image and generate a heightmap. And then the heightmap is rotated with 16 different orientations and fed into two DenseNet tower structures and convolutional network to get pixel-wise Q value tables, corresponding to the pushing and grasping actions respectively. The system will select the grasp or push action according to the coordination mechanism. If push action is selected, the system will initiate an active pushing mechanism to execute the push actions with the highest Q value in the Q value table of pushing. Otherwise, the system will perform the grasp action with the highest Q value in the grasp action-value table.

state s_t is:

$$a_t = argmax_{a \in A} Q(s_t, a) \tag{2}$$

B. Deep Q-Network

When the states and action space are discrete and the dimension is not high, table-based reinforcement learning algorithms such as Q-learning can use Q-Table to store the Q value of each state-action pair, but when the states and action space are high-dimensional like robot manipulation, using Q-Table is not practical. The usual practice is to turn the Q-Table update problem into a function fitting problem.

Deep Q-Network (DQN) [23] is one of the deep reinforcement learning methods, which combines Convolutional Neural Network (CNN) and Q-Learning to generate Q value by fitting a function $Q_\pi(s,a;\theta)$ instead of a Q-Table. DQN can use end-to-end reinforcement learning to learn successful policies directly from high-dimensional sensory inputs. In this method, it modifies standard online Q-learning in two key ways to make it suitable for training large networks. First, it introduces experience replay, which stores the agent's experiences at each time-step. This technique allows for higher data efficiency and breaks the correlations of samples to reduce the variance of the updates. Second, it uses a separate network to generate the targets in the update. This further improves the stability of the method. At each iteration i, the Q network is trained using the loss function:

$$L_i(\theta_i) = E\left[\left(y_i - Q(s,a;\theta_i)\right)^2\right] \tag{3}$$

where $y_i = r + \gamma \max_{a'} Q(s',a';\theta_{i-1})$ is the q-target value calculated by target network.

C. Fully Convolutional Action-Value Function

Fully convolutional networks (FCN) [24] revolutionizes pixel-level classification of images and solves the problem of image segmentation at the semantic level. Zeng et al. [8] introduced FCN into DQN by modeling the Q function as a feedforward full convolutional network, which is referred to as Fully Convolutional Action-Value Functions (FCAVF). First, the action space is discretized into pixel-level motion primitives corresponding to the pixels of the scene image. Then the robot manipulation task is described as a pixel-wise segmentation problem. FCN is trained to take images of the workspace as input and infer pixel-wise expected Q values for actions. The action with the highest Q value would be executed at the corresponding pixel location. Compared with training end-to-end policies with deep reinforcement learning, this method has lower complexity and is more conducive to training.

IV. System Overview

The pipeline of our proposed system is shown in Fig. 2. First, the RGB image and the depth image of the cluttered scene are obtained from a fixed-mounted camera. Then the images are re-projected onto an orthographic RGB-D heightmap as a representation of the current state. And the heightmaps are rotated with 16 different orientations and then fed into two DenseNet tower structures and convolutional networks to get pixel-wise Q value tables, corresponding to the grasping and pushing actions respectively. Finally, the system will select the grasp or push action according to the coordination mechanism. If push action is selected, the system will initiate an active pushing mechanism to perform push action with the highest Q value in the pixel-wise predictions of push actions. Otherwise, the system will perform grasp action with the highest Q value of the grasp action-value table. The above process will be repeated until all objects in the workspace are successfully caught.

V. Method

This section introduces the method of our proposed system

in detail.

A. State and Action Representation

The state is represented as an RGB-D heightmap image of the environment. The RGB-D heightmap is obtained by transferring the RGB-D image observed by a fixed-mounted camera to a style which is orthographically back-project upwards in the gravity direction (same as in [8]). The workspace of the robot is a $448mm \times 448mm$ surface, which is divided into 224×224 action grids on average. Therefore, the resolution of the heightmap in the workspace is $2mm^2$.

We define two actions (e.g., push and grasp) in our system. The action $a \in \{grasp, push\}$ is parameterized as a vector (x, y, z, ψ), where (x, y, z) denotes the middle position of the gripper, and $\psi \in [0, 2\pi]$ denotes the rotation of the gripper in the table plane. When the robot chooses to execute the grasp action, the gripper moves to the coordinate point (x, y, z) and rotates ψ, then closes the fingers to grasp. When the agent chooses to execute the push action, the gripper will first close the fingers, then move to the coordinate point (x, y, z) and make a linear movement of $10cm$ in length along the direction ψ.

