Grasp Planning Based on Deep Reinforcement Learning: A Brief Survey

Zixin Tang, Xin Xu and Yifei Shi

College of Intelligence Science and Technology
National University of Defense Technology

2021 China Automation Congress (CAC) | 978-1-6654-2647-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/CAC53003.2021.9727526

***Abstract*—Grasping is a fundamental ability for robots to interact with the environment. Recent advances show great progress on grasping unstructured household objects by leveraging deep reinforcement learning. This paper presents a brief survey on grasp planning based on deep reinforcement learning. The goal of grasp planning is to determine the actions of the robotic manipulator moving from its initial position to the target one so that the object could be grasped and manipulated. However, objects could be stacked or occluded, making the target object ungraspable by applying a simple grasp action. To tackle this issue, auxiliary actions (such as pushing) should be conducted along with grasping. According to whether auxiliary actions are performed, we categorize the existing grasping planning methods into two classes: methods without auxiliary actions and methods with auxiliary actions. We review them in details and show the advantages and limitations of each type. Furthermore, we show several popular evaluation metrics that are widely used in grasp planning. Finally, difficulties and future research directions are discussed.**

***Index Terms*—robot, grasp planning, deep reinforcement learning**

I. Introduction

Robotic grasping has a wide range of application prospects in industry and households. Grasping regular industrial parts has been relatively mature while grasping household objects puts higher demands on the flexibility and intelligence of robots due to various shapes and complex textures. Hence, grasping unstructured household objects is still an unsolved problem.

Grasp planning (GP) is a key component of robotic grasping, which forms the basis of some manipulation tasks [1], such as Pick-and-Place [2], Insertion [3], and Grasp the Invisible [4]. Formally, vision-based robotic grasping involves four key tasks, object localization, pose estimation, grasp detection, and motion planning, which can be accomplished independently or jointly [5]. Grasp planning (GP) is a combination of these four tasks. Namely, the robotic arm plans paths from the initial pose to the target grasp pose using scene information obtained from sensors. The path should satisfy the kinematic constraints of the robotic arm and avoid obstacles. The extra motion should also be prevented from the perspective of grasp efficiency. However, to grasp objects in cluttered scenes, some auxiliary actions, such as pushing, should be performed to create feasible grasping [6] [4] [7].

In recent years, with the powerful representation capabilities of neural networks [8], deep reinforcement learning (DRL) has shown promising sequential decision-making abilities in many tasks, such as video games [9] or robotic manipulation tasks [6] [2]. Since agent learns from interacting with the environment by trial-and-error, a large number of trials are required to train DRL algorithms, especially with sparse rewards [10] [11]. However, the training requirements of large amounts of data hinder the application of DRL algorithms to real-world robotic manipulation tasks.

As a result, many DRL based grasp planning methods only infer 3/4 DoF grasps, such as top-down grasps [12] [11]. Due to the limitation of DoF, some household objects like horizontal plates are hard to grasp. In order to alleviate the immense exploration search space faced by 6 DoF grasp planning, learning from demonstrations [13] and Sim2Real [7] for DRL based grasp planning are widely studied recently. Although there are some hurdles for applying DRL in robotic grasp planning, it is worth noting that DRL has been playing a more and more important role in it.

In this work, we present a brief survey on grasp planning based on deep reinforcement learning. According to whether auxiliary actions are performed, we divide them into two categories: GP without auxiliary actions [14] [3] [2] [15] and GP with auxiliary actions [4] [16] [17] [7].

For GP without auxiliary actions, according to whether the target grasp pose is detected before, grasp planning can be divided into open-loop methods [18] and closed-loop methods [11] [12] [7] [15]. For the open-loop method, the target grasp pose is known before planning, hence it only needs to plan the motion trajectory. Whereas, due to the tight coupling with the grasping detection algorithm [19] [20], there are two shortcomings. First, perceptual errors in the grasp detection algorithm will sensitively affect the results of planning. More importantly, since it is essentially a two-stage method, disturbances in the scene are hard to cope with, such as moving objects. On the contrary, the close-loop method directly plans grasp paths in an end-to-end manner without prior knowledge of grasp poses, which is more robust to perceptual noises, kinematic errors as well as robust in dynamic scenes [14]. Hence, we only exhaustively review the close-loop method for GP without auxiliary actions.

