Learning High-DOF Reaching-and-Grasping via Dynamic Representation of Gripper-Object Interaction

QIJIN SHE*, National University of Defense Technology, China

RUIZHEN HU*, Shenzhen University, China

JUZHAN XU, Shenzhen University, China

MIN LIU, Chinese Academy of Military Science, China

KAI XU†, National University of Defense Technology, China

HUI HUANG†, Shenzhen University, China

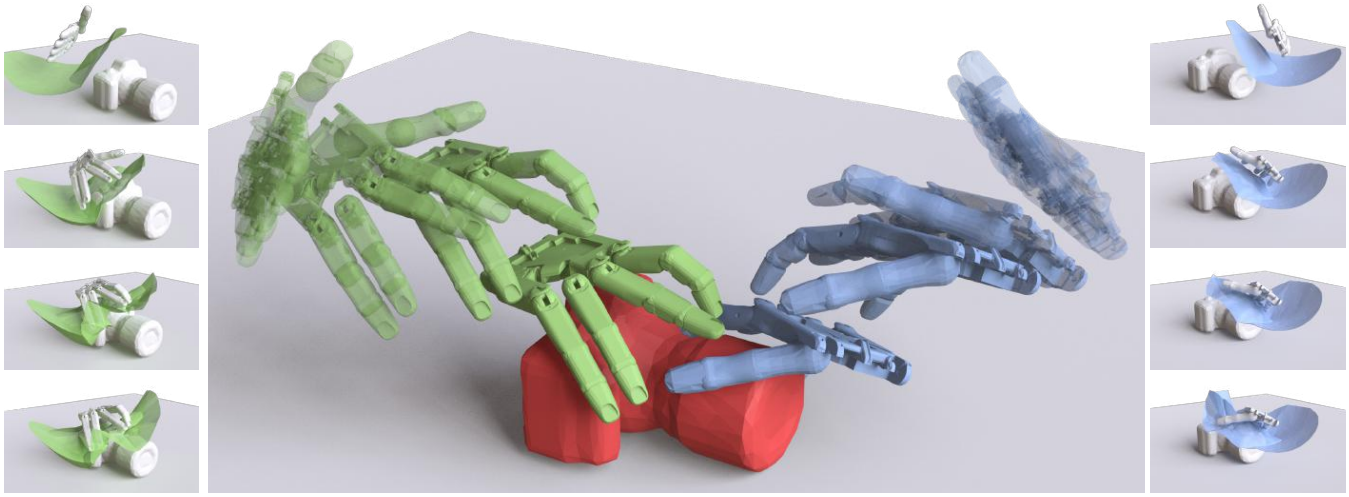


Fig. 1. Results of our reaching-and-grasping motion planning method for the red camera put on the table given two different initial configurations of the gripper shown in green and blue colors, respectively. For each result, we show four key frames of the reaching process and final grasping pose in the middle and show the corresponding Interaction Bisector Surfaces (IBSs) used to encode the interactions on the side. Note that we enlarge the distance between the gripper and the camera for the frames shown in the middle to show the gripper configurations more clearly.

We approach the problem of high-DOF reaching-and-grasping via learning joint planning of grasp and motion with deep reinforcement learning. To resolve the sample efficiency issue in learning the high-dimensional and complex control of dexterous grasping, we propose an effective representation of grasping state characterizing the spatial interaction between the gripper and the target object. To represent gripper-object interaction, we adopt Interaction Bisector Surface (IBS) which is the Voronoi diagram between two close by 3D geometric objects and has been successfully applied in characterizing spatial relations between 3D objects. We found that IBS is surprisingly effective as a state representation since it well informs the fine-grained control of each finger with spatial relation against the target object. This novel grasp representation, together with several technical contributions including a fast IBS approximation, a novel vector-based reward and an effective training strategy, facilitate learning a strong control model of high-DOF grasping with good sample efficiency, dynamic adaptability, and cross-category generality. Experiments show that it generates high-quality dexterous grasp for complex shapes with smooth grasping motions.