B. Model Structure

We adopt FCAVF to get pixel-wise predictions of future expected Q values for the pushing and grasping actions, just like in [8]. We use two FCN to approximate the Q map of the pushing and grasping actions respectively. We first rotate the input heightmap into 16 orientations and consider only horizontal pushes (to the right) and grasps after the corresponding rotation. And then the color channel (RGB) and the channel-wise cloned depth channel (DDD) of the heightmap are separately as the inputs of the two DenseNet towers [25], followed by channel-wise concatenation and 2 additional convolutional layers with bilinear interpolation (as shown in Fig. 2). Finally, the total output is 32 pixel-wise maps of Q values (16 for pushes in different rotations, and 16 for grasps at different rotations). Q value in the 32 pixel-wise maps represents the future expected return for performing the push action or grasp action at the corresponding position and rotation angle. The action which has the highest Q value in the pixel-wise maps of one motion primitive behavior would be chosen.

C. Dispersion Degree

For push action, it is not an effective way to only consider the changes of the environment after pushing as the positive rewards [8]. This cannot guarantee to spread out the objects from dense to sparse and sometimes even makes the objects closer to each other. Taking into account this condition, we introduce the metric of *Dispersion Degree*, which is designed to calculate the spread-out degree of objects in the scene. With this metric, the pushing network could be trained to guide the robot to push the objects apart purposefully.

We first calculate the distance between the center coordinates of the objects which could be segmented out from the scene using the semantic segmentation methods (FCN [24], SegNet [26] et al.) or object detection methods (Yolo [27], SSD [28] et al.). When the distance between two objects is greater than the finger-opening distance of the gripper η, it means that the objects do not affect each other when being grasped. Therefore, the dispersion distance between objects is defined as follows:

$$d\left(p_i, p_j\right) = \begin{cases} \left\| p_i - p_j \right\|_2, if \left\| p_i - p_j \right\|_2 < \eta \\ \eta, if \left\| p_i - p_j \right\|_2 \geq \eta \end{cases} \tag{4}$$

Here p_i, p_j means the center coordinate of the object i and j respectively; η means the finger-opening distance of the gripper. Taking k as the number of all objects in the scene, the *Dispersion Degree* α_t of the state s_t is defined:

$$\alpha_t = \sum_{i=1}^{k} \sum_{j=1}^{k-1} \frac{d(p_i, p_j)}{k(k-1)\eta} \tag{5}$$

According to the formula defined above, the larger α_t is, the greater the dispersion of objects in the scene. If the sum of dispersion distance of the object i with others is

$$\sum_{j=1}^{k-1} d\left(p_i, p_j\right) = (k-1)\eta \tag{6}$$

Then this object is set to be *independent* and could be grasped freely. When all objects in the scene are *independent*, the objects have no influence on each other and there is no need to push.

D. Reward

We specify the rewards for grasping and pushing actions respectively. We assign $R_g\left(s_t, s_{t+1}\right) = 1$ if a grasp is successful (the object is grasped to a specified position) and $R_p\left(s_t, s_{t+1}\right) = 1$ when $\alpha_{t+1} - \alpha_t > \delta$. In other conditions, we think the actions are failed and the rewards are set to be zero. In order to reduce noise interference, we set $\delta = 0.005$.

E. Coordination mechanism

In cluttered scenes, low action efficiency is usually caused by two situations. The first situation is when objects are too close to each other. The occlusion between objects brings great difficulties to reach the planned grasps. The second situation is when there are excessive pushes. When there is a good grasping point in the scene while the robot still chooses the push action to completely disperse the objects (sometimes even pushes the objects out of scene), that makes the action efficiency low.

Considering the above mentioned cases, we introduce the coordination mechanism into our proposed system. Instead of selecting actions with only the highest Q value, we take DD, the highest Q value for grasps Q_g and the highest Q value for pushes Q_p into consideration. So the final metric ϕ is defined as a weighted sum of the above three metrics:

$$\phi = \lambda_d * \alpha + \lambda_g * S(Q_g) + \lambda_p * (1 - S(Q_p)) \tag{7}$$

where $S(x)$ is a sigmoid function defined by $S(x) = 1/(1+e^{-x})$ and $\lambda_d + \lambda_g + \lambda_p = 1$. So $\phi \in [0,1]$. The objects in the scene are considered suitable for grasping when the metric of current scene $\phi_i > 0.5$.