This work was supported by National Natural Science Foundation of China (61825305, 62002379) and the Zhejiang Lab's International Talent Fund for Young Professionals.

For GP with auxiliary actions, they focus on tackling scenes with non-graspable objects. Rather than directly plan a path from the initial pose to the target pose, these methods make reasonable decisions on performing both prehensile and non-prehensile actions in sequence to generate feasible grasp poses. When grasp poses are detected, some existing planning algorithms [21] could be used to perform motion planning. This means that the problems with open-loop methods will exist in these methods, like unable to tackle dynamic scenes and kinematic noises.

The remainder of the paper is organized as follows: Section II provides the basic concepts of DRL. Representative works and comparisons between them are provided in section III. In addition, we summarize the advantages and challenges of each class method and then provide wildly used evaluation metrics. Finally, difficulties and future research directions are discussed in Section IV.

II. A BRIEF REVIEW OF DEEP REINFORCEMENT LEARNING

Reinforcement learning (RL) is formulated as a Markov decision process (MDP): at time t, the agent receives a state $s_t \in \mathbb{S}$ from environment, and executes an action $a_t \in \mathbb{A}$ according to the policy $\pi(s_t)$. Then the environment transits to a new state $s_{t+1} \in \mathbb{S}$ and provides the agent with an immediate reward $R(s_t, a_t, s_{t+1})$. The goal of the agent is to find the optimal policy π^* through interaction with the environment by trial-and-error that maximizes the expected discounted future rewards, where we define the whole discounted future reward as $\mathbb{G}_t = \sum_{i=t}^{\infty} \gamma^{i-t} R_i(s_i, a_i, s_{i+1})$, $0 < \gamma < 1$ is a discount factor to weigh the importance of future rewards.

The most representative method of model-free RL algorithms is DQN [9], which belongs to value-based methods. DQN estimates the state-action value function $Q_\theta(s_t, a_t)$ to a target value y_t by minimizing temporal difference error δ_t iteratively:

$$\begin{aligned} \delta_t &= |Q_\theta(s_t, a_t) - y_t| \\ y_t &= R_t(s_t, a_t, s_{t+1}) + \gamma \max_{a' \in \mathbb{A}} (Q_{\bar{\theta}}(s_{t+1}, a')) \end{aligned} \tag{1}$$

where $Q_\theta(s_t, a_t)$ denotes the estimation of $\mathbb{E}_\pi[\mathbb{G}_t \mid s_t, a_t]$. Other variants include Double DQN [25], Dueling DQN [26], etc.

Policy-based methods [27] [28] are the second category of model-free RL algorithms which learn policy directly to maximize the expected future rewards. Among them, Proximal Policy Optimization (PPO) [27] and Soft Actor-Critic (SAC) [28] are widely used in grasp planning.

PPO is an on-policy policy gradient algorithm in the Actor-Critic style. One branch is to estimate state value function $V(s_t)$, and the other branch takes the clipped surrogate objective to conservatively optimize π_{old} with the constraints of not pushing updated policy π too far away from π_{old}:

$$\mathcal{J}(\pi) = \mathbb{E}_{(s_t, a_t) \sim \rho_{\pi_{old}}} [\min(r_t A_t, \text{clip}(r_t, 1-\epsilon, 1+\epsilon) A_t)] \tag{2}$$

where $r_t = \frac{\pi(a_t|s_t)}{\pi_{old}(a_t|s_t)}$ is the important sampling coefficient and A_t is an advantage function that indicates the benefit of the current action compare to the average, one of the forms is calculated as $A_t = (r_t + V(s_{t+1}) - V(s_t))$.