*Joint first authors.
†Corresponding authors.

Authors' addresses: Qijin She, qijinshe@outlook.com, National University of Defense Technology, China; Ruizhen Hu, ruizhen.hu@gmail.com, Shenzhen University, China; Juzhan Xu, juzhan.xu@gmail.com, Shenzhen University, China; Min Liu, gfsliumin@gmail.com, Chinese Academy of Military Science, China; Kai Xu, kevin.kai.xu@gmail.com, National University of Defense Technology, China; Hui Huang, hhzhiyan@gmail.com, Shenzhen University, China.

XXXX-XXXX/2022/4-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

CCS Concepts: • **Computing methodologies** → **Computer graphics**; **Shape modeling**; **Mesh geometry models**.

ACM Reference Format:

Qijin She, Ruizhen Hu, Juzhan Xu, Min Liu, Kai Xu, and Hui Huang. 2022. Learning High-DOF Reaching-and-Grasping via Dynamic Representation of Gripper-Object Interaction. 1, 1 (April 2022), 14 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Robotic grasping is an important and long-standing problem in robotics. It has been drawing increasingly broader attention from the fields of computer vision [Saxena et al. 2010], machine learning [Kleeberger et al. 2020] and computer graphics [Liu 2009; Pollard and Zordan 2005], due to the interdisciplinary nature of its core techniques. Traditional methods typically approach grasping by breaking the task into two stages: static grasp synthesis followed by motion planning of the gripper. In the first stage, a few candidate grasp poses are generated. The second stage then plans a collision-avoiding trajectory for the robotic arm and gripper to select a feasible grasp and execute it. The main limitation of such a decoupled approach is that the two stages are not jointly optimized which could lead to suboptimal solutions.

An integrated solution to grasp planning and motion planning, which is often referred to as the *reach-and-grasp problem*, remains a challenge [Wang et al. 2019]. A few works have attempted integrated grasp and motion planning by formulating a joint optimization problem. The advantage of an integrated planner is that motion planning and grasp planning impose constraints on each other. However, these existing methods still rely on pre-sampled grasp candidates over which a probabilistic distribution for selection is computed, making it highly reliant on the quality of the candidates. Wang et al. [2019] introduce online grasp synthesis to eliminate the need for a perfect grasp set and grasp selection heuristics. Nevertheless, such an approach optimizes over a discrete set of grasp candidates, which limits the grasping space explored.

Reinforcement learning (RL) [Sutton and Barto 2018] models offer a counterpoint to the planning paradigm. Rather than optimizing for grasp selection and motion planning, the idea is to use closed-loop feedback control based on sensory observations so that the agent can dynamically update its strategy while accumulating new observations. More recent advances of RL allow for continuous, high-dimensional actions which are especially suitable for continuous exploration of reach-and-grasp planning. Albeit offering promising solutions, the sample efficiency issue of RL hinders its application in highly complex control scenarios, such as dexterous grasping of a high-DOF robotic hand (e.g., a 24-DOF five-fingered gripper).

We argue that the main cause of the limitation above is the lack of an effective representation of observations. Indeed, even with deep neural networks as powerful function approximators, it is still too difficult to fit a function mapping raw sensory observations (camera images) to low-level robot actions (e.g., motor torques, velocities, or Cartesian motions). Therefore, learning complicated control of high-DOF grippers calls for an informative representation of the intermediate states during the reaching-and-grasping process. Such representation should well inform the RL model about the *dynamic interaction* between the gripper and the target object.