TABLE I. CURRICULUM DESIGN FOR TRAINING

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Number	2	3	4	5	7	10	10
Shape	Square	Square	Square	Square	Square	Square	Random
Color	Random	Random	Random	Random	Random	Random	Random

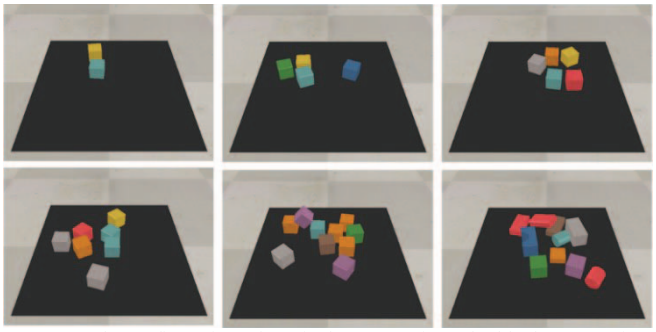


Fig. 3. Example configuration of curriculum scenarios. The blocks are set in a limited area to simulate the dense clutter.

VI. EXPERIMENT AND RESULT

We implemented a series of experiments to evaluate our proposed system. The goals of our experiments are to answer the following research questions: (1) Does our formulation make the robot learn to disperse objects? (2) Does our proposed method effectively reduce the situation of useless push? (3) Does our approach improve task performance compared to the baseline method?

A. Training

We place a UR5 robot arm and a statically mounted camera in V-REP [29] with bullet 2.83 as the simulation setup. And we add 10 blocks with different shapes and colors in a limited workspace to simulate the dense cluttered environment. In order to get satisfied policies, we separately train the robot to learn the pushing and grasping policies.

Pushing Policy Training with Curriculum Learning: In dense clutter, the objects are so close that the spread-out states have little changes explicitly after one push action. This induces that the system would take a long time to converge to a satisfactory solution. In order to solve this problem, we use the curriculum learning method [30]. Curriculum learning is similar to the human learning mechanism: learning simple skills first, and then learning difficult ones. In our system, we can control the curriculum difficulty level by changing the number and shape of objects added to the scene. During training in the scene, we add $2 \sim 10$ blocks to the scene to design curricula (Class 0~6), as shown in TABLE I. The blocks are set in a limited area to simulate the dense clutter. Some curriculum scenarios are shown in Fig. 3. In all curriculum learning, we reset the scene only when the objects in the scene are all *independent* as defined in Section V.C. Fig. 4 illustrates the training process of curriculum learning. Pushing performance in training is measured by the success rate percentage of the last $i = 300$ push attempts. From the curve of the training results, we can see that the curriculum learning can increase the generation speed and accelerate the convergence speed.

Grasping Policy Training: In the scene where the grasp policy is trained, we added 10 blocks to the scene under the condition

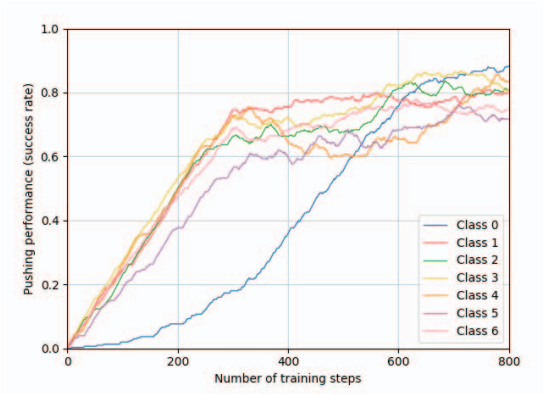


Fig. 4. Performance of our policies trained in 7 classes. Different color lines indicate the push success rate of different classes. From the curve of the training results, we can see that the learning efficiency of later courses is greater than the previous courses.