Differently, SAC is an off-policy RL algorithm based on the maximum entropy framework, where the objective function is softly changed by augmenting with an expected entropy term as shown in (3). Here, α, called temperature parameter, trades-off the importance of accumulative future rewards and the entropy term which has an influence on the stochasticity of the policy π. ρ_π represents the state-action distribution induced by π as well as state transition probability distribution of the environment.

$$\begin{aligned} \mathcal{J}(\pi) = \sum_{t=0}^{\infty} \mathbb{E}_{(s_t, a_t) \sim \rho_\pi} \Bigg\{ \sum_{l=t}^{\infty} \gamma^{l-t} \mathbb{E}_{(s_l, a_l) \sim \rho_\pi} \Big[R(s_l, a_l, s_{l+1}) \\ + \alpha \mathcal{H}(\pi(\cdot \mid s_l)) \mid s_t, a_t \Big] \Bigg\} \end{aligned} \tag{3}$$

Since entropy facilitates exploration, SAC is efficient for continuous state and action spaces, so it is suitable for robotic grasping tasks.

III. METHODS OVERVIEW

Grasp planning denotes that the manipulator is required to plan a feasible and efficient path from the initial configuration to the target grasp pose using scene information obtained from sensors. Formally, a path should obey the following properties: 1) violation of joint restrictions should not be allowed for every configuration in the path, 2) the ability to avoid collisions with the environment as well as objects in the scene is critical, 3) redundant motions in the path will decline grasp efficiency, while some auxiliary actions, such as pushing or lifting, are allowed to create feasible poses for initial non-graspable objects [16].

In this paper, we divide grasp planning algorithms based on DRL into two categories: without auxiliary actions and with auxiliary actions. Table I compares the basic properties among these methods, like in Sim2Real style or others, use demonstrations or not, etc. We then show the advantages and challenges of each class method. Finally, we list widely used evaluation metrics in grasp planning tasks.

A. GP without auxiliary actions

These methods consider the most general condition that there are feasible grasp poses for objects in the initial scene arrangement. Hence they plan paths from initial configurations to target grasp poses without auxiliary actions. According to the type of RL algorithms, we further divide them into value-based or policy-based subclasses.

Value-based Methods Levine et al. [10] presented a learning-based approach for hand-eye coordination in robotic grasping from a monocular camera mounted behind the arm. Two components are devoted to continuously control a robotic gripper to the pose with a high grasping success rate, where the first serves the second. The first part is a grasp success predictor used for predicting the probability of successful grasp given the visual inputs of the scene and the motion

TABLE I
COMPARISONS OF GRASP PLANNING METHODS

	Paper list	*RL Alg.*	*Sim/Real*	*6 DoF*	*Demonstrations*	*Unstructured objects*	*Observations*
Without auxiliary actions	Levine et al. [10]	Value-based[1]	Real	▫	▫	▫	RGB
	Viereck et al. [14]	Value-based[1]	Sim2Real	▫	▫	▫	Depth
	Kalashnikov et al. [11]	Value-based	Real	▫	▫	▫	RGB & state vector
	Bodnar et al. [22]	Value-based	Sim. & Real	▫	▫ (Sim.) ▫ (Real)	▫	RGB & state vector
	Julian et al. [23]	Value-based	Real	▫	▫	▫	RGB & state vector
	Akinola et al. [3]	Value-based	Sim.	▫	▫	▫	RGB & state vector
	Chen et al. [12]	Policy-based	Sim2Real	▫	▫	▫	RGB
	Zhan et al. [2]	Policy-based	Real	▫	▫	▫	RGB
	Hermann et al. [24]	Policy-based	Sim2Real	▫	▫	▫	RGB & state vector
	Song et al. [15]	Value-based	Real	▫	▫	▫	RGBD
With auxiliary actions	Zeng et al. [6]	Value-based	Sim2Real	▫	▫	▫	RGBD
	Yang et al. [4]	Value-based	Sim2Real	▫	▫	▫	RGBD
	Liang et al. [16]	Value-based	Sim2Real	▫	▫	▫	RGBD
	Sun et al. [17]	Policy-based	Sim2Real	▫	▫	▫	EE[2] & Object pose
	Tang et al. [7]	Value-based	Sim2Real	▫	▫	▫	RGBD

[1]denotes it is not a standard RL method.
[2]denotes the end-effector.

command. The architecture of it is shown in Fig. 1. Then, with the cross-entropy method (CEM) [29] and two heuristics, a servoing mechanism utilizes the predictor to continuously provide promising motion commands. Over 800,000 grasp attempts were collected by 6 to 14 manipulators with slightly different poses to the camera in two months to train the predictor. This difference makes calibration between robot and camera unnecessary. Since the motion command is a vector from the current pose to the grasp pose rather than an immediate action, it can only be interpreted as standard reinforcement learning when $T = 2$.