In this work, we advocate the use of *Interaction Bisector Surface (IBS)* for representing gripper-object interaction in RL-based reach-and-grasp learning. IBS, computed as the Voronoi diagram between two geometric objects, was originally proposed for indexing and recognizing inter-protein relation in the area of biology [Kim et al. 2006]. In computer graphics, it has been successfully applied in characterizing fine-grained spatial relations between 3D objects [Hu et al. 2015; Zhao et al. 2014]. We found that IBS is surprisingly effective as a state/observation representation in learning high-DOF reach-and-grasp planning. Gripper-object IBS well informs the global pose of the gripper and the fine-grained local control of each finger with spatial relation against the target object, making the map from observation to action easier to model. For different initial configuration of the gripper, i.e., different relative pose to the object, our method is able to provide different motion sequences and form different final graspings as shown in Figure 1. Moreover, during the reaching-and-grasping process, the dynamic change of relationship between the gripper and object can be well-reflected by the corresponding IBS, which enables our method to deal with moving objects with dynamic pose change, going beyond static object grasping of most previous works. In addition, as IBS is defined purely based on the geometry of the gripper and the object without any assumption of the object semantics or affordance, our method can generalize well to objects of unseen categories.

To speed up the computation of IBS for efficient training and testing, we propose a grid-based approximation of IBS as well as post-refinement mechanism to improve accuracy. Empirical studies show that the approximation is accurate to capture the important interaction information and fast enough to enable online computation in an interactive frame rate. To capture richer information of interaction, we propose a combination of local and global encoders for multi-level feature extraction of IBS, based on its segmentation corresponding to the different components of the gripper.

We adopt Soft Actor-Critic (SAC) [Haarnoja et al. 2018], an off-policy model, as our RL framework. Aside from the core design of state representation, we introduce two other critical designs to make our model more sample efficient and easier to train. To learn collision-avoiding finger motions, we impose finger-object contact information as a constraint of RL. A straightforward option is to design the reward to punish finger-object penetration. This can greatly complicate model training. Thus, our *first key design* is to enhance the standard scalar Q value into a vector storing finger-wise contact information. Such disentangled Q representation provides a more informative dictation on learning contact-free motions.

To deal with the continuous action space in grasp planning, we opt to learn from imperfect demonstrations synthesized offline with heuristic planning from different initial configurations to the final grasp poses generated GraspIt! [Miller and Allen 2004]. The demonstration grasping trajectories, possibly containing gripper-object penetration, are stored in the experience replay buffer [Mnih et al.

2015] of SAC. Our *second key design* is to bootstrap the learning with a bootstrapping replay buffer containing imperfect demonstrations possibly with collision. We then inject the replay buffer with an increasing amount of experiences sampled from the currently learned policy and rectified to be collision-free. This forms an enhanced replay buffer based on which a gradually refined policy can be learned. Such a double replay buffer scheme helps learn a strong control planning model fairly efficiently.

Our method generates high-quality dexterous grasps for unseen complex shapes with smooth grasping motions. Furthermore, our method can dynamically adjust and adapt to object movement during grasping, thus allowing to grasp moving objects. When compared to other baseline methods, our method consistently achieves a higher success rate on different datasets. Our method is also robust to partial observations of target objects. Our contributions include:

- A novel state representation for learning reach-and-grasp planning based on gripper-object interaction bisector surface, along with an accurate approximation for fast training.
- A combination of local and global encoders for multi-level feature extraction of IBS to capture richer information of gripper-object interaction.
- A new vector-based representation of Q value encoding not only regular rewards but also finger-wise contact information for efficient learning of contact-free grasping motion.
- A double replay buffer mechanism for learning collision-avoiding grasping policy from imperfect demonstrations.

2 RELATED WORK

Robotic grasping has a large body of literature. Existing approaches can generally be classified into analytical and data-driven methods. Analytical (or geometric) approaches analyze the shape of the target object to synthesize a suitable grasp [Sahbani et al. 2012]. Data-driven (or empirical) approaches based on machine learning are gaining increasing attention in recent years [Bohg et al. 2013; Kleeberger et al. 2020] .