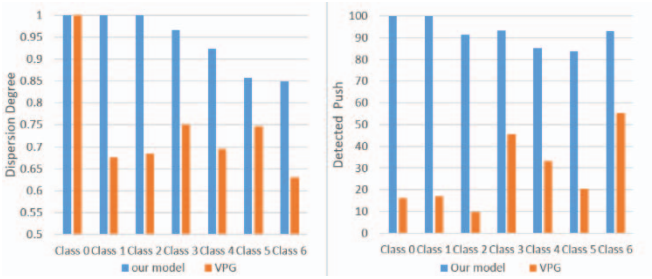


Fig. 5. Performance comparison of curricula between our method and VPG at testing. Our proposed method shows higher performance in all metrics.

with enough dispersion degree. These 10 blocks include not only graspable objects with random colors and shapes but also objects that are too large for the gripper to grasp. This design would make the robot to learn not to grasp in the dense locations, which could improve the grasp success rate of the robotic grasping system.

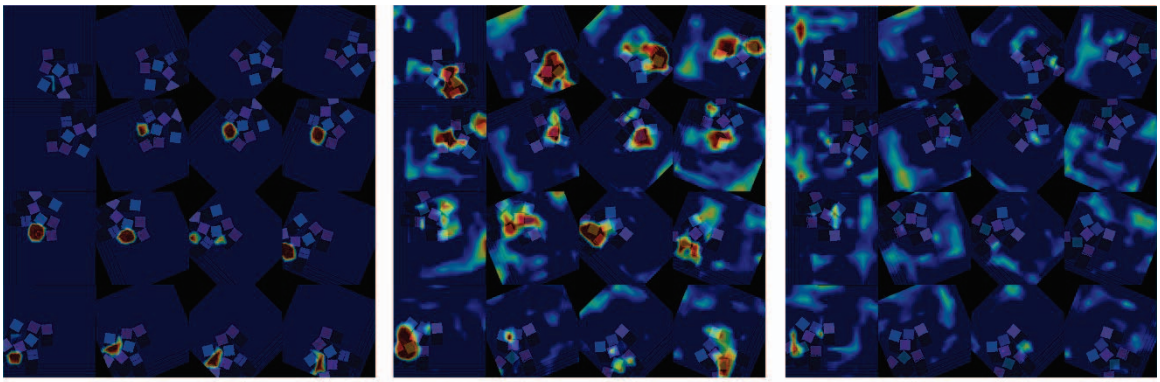
We train all of our models in PyTorch with version 0.3. Our proposed method is optimized by Stochastic Gradient Descent (SGD) with the momentum of 0.9 and 2×10^{-5} weight decay. The learning rate is set to 10^{-5}. The network uses a prioritized experience replay [31] with stochastic rank-based prioritization when training. The method uses the future discount factor $\gamma = 0.5$. The weight parameters $\lambda_d, \lambda_g, \lambda_p$ in coordination mechanism are set to 0.33.

B. Evaluation results

1) Push Experiment Results

We compare the push model with that in VPG [8]. For each test, we execute 30 test runs every class and use two metrics to evaluate its performance: 1) the average dispersion degree of state after a fixed number of pushes, 2) the average percentage of pushes with detectable changes to the environment after a certain amount of pushes. Here we set the number to be 10. Note that the higher these two metrics, the better the model.

The evaluation results of our model in 7 classes are summarized in Fig. 5. The results show that our push model performs better than the baseline in all classes. Compared with the push model trained in VPG, our method reduces the probability of useless pushing action in the area with no objects and increases the ability to disperse objects in dense clutter.



(a) Our Prediction (b) Baseline Successful Prediction (c) Baseline Failed Prediction

Fig. 6. Examples of Visualized predicted Q function under testing case in simulation. Each pixel in the heightmap represents a different execution position of the motion primitive. And there are 16 heightmaps that repeat 16 different rotations to explain the different pushing angles. The red circle indicates the highest Q value in each prediction. Fig. 6a is the prediction from our model, where all high Q values are concentrated around the objects. Though Fig. 6b is a successful prediction by the baseline method, but there are also high predictions above the object or in the area with no objects nearby. This indifferent prediction about the dense area and sparse area causes typical failures for the baseline method as is in Fig. 6c.