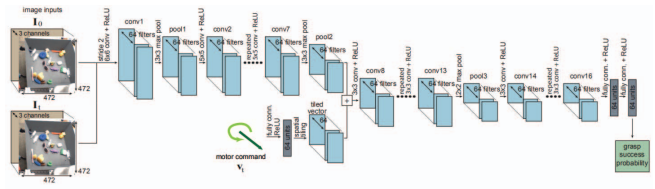

Fig. 1. The architecture of the CNN grasp predictor [10]. The input image I_t, as well as the pre-grasp image I_0, are fed into a large convolutional neural network. The motor command v_t processed by one fully connected layer is concatenated with the feature maps of pool2 after tiling the output to the same shape. Finally, the network outputs the probability of successful grasp by executing the command v_t.

Viereck et al. [14] learned a close-loop controller (CTR) in Sim2Real style. A LeNet-like convolutional neural network (CNN) takes as input a depth image I and planar pose $a = (x, y, \theta)$, approximates a distance $d(I, a)$ to the nearest grasp. The training data consists of a large amount of image-action pairs that is collected in simulation using Open RAVE [30]. For each scene, CTR acquires the current depth image from sensors and samples a collection of candidate grasp poses. Then they use the trained CNN to select the min-cost one as the action. And a fractional step action is executed for stability. This process provides a continuous servoing mechanism to close-loop grasping which shows a better performance than the grasp pose detection (GPD) [31] based open-loop method when tackling with perceptual errors and dynamic scenes.

Kalashnikov et al. [11] proposed a continuous action version of Q-learning algorithm, QT-Opt, as an alternative to the actor-critic setting to improve the stability and scalability of the algorithm, shown in Fig. 2. QT-Opt learns a generalizable vision-based 4 DoF grasp planning method from a monocular RGB camera mounted over the shoulder. They collected over 580k grasp attempts over the course of four months across 7 robots. Then, the whole network is optimized with the cross-entropy method (CEM) [29] using off-policy data and joint finetuned with 28k on-policy data obtained from online trial-and-error.

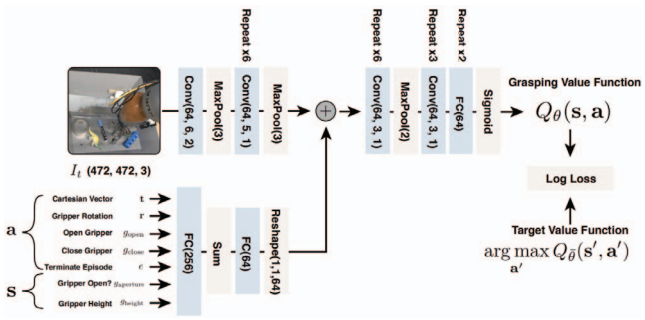

Fig. 2. The architecture of grasping Q-Function [11]. The state is defined as $s = (I, g_{aperture}, g_{height})$, and the action $a = (t, r, g_{close}, g_{open}, e)$ is a mixture of five-dimensional continuous vector and two-dimensional discrete vector, where $t \in \mathbb{R}^3$ denotes translation, $r \in \mathbb{R}^2$ is a sine-cosine code of rotation, (g_{close}, g_{open}) is a one-hot vector of open/close command, and e decides the termination or not. Sigmoid is used to activate the final Q-values so as to it is in the range of [0, 1].

Further, Bodnar et al. [22] proposed Quantile QT-Opt (Q2-Opt) built on the top of QT-opt which adopts a distributed Q-learning. They examined the positive impacts of distributed variants both on simulated and real robotic grasping tasks. Julian et al. [23] further demonstrated that a simple fine-

tuning of off-policy DRL using only very little new data can substantially improve the performance of the base policy on significantly modified tasks, where the base policy is the QT-Opt trained using 608,000 off-policy and on-policy grasp attempts totally. Akinola et al. [3] presented a multi-view approach to solve robotic manipulation tasks in a close-loop fashion. They utilized a QT-Opt-like framework to process images obtained by multiple uncalibrated cameras mounted in different views. This multi-view approach is examined in Simulation with Bullet Physics simulator [32] on Stacking and Insertion tasks. The results show a superior grasp planning performance compared to the single-view method.