Analytical robotic grasping. With known object shapes, analytical methods search for grasp poses that maximize a certain grasp quality metric, and these methods can mainly be classified into discrete sampling-based techniques or continuous optimization techniques. Some sampling-based techniques search in the space of grasp poses [Miller and Allen 2000, 2004], while others search contact points or contact areas on surfaces of objects and then searches for collision-free grasp poses that realize the given set of contact points or areas [Chen and Burdick 1993; Hang et al. 2017; Liu et al. 2020a; Pan et al. 2022]. Compared to sampling-based techniques, continuous optimization techniques plan grasp poses by optimizing differentiable losses [Kiatos and Malassiotis 2019; Maldonado et al. 2010], which are more efficient. Recent work [Liu et al. 2020b] proposes a differentiable grasp quality, which can be used for continuous optimization and deep learning methods. For high-DOF grippers, however, sampling-based techniques are computationally costly due to the large search space. Continuous optimization techniques for high-DOF grasp planning apt to find local optimal grasp poses. Moreover, analytical methods are difficult to generalize to incomplete or unknown objects.

Learning-based robotic grasping. Learning-based methods are typically split into supervised learning and reinforcement learning. For supervised learning, grasp annotations can be collected either by humans [Depierre et al. 2018], with simulation [Mahler et al. 2016], or through real robot tests [Levine et al. 2018]. Supervised grasp learning can be categorized as discriminative or generative depending on whether the grasp configuration is the input or output of the learned model. While discriminative approaches sample grasp candidates and rank them using a neural network [Mahler et al. 2018], generative approaches directly generate suitable grasp poses [Morrison et al. 2018].

Early learning-based works mainly focus on generating grasps for target objects with low-DOF grippers (such as parallel-jaw grippers) [Gualtieri et al. 2016; Saxena et al. 2007]. Such work learns to regress grasp quality or to predict grasp success [Fang et al. 2018; Lu et al. 2020a,b; Mahler et al. 2017, 2016; Van der Merwe et al. 2019], but still needs sampling-based techniques to search for better grasps. Liu et al. [2019; 2020b] use deep neural networks to directly regress high-DOF grasps based on the input of voxels or depth images.

Reinforcement learning for robotic grasping. Deep reinforcement learning (RL) has been shown as a promising and powerful technique to automatically learn control policies by trial and error. Based on raw sensory inputs, dexterous grasping behaviors can be performed. A comparative study of RL-based grasping methods is given in [Quillen et al. 2018]. QT-Opt [Kalashnikov et al. 2018] learns various manipulation strategies with dynamic responses to disturbances. Song et al. [2020] present an RL-based closed-loop 6D grasping of novel objects with the help of human demonstrations. The learned policy can operate in dynamic scenes with moving objects. Rajeswaran et al. [2017] show that model-free RL can effectively scale up to complex manipulation tasks with a high-DOF hand. Mandikal and Grauman [2021] introduce an approach for learning dexterous grasps, and the key idea is to embed an object-centric visual affordance model within a deep reinforcement learning loop to learn grasping policies that favor the same object regions favored by people. Andrychowicz et al. [2020] explore RL for dexterous in-hand manipulation by reorientating a block. Starke et al. [2019] present a good example of conditioning motion behavior on the geometry of the surrounding 3D environment. Ficuciello et al. propose methods that use RL to search good grasp poses [2016] as well as good grasp motion trajectories [2019] in a synergies subspace, and these methods need good initial parameters to reduce the searching space while finding initial parameters need additional imitation learning or human efforts. To overcome this limitation, Ficuciello et al. [2019] introduce a visual module to predict initial parameters given visual information of objects. The main weakness of their methods is that the learning process should be done for every object respectively to achieve a good grasp.