TABLE II. RESULTS ON RANDOM ARRANGEMENTS(MEAN %)

	Completion	Grasp Success	Action Efficiency
VPG	100.0	67.7	60.9
Ours	100.0	77.5	76.8

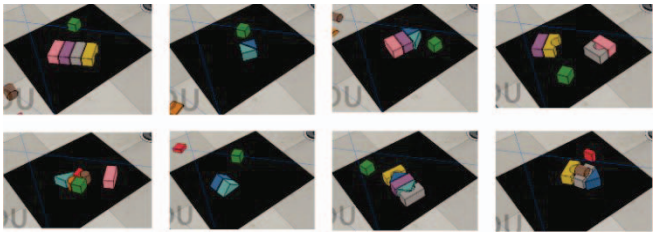


Fig. 7. Example configuration of challenging scenes. The blocks are set side by side.

TABLE III. RESULTS ON CHALLENGING ARRANGEMENTS (MEAN %)

	Completion	Grasp Success	Action Efficiency
VPG	82.7	77.2	60.1
Ours	100.0	88.7	68.6

Fig. 6 illustrates the visualized Q predictions of our method and that in VPG while executing the task of Class 4. From Fig. 6a, we can see that our method can successfully identify the locations of clustered objects, and the predicted high Q values are around the objects. However, the baseline method cannot guarantee the exact push locations. Fig. 6b shows an example of a successful baseline method prediction in a test scene, there are high Q values in the area of the objects and also in the area with no objects nearby. Fig. 6c shows the failure of the baseline method, where the highest Q value point is directly in the area with no objects nearby.

2) Grasping System Experimental Results

We test our proposed grasping system by evaluating the performance of the method in different test runs. In these test runs, the system needs to successfully grasp objects from the desktop in different scene settings.

We compare the proposed system with VPG. For each test, we execute n test runs $(n \in\sim 10,30)$ every scene and use three metrics to evaluate the performance: 1) The average percentage of completion in n test runs, which measures the ability of a strategy to complete a task by picking up all objects in the scene without failures of more than 10 consecutive attempts, 2) the average success rate of grasp per completion, 3) the action efficiency which is defined as $\frac{n_{object}}{n_{action}}$ (where n_{object} is the number of blocks in test and n_{action} is the number of actions per completion) to describe the policy succinctness to complete the task.

We first test the methods with 30 randomly placed objects in cluttered environments. The experimental results are summarized in TABLE II, which shows that our proposed system is better than VPG in all metrics. And we can also see that for our system, the grasp success and action efficiency are similar. That means our method select more grasps to complete the tasks and the grasp actions are more effective. The reason is that we place large objects to the environment when training the grasp policy and this makes robot to learn not to grasp the objects in dense area.

We then test the methods in 11 challenging environments with tightly packed blocks (some scenes are shown in Fig. 7). These configurations mainly include objects that are set side by side, which is hard to grasp if no synergies of actions. This setting is to verify that the system has generalization capabilities to handle multiple situations. The comparison results are summarized in TABLE III. Our method can still greatly improve the completion rate and the grasp success rate. This is mainly due to the active pushing and coordination mechanism. Compared with TABLE II, we can see that the action efficiency of our method is lower in challenging environment than that in the environment with randomly placed objects. It means our method could use more pushes to better grasp the objects.

VII. CONCLUSIONS

In this work, we present a novel robotic grasping system for better grasping in dense clutter with deep reinforcement learning. We introduce an active pushing mechanism with a new metric

of dispersion degree to describe the physical properties of the scene. And we also build a grasping framework with the synergies between pushing and grasping actions. Empirical results demonstrate our proposed method can effectively reduce the useless pushes and improve the grasp success rate. And also, the trained policies could be generalized to more challenging test environments. The next step for us is to apply our method in real-world conditions to further test the performance under the influence of environmental noises. Also we would train our method to test in more dense and cluttered environments with large varieties of objects except for blocks.

ACKNOWLEDGMENT

This work was supported by the National Key R&D Program of China (grant 2019YFB1311901) and partly by Science and Technology on Space Intelligent Control Laboratory for National Defense, No. KGJZDSYS-2018-09, the National Natural Science Foundation of China under Grant U1713222, 61773378.

REFERENCES

[1] Zeng A, Song S, Yu K T, et al. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching[C]//2018 IEEE international conference on robotics and automation (ICRA). IEEE, 2018: 1-8.