Policy-based Methods Chen et al. [12] proposed a two-stage training framework that trains perception and policy separately. In the perception stage, mask R-CNN [33] in conjunction with PCA detects center points and principal directions of objects from an RGB image. Then the object pose is concatenated with the manipulator configuration to represent a six-dimensional continuous state. Further, they employed PPO [27] to learn the control policy that guides the next offset of moving direction and angle of the gripper. Benefiting from taking a low-dimensional state as input, the learned policy can be directly transferred to the real robotic system without additional adjustment after about 30 minutes of training in simulation V-REP [34].

Besides, Hermann et al. [24] proposed Adaptive Curriculum Generation from Demonstrations (ACGD) that adaptively set the learning difficulty for the agent to relieve the catastrophic exploration problem caused by DRL with sparse rewards and continuous state-action spaces. They use PPO [27] to train the policy network with few demonstrations (10). The network takes as input a mixture of an RGB image as well as a discrete proprioceptive vector, and outputs a five-dimensional control action $a = [\Delta x, \Delta y, \Delta z, \Delta\theta, a_{gripper}]$, where the former three dimensions denote the translation offset of the gripper in Cartesian space, $\Delta\theta$ is the increment of rotation angle in the z-axis, and $a_{gripper}$ indicates a discrete open-close command. They trained the policy in simulator PyBullet [32] and applied it to the real robotic environment with zero-shot transfer. Evaluations on two manipulation tasks, block stacking and pick-and-stow, show higher performances of ACGD compared to other technologies like behavior cloning (BC) [35].

In contrast to the above methods, either collecting a large number of expert demonstrations which is time-consuming and laborious or using Sim2Real style, Zhan et al. [2] presented a novel framework for efficient robotic manipulation (FERM) that can directly learn from the real environment with ten demonstrations. As shown in Fig. 3, FERM contains three processes. First, they collected few expert demonstrations to initial replay buffer. Then, a contrastive learning method [36] in self-supervised learning is introduced to pre-train a convolutional encoder with query-key pairs augmented from demonstrations, which is proven to learn representative features. Finally, they adopted SAC [28] to learn top-down grasping policies with data-augmented images captured from two RGB cameras. They tested it on six manipulation tasks, Reach, Pickup, Move, Pull, Light Switch, and Drawer Open. FERM has demonstrated that it has fast learning capabilities through which it can learn effective policies within 15-50 minutes on average. However, they grasped structural blocks instead of household objects.

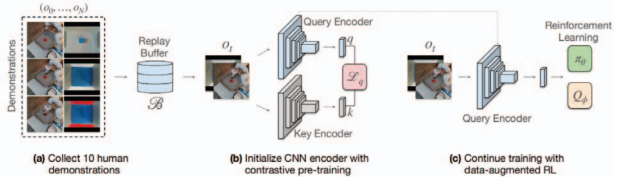

Fig. 3. The architecture of FERM [2]. First, demonstrations are collected and stored in a replay buffer. These observations are used to pre-train the encoder with a contrastive loss. The encoder and replay buffer are then used to train an DRL agent using an offline data-augmented RL algorithm.

Grasp planning with high DoF has critical practical significance, especially for grasping unstructured household objects, such as horizontal plates. Song et al. [15] introduced a piece of low-cost portable grabber equipment to collect gripper-centric demonstrations and presented an action-view based 6 DoF close-loop grasping method. They used Truncated Signed Distance Function (TSDF) fusion to aggregate a complete 3D representation of the scene and then generated virtual action-view observations. Taking as input current and future observations, the action selection network outputs the corresponding Q maps rather than Q values for preserving local visual and geometrics features. Since there is a gap between demonstrations and real robot data, they used demonstrations to pre-train the network and fine-tuned it with real interactions by trial-and-error.