Training RL models in the real environment is usually prohibitive since it requires a large number of trials and errors. A straightforward approach is to train them in a simulation environment and then transfer the learned policy to the real world with sim-to-real

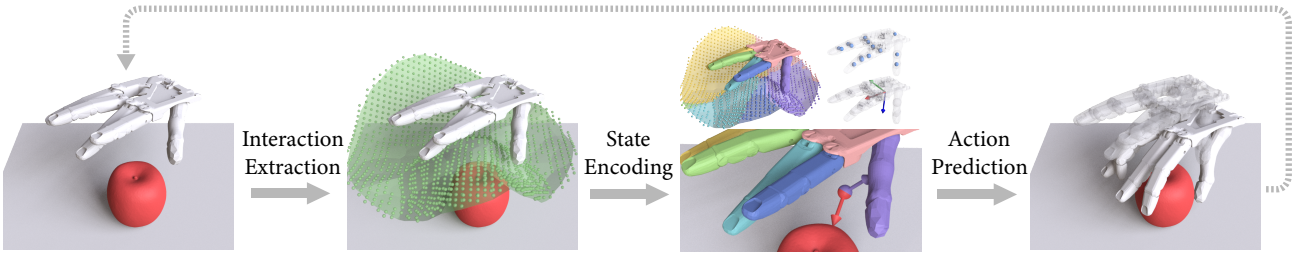


Fig. 2. Overview of one iteration of our grasping motion planning method. Given an object in a scene context with the current gripper configuration, our method first generates the sampled IBS to represent the interaction between the scene and gripper, then a set of local and global features are extracted from the given state to predict the action that changes the configuration of the gripper so that it moves closer to the given object and forms a better grasping. The updated configuration after applying the predicted action is then passed through the same pipeline to predict the subsequent action.

techniques [James et al. 2019; Peng et al. 2018]. Our work trains an RL model for dexterous high-DOF grasping. We focus on how to train such complicated planning policies with the help of an effective representation of gripper-object interaction and leave the issue of sim-to-real transfer for future work.

Grasp representation. Existing grasping models have adopted various representations to describe the shape of the target object to be grasped, such as voxels [Varley et al. 2017], depth images [Viereck et al. 2017], multi-view images [Collet and Srinivasa 2010], or geometric primitives [Aleotti and Caselli 2012]. Some works represent a grasp with Independent Contact Regions (ICRs) [Fontanals et al. 2014; Roa and Suárez 2009]. These regions are defined such that if each finger is positioned on its corresponding contact region, a force-closure grasp [Nguyen 1988] is always obtained, independently of the exact location of each finger.

Our work opts to characterize the interaction between the gripper and the object using Interaction Bisector Surface (IBS) [Zhao et al. 2014]. Interaction Bisector Surface (IBS) captures the spatial boundary between two objects. It can provide a more detailed and informative interaction representation with both geometric and topological features extracted on the IBS. Hu et al. [2015] further combined IBS with Interaction Region (IR), which is used to describe the geometry of the surface region on the object corresponding to the interaction, to encode more geometric features on the objects. Pirk et al. [2017] build a spatial and temporal representation of interactions named interaction landscapes.

Karunratanakul et al. [2020] proposes an implicit representation to encode hand-object grasps by defining the distance to human hand and objects for points in space, and then use it to generate the grasping hand pose for a given static shape. In contrast, our method utilizes constantly updated IBS as state representation to plan the dynamic motion of the gripper for the object reach-and-grasp task.

3 OVERVIEW

Given the input scene segmented into foreground object and background context and the initial configuration of the gripper, our goal is to output a sequence of collision-free actions which move the gripper towards the object to perform a successful grasping. We train a deep reinforcement learning model in which the state is given by the gripper-object interaction and the actions are designed as the gripper configuration changes. Figure 2 gives an overview of one step of the planned reaching-and-grasping motion.

Our method starts by extracting an informative representation of the interaction between the given object and the current gripper configuration. A set of local and global features are extracted from the given state to predict the action that changes the configuration of the gripper so that it moves closer to the given object and forms a better grasping. The updated configuration after applying the predicted action is then passed through the same pipeline to predict the subsequent action.