[2] Goyal S, Ruina A, Papadopoulos J. Planar sliding with dry friction part 1. limit surface and moment function[J]. Wear (Amsterdam, Netherlands), 1991, 143(2): 307-330.

[3] Hogan F R, Rodriguez A. Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics[J]. arXiv preprint arXiv:1611.08268, 2016.

[4] Finn C, Levine S. Deep visual foresight for planning robot motion[C]//2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2017: 2786-2793.

[5] Clavera I, Held D, Abbeel P. Policy transfer via modularity and reward guiding[C]//2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017: 1537-1544.

[6] Boularias A, Bagnell J A, Stentz A. Learning to manipulate unknown objects in clutter by reinforcement[C]//Twenty-Ninth AAAI Conference on Artificial Intelligence. 2015.

[7] Kalashnikov D, Irpan A, Pastor P, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation[J]. arXiv preprint arXiv:1806.10293, 2018.

[8] Zeng A, Song S, Welker S, et al. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning[C]//2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018: 4238-4245.

[9] Deng Y, Guo X, Wei Y, et al. Deep Reinforcement Learning for Robotic Pushing and Picking in Cluttered Environment[C]//2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2019: 619-626.

[10] Xiang Y, Schmidt T, Narayanan V, et al. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes[J]. arXiv preprint arXiv:1711.00199, 2017.

[11] Mitash C, Boularias A, Bekris K. Robust 6d object pose estimation with stochastic congruent sets[J]. arXiv preprint arXiv:1805.06324, 2018.

[12] Ponce J, Sullivan S, Boissonnat J D, et al. On Characterizing and Computing Three- and Four-Finger Force-Closure Grasps of Polyhedral Objects[C]// IEEE International Conference on Robotics & Automation. IEEE, 1993.

[13] Mirtich B, Canny J. Easily computable optimum grasps in 2-D and 3-D[C]//Proceedings of the 1994 IEEE International Conference on Robotics and Automation. IEEE, 1994: 739-747

[14] Zeng A, Song S, Nießner M, et al. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017: 1802-1811.

[15] Zeng A, Yu K T, Song S, et al. Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge[C]//2017 IEEE international conference on robotics and automation (ICRA). IEEE, 2017: 1386-1383.

[16] Jiang Y, Moseson S, Saxena A. Efficient grasping from rgbd images: Learning using a new rectangle representation[C]//2011 IEEE International conference on robotics and automation. IEEE, 2011: 3304-3311.

[17] Pinto L, Gupta A. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours[C]//2016 IEEE international conference on robotics and automation (ICRA). IEEE, 2016: 3406-3413.

[18] Jang E, Vijayanarasimhan S, Pastor P, et al. End-to-end learning of semantic grasping[J]. arXiv preprint arXiv:1707.01932, 2017.

[19] Gualtieri M, ten Pas A, Platt R. Pick and place without geometric object models[C]//2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018: 7433-7440.

[20] Dogar M R, Srinivasa S S. A planning framework for non-prehensile manipulation under clutter and uncertainty[J]. Autonomous Robots, 2012, 33(3): 217-236.

[21] Megha Gupta, Gaurav S. Sukhatme. Using Manipulation Primitives for Brick Sorting in Clutter[C]// IEEE International Conference on Robotics & Automation. IEEE, 2012.

[22] Sutton R S, Barto A G. Reinforcement learning: An introduction[M]. MIT press, 2018.

[23] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.

[24] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 3431-3440.

[25] Huang G, Liu Z, Van Der Maaten L, et al. Densely connected convolutional networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 4700-4708.

[26] Badrinarayanan V, Kendall A, Cipolla R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation[J]. IEEE transactions on pattern analysis and machine intelligence, 2017, 39(12): 2481-2495.

[27] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 779-788.

[28] Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]//European conference on computer vision. Springer, Cham, 2016: 21-37.

[29] Rohmer E, Singh S P N, Freese M. V-REP: A versatile and scalable robot simulation framework[C]//2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2013: 1321-1326.

[30] Bengio Y, Louradour J, Collobert R, et al. Curriculum learning[C]// Proceedings of the 26th annual international conference on machine learning. 2009: 41-48.

[31] Schaul T, Quan J, Antonoglou I, et al. Prioritized experience replay[J]. arXiv preprint arXiv:1511.05952, 2015.