Advantages and Challenges These methods utilize a close-loop manner to grasp objects, which are more robust to perceptual errors, kinematic noises, and dynamic scenes. The main limitation of them is that they will fail in scenes where there is no feasible grasp poses initially, such as the object pose exceeds the gripping range of the end-effector, the target object is buried or blocked by obstacles. In addition, due to the inefficient learning caused by DRL with sparse rewards and continuous state-action spaces, grasping unstructured objects requires a large amount of data to train, which either comes from expert demonstrations or simulation learning. However, collecting demonstrations is time-consuming and laborious, while learning in Sim2Real style puts a gap in real robotics tasks.

B. GP with auxiliary actions

There are increasing researches focus on grasp planning with auxiliary actions that are essential for objects from ungraspable poses. Rather than directly plan a path from the initial pose to the target pose, some non-prehensile manipulations such as pushing or lifting need to be performed synergistically to create viable grasping postures.

Zeng et al. [6] explored the synergies between pushing and grasping through the value-based RL algorithm DQN [9]. Two fully convolutional networks, trained with Huber loss function [37], infer the pixel-wise Q maps for pushing and

grasping motion primitives. Then the action corresponding to the maximum value in the maps is executed with a collision-free IK solver [21]. They trained the policy in V-REP [34] and transferred it to the real environment without further training. Results show that sequential executions of pushing and grasping with appropriate policies have greatly improved the completion and grasp success rate, defined in Section III-C.

Yang et al. [4] provided a target-oriented method in a critic-policy format to solve the Grasping the Invisible problem. They endow a robot with the ability of explorative pushing, singulation by target-oriented pushing, and target-oriented grasping. As shown in Fig. 4, two subtasks, exploration and coordination, are conducted sequentially. With the assumption of a visible target, the critic takes as input heightmaps composed of color heightmap, depth heightmap, and target mask heightmap. Then it predicts the target-oriented pushing and grasping maps that denote the state-action values of every pixel in each primitive motion. The second part is the policy, which consists of two components served for different subtasks: the exploration policy (Explorer) and the coordination policy (Coordinator). When the target is invisible, the Explorer is utilized to provide the most promising point for explorative pushing with the help of the target-agnostic pushing map A_p, clutter map C_p as well as pushing failure map F_p. After discovering the target, they proposed a classifier-based policy Coordinator, which is used to macroscopically determine the type of primitive motions to be executed for sequential singulation and grasping. Then the critic serves precise actions. The reward function is carefully designed in a pre-post format.

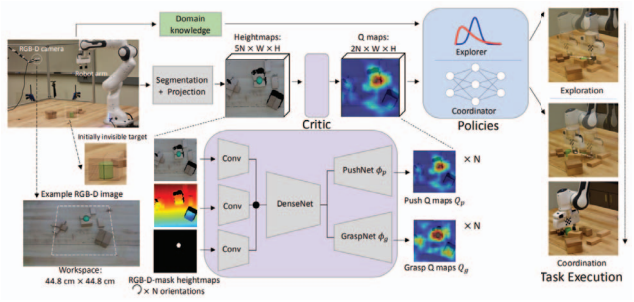

Fig. 4. Overview of the critic-policy framework [4].

Liang et al. [16] proposed a Knowledge Induced DQN (KI-DQN) with a single Shovel-and-Grasp (SaG) motion primitive for Slide-to-Wall grasping problem that requires the use of environments (like walls) to grasp objects that are not graspable in initial poses. They endowed DQN [9] with heuristically defined knowledge that the predicted action points should locate on the object. The variant of DQN improves the training efficiency and adaptability to unseen walls even in Sim2Real style.

Sun et al. [17] presented a SAC [28] based DRL method to learn pre-grasp policy for manipulation with a spherical end-effector. The pre-grasp policy is required to create feasible grasping conditions through collaborative pushing and lifting for objects from ungraspable poses. They use robot sensors to detect end-effector poses and external sensors to detect object poses, all of which form the input of the policy network. The network outputs a three-dimensional continuous action that denotes the 3 DoF offset on position and orientation of the end-effector. PyBullet [32] provides a simulation environment to train the network. Once a feasible grasp pose is established, the grasp panning algorithm can be applied successfully.

Tang et al. [7] developed the work of Zeng et al. [6] to learn collaborative planar pushing and 6 DoF grasping policies. As shown in Fig. 5, they trained a two-branch deep fully convolutional network with DRL. Specifically, they integrated a grasping pose generator (GPG) [38] into the grasping network to generate grasp poses for each sampled grasp point. Then the poses combined with corresponding RGB features are concatenated and further processed by a fully connected layer, after which the network outputs pose-wise Q values. The comparisons with the baseline show the superiority with high DoF of grasping.