Interaction extraction. We use IBS [Zhao et al. 2014] to represent the interaction. Since the computation of accurate IBS is time-consuming, we design an IBS sampler to obtain a discretized and simplified version of IBS in a much more efficient manner. The set of sampled IBS points is then moved to be closer to the exact IBS.

State encoding. The gripper configuration is encoded by its global position and orientation as well as local joint angels with (6+18)-DOF. Based on the part components of the gripper model, i.e., five fingers and one palm, we adopt a multi-level representation of IBS inspired by the work of Zhao et al. [2017]. More specifically, for each sampled IBS point, we first find its nearest points on the scene and the gripper, and then encode it with a set of information, consisting of its own coordinate, the component labels of those two nearest points as well as the spatial relationship relative to each nearest point. Therefore, the components of the gripper model, including five fingers and one palm, naturally induce a segmentation of the IBS based on the association between the gripper and the IBS. We then combine local features extracted separately for each IBS segment and global feature for the entire IBS to form a multi-level description of gripper-object interaction.

Action prediction. To predict the action given the current state, we design a policy network that takes both local and global features from the current configuration and outputs the configuration change of the gripper and terminate value. The policy network

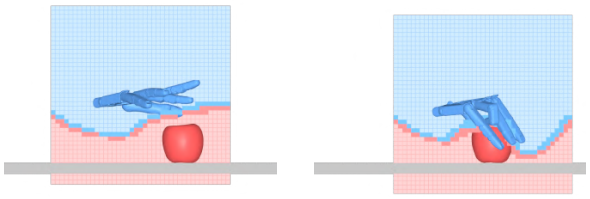


Fig. 3. Examples of IBS sampled in two intermediate states when the gripper moves towards the object. For each state, we only show one vertical slice of the grids around the gripper to view the change of the IBS more clearly. Grids that are closer to the object are colored in red, and others are colored in blue. The IBS by definition is located on the boundary of those two colored regions, and we take the grid points in blue cells on the boundary to be the set of sampled IBS points.

is trained via reinforcement learning using the Soft Actor-Critic method [Haarnoja et al. 2018]. To avoid gripper-scene collision and self-collision of the gripper during the reaching-and-grasping process, we design a new vector-based representation of Q value encoding not only regular rewards but also finger-wise contact information for efficient learning of contact-free grasping motion. To further accelerate the training, we generate a set of imperfect demonstration data to bootstrap the learning and to converge to the final policy in a more efficient and effective manner.

4 METHOD

4.1 IBS sampler

The IBS is essentially the set of points equidistant from two sets of points sampled on the scene and the gripper, respectively. The computation of exact IBS requires the extraction of the Voronoi diagram, which is time-consuming. To trade-off between efficiency and accuracy, we compute IBS only within a given range and discretize the space to obtain an approximation of IBS, as shown in Figure 3.

More specifically, IBS is only computed within the sphere which is located at the centroid of the palm with a radius of r. The bounding box of the sphere is discretized into a k^3 grid for IBS point sampling. For each of the grid points, i.e., the center of each grid cell, we compute its distances to both the scene d_s and the gripper d_g, and the points having $\delta = d_g - d_s = 0$ are IBS points. To find such set of points, we store the δ values on grid cells and extract the zero-crossing points on the fly. Note that to accelerate the whole process, the δ values on only a partial set of cells around the exact IBS are computed in a region growing manner. More specifically, we first find the cell that is closest to the middle position of the line connecting the centroid of the foreground object in the scene and the palm, and compute its δ value. Then the computation grows outwards with the neighboring cells until sufficient grid cells that share the same sign have been found.

Figure 3 shows the sampling process of IBS in two different states. We can see that the grid is divided by IBS into two parts, i.e., one with points closer to the scene (in red color) and the other with points closer to the gripper (in blue color). Only the δ values on the grid cells close to the exact IBS are computed to get the initial set of sampled IBS points (highlighted with solid colors). Note that such set of sampled IBS contains two layers of grid points which has redundant information, thus we only keep the layer corresponding to the gripper as our IBS point set (the solid layer in blue).