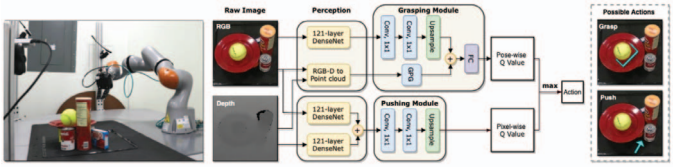

Fig. 5. The architecture of 6 DoF pushing and grasping polices [7].

Advantages and Challenges These methods make reasonable sequential decisions of non-prehensile and prehensile manipulations, which are suitable for initially non-graspable scenes. However, they produce target grasp poses instead of feasible motion trajectories to them for grasping actions. So some mature planning algorithms [21] are leveraged to plan paths. This means that the problems with open-loop methods will exist in these methods, like being unable to tackle dynamic scenes and sensitive to kinematic noises.

C. Evaluation Metrics

When evaluating the performance of GP methods, appropriate metrics should be considered. First, grasping robustness shows the robustness of the GP method when grasping objects, that is, the object is successfully grasped without dropping. Second, the GP method should help to accomplish the manipulation task. For example, robot is required to use the GP method to clear the scene for the problem of grasping in the cluttered scene [15]. We need to evaluate the performance of the GP method to accomplish the task. Third, since auxiliary actions are supposed to be performed only when needed, we should consider the action efficiency for GP with auxiliary actions.

Here, we summarize evaluation metrics and their formulation that are wildly used in grasp planning tasks.

- Grasp Success Rate (GS): the rate of successful grasp attempts in total grasp attempts. This metric reflects the grasping robustness of methods.

$$GS = \frac{\#Successful\ grasp\ attempts}{\#Total\ grasp\ attempts}$$

- Task Success Rate (Completion Rate): the average success rate over N runs. Each run is considered as a successful run if the robot successfully completes the task under some conditions, like at the end of the episode, within the maximum number of actions or less than M consecutive failed trials. This metric denotes the ability of methods to complete tasks.

$$Task\ Success\ Rate = \frac{\#Success\ runs}{\#Total\ runs}$$

- Action Efficiency (AE): the rate of the number of objects in the total number of actions executed until completion. This metric denotes the action efficiency of methods with auxiliary actions to complete tasks.

$$AE = \frac{\#objects}{\#Total\ actions}$$

IV. Conclusions and Future Directions

In this paper, we provided a brief survey on grasp planning based on deep reinforcement learning, which is summarized from two aspects: without auxiliary actions and with auxiliary actions. Since grasp planning in a close-loop manner is more robust to perceptual noises, kinematic errors as well as robust in dynamic scenes than open-loop approaches, we only reviewed the close-loop method for GP without auxiliary actions in details. However, they will fail in scenes where there are no feasible grasp poses initially, such as the object pose exceeds the gripping range of the end-effector, the target object is buried or blocked by obstacles. For non-graspable objects, GP with auxiliary actions makes reasonable decisions on performing prehensile and non-prehensile actions synergistically to generate feasible grasp poses. The detailed comparisons between them are provided. We also concluded the advantages and challenges of each class method. Furthermore, we summarize the evaluation metrics that are wildly used in grasp planning tasks.

There are hurdles of inefficient learning caused by DRL with sparse rewards and continuous state-action spaces. Generally, two solutions have been developed to alleviate the problems: using Sim2Real way or learning from expert demonstrations. However, training a 6 DoF grasp planning method is still an unsolved problem because of the gap between simulation and real robotic environment. Another challenge is the heavy price incurred by collecting demonstrations. Hence, the future directions worth considering include further exploring the ability of curriculum learning to increase transferability, collecting demonstrations autonomously without human participation, and designing a coding method for discretizing continuous actions.

In addition, since close-loop methods for GP without auxiliary actions show the advantages of grasping in dynamic scenes while methods with auxiliary actions are suitable for initially non-graspable scenes, designing a framework that combines the advantages of both types is a promising direction for solving the problem of grasping objects with non-graspable poses in dynamic scenes.