Since the initial set of sampled IBS points are grid centers and the approximation accuracy is highly affected by the grid resolution, we further refine the locations of these points to make them approach the exact IBS. For each point p, we first locate its nearest points on scene p_s and gripper p_g. Without loss of generality, let us assume that the distance d_g between the point to the gripper is larger than the distance d_s between the point and the scene. We then need to move p towards p_g to make it closer to the exact IBS: $p' = p + \Delta_p \times \overrightarrow{pp_g}$. The detail of how to derive an appropriate Δ_p can be found in the supplementary material. Once the location of point p is updated, we update its nearest points on the scene and on the gripper.

To gauge how close the sampled IBS is to the exact IBS as well as the effectiveness of our refinement method, we measure the IBS approximation error using the Chamfer distance between the points sampled from the grids and those sampled on the exact IBS surface. Figure 4 shows how the Chamfer distance changes during the reach-and-grasp process before and after the refinement under different grid resolutions $k = 20$, 40, and 80. We can find several interesting properties. Firstly, approximation error is relatively stable during the reach-and-grasp process for each setting and gets slightly higher towards the end as the IBS surface becomes more complex. Secondly, approximation error drops significantly after the refinement for different grid resolutions, which shows the effectiveness of the refinement process. Thirdly, approximation error generally increases with the decrease of grid resolution, especially when no refinement is involved. However, the computation time increases with the grid resolution. To obtain a good balance, we set grid resolution $k = 40$ and sample $n = 4096$ points for IBS approximation.

4.2 State and action representation

State representation. Finding an informative representation for a given state is the key to guiding the movement of the gripper towards a successful grasping of the object. Our key observation is that the IBS between the scene and gripper together with the gripper configuration provide rich information about the intermediate state.

For gripper configuration, we use a (6+18)-DOF Shadow Hand in all our experiments, where the first 6-DOF encodes the global orientation and position of the gripper and the remaining 18-DOF encodes the joint angles. To better describe the local context around the gripper to guide its configuration change, we set the origin of the world coordinate to the centroid of the palm, as the setting in the IBS sampling process, to encode the spatial interaction features between the gripper and the scene. We found that our method gets similar performance when using the centroid of the object as the origin of the world coordinate.

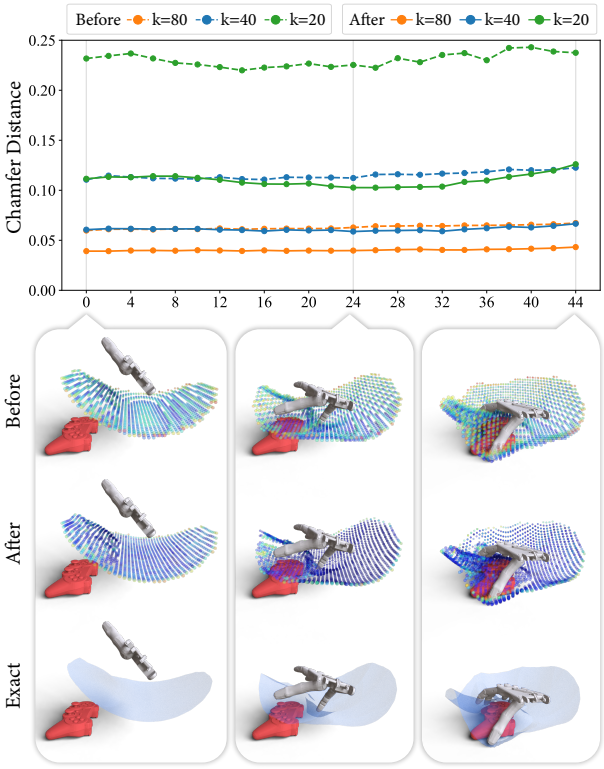


Fig. 4. IBS approximation error during the reaching-and-grasping process before and after the refinement under different grid resolutions $k = \{20, 40, 80\}$. The line chart shows how the approximation error changes during the process under different settings, and below we show the corresponding IBS samples before and after refinement compared to the exact IBS surface for three representative frames under $k = 40$. We use the jet colormap on the sampled points to indicate the approximation error, which shows that the approximation accuracy is highly improved after refinement.