References

[1] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning," *arXiv preprint arXiv:1910.11956*, 2019.

[2] A. Zhan, P. Zhao, L. Pinto, P. Abbeel, and M. Laskin, "A framework for efficient robotic manipulation," *arXiv preprint arXiv:2012.07975*, 2020.

[3] I. Akinola, J. Varley, and D. Kalashnikov, "Learning precise 3d manipulation from multiple uncalibrated cameras," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4616–4622.

[4] Y. Yang, H. Liang, and C. Choi, "A deep learning approach to grasping the invisible," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2232–2239, 2020.

[5] G. Du, K. Wang, and S. Lian, "Vision-based robotic grasping from object localization pose estimation grasp detection to motion planning: A review," *arXiv preprint arXiv:1905.06658*, 2019.

[6] 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.

[7] B. Tang, M. Corsaro, G. Konidaris, S. Nikolaidis, and S. Tellex, "Learning collaborative pushing and grasping policies in dense clutter."

[8] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

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

[11] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke *et al.*, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *arXiv preprint arXiv:1806.10293*, 2018.

[12] Z. Chen, M. Lin, Z. Jia, and S. Jian, "Towards generalization and data efficient learning of deep robotic grasping," *arXiv preprint arXiv:2007.00982*, 2020.

[13] S. Schaal *et al.*, "Learning from demonstration," *Advances in neural information processing systems*, pp. 1040–1046, 1997.

[14] U. Viereck, A. Pas, K. Saenko, and R. Platt, "Learning a visuomotor controller for real world robotic grasping using simulated depth images," in *Conference on Robot Learning*. PMLR, 2017, pp. 291–300.

[15] S. Song, A. Zeng, J. Lee, and T. Funkhouser, "Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4978–4985, 2020.

[16] H. Liang, X. Lou, and C. Choi, "Knowledge induced deep q-network for a slide-to-wall object grasping," *CoRR*, vol. abs/1910.03781, 2019.

[17] Z. Sun, K. Yuan, W. Hu, C. Yang, and Z. Li, "Learning pregrasp manipulation of objects from ungraspable poses," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9917–9923.

[18] S. Schaal, "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics," in *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.

[19] H. Liang, X. Ma, S. Li, M. Görner, S. Tang, B. Fang, F. Sun, and J. Zhang, "Pointnetgpd: Detecting grasp configurations from point sets," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3629–3635.

[20] M. Gou, H.-S. Fang, Z. Zhu, S. Xu, C. Wang, and C. Lu, "Rgb matters: Learning 7-dof grasp poses on monocular rgbd images," *arXiv preprint arXiv:2103.02184*, 2021.

[21] R. Diankov, "Automated construction of robotic manipulation programs," 2010.

[22] C. Bodnar, A. Li, K. Hausman, P. Pastor, and M. Kalakrishnan, "Quantile qt-opt for risk-aware vision-based robotic grasping," *arXiv preprint arXiv:1910.02787*, 2019.

[23] R. Julian, B. Swanson, G. S. Sukhatme, S. Levine, C. Finn, and K. Hausman, "Never stop learning: The effectiveness of fine-tuning in robotic reinforcement learning," *arXiv e-prints*, pp. arXiv–2004, 2020.
[24] L. Hermann, M. Argus, A. Eitel, A. Amiranashvili, W. Burgard, and T. Brox, "Adaptive curriculum generation from demonstrations for sim-to-real visuomotor control," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 6498–6505.
[25] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.
[26] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1995–2003.
[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[28] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.
[29] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of operations research*, vol. 134, no. 1, pp. 19–67, 2005.
[30] R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, vol. 79, 2008.
[31] M. Gualtieri, A. Ten Pas, K. Saenko, and R. Platt, "High precision grasp pose detection in dense clutter," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 598–605.
[32] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.
[33] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.
[34] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1321–1326.
[35] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5628–5635.
[36] A. Srinivas, M. Laskin, and P. Abbeel, "Curl: Contrastive unsupervised representations for reinforcement learning," *arXiv preprint arXiv:2004.04136*, 2020.
[37] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.
[38] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp pose detection in point clouds," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1455–1473, 2017.