For each point p on the sampled IBS, we store the following information as illustrated in Figure 5:

- coordinate $c = (x, y, z) \in R^3$
- distance to the scene $d_s^p \in R$
- unit vector pointing to the nearest point p_s on the scene $v_s^p \in R^3$
- indicator of whether p_s is located on the foreground object $b_s^p \in \{0, 1\}$
- distance to the gripper $d_g^p \in R$
- unit vector pointing to the nearest point p_h on the gripper $v_g^p \in R^3$
- one-hot indicator of the gripper component that p_g belongs to $c_g^p \in \{0, 1\}^6$
- value defined on p_g indicating which side of the gripper it located on $a_g^p \in [-1, 1]$

Here, $a_g^p = n_p \cdot d_{up}$ is the dot product of the normal direction n_p of point p_g on gripper in rest pose and the upright direction d_{up} perpendicular to the palm and pointing outwards.

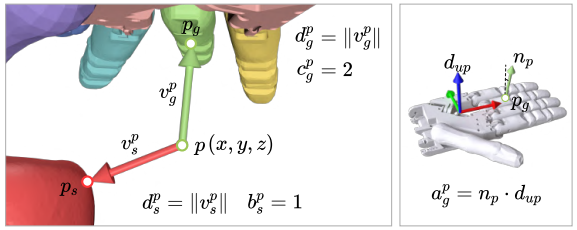


Fig. 5. Informative representation of each sampled IBS point p, which includes a set of relative spatial information to either the scene or the gripper. A more detailed explanation of the notations can be found in Section 4.2.

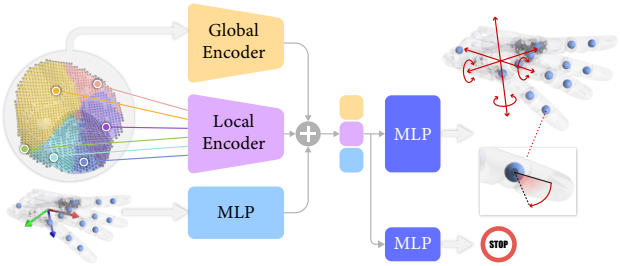


Fig. 6. Grasp planner network used in our work. Given the rich information stored in the sampled IBS and current gripper configuration, we use both global and local encoders to extract multi-level features from the sample IBS and then concatenate them with the feature extracted from the gripper via MLP. The concatenated feature is passed to two other MLPs to get the final predicted action, one of which is defined as the global translation and rotation of the gripper as well as the local joint rotation, and the other is defined as the termination probability.

Action representation. The action consists of two parts. The first part is defined as the gripper configuration change, which is also with (6+18)-DOF. For each single action, we restrain the change of each parameter within $[-0.25cm, 0.25cm]$ for global translation of the entire gripper, and $[-0.025, 0.025]$ in radian for both global rotation angle of the entire gripper or local rotation angle of each joint. The other part is the stop action, defined as a termination value to indicate the possibility of terminating the planning process. The exact termination mechanism is provided in Section 4.3. .

4.3 Network and reward design

Network architecture. Figure 6 shows the network architecture of our grasp planner (actor network). For the sampled IBS, both local and global features are extracted to concatenate with the feature extracted from the gripper configuration, and then the concatenated feature is passed to two other MLPs to get the predicted action, one for the gripper configuration change and the other for the terminal value. We use PointNet [Qi et al. 2017] for both local and global IBS encoder, and the global encoder takes the whole IBS as input while the local encoder takes each IBS component corresponding to different gripper components as input.

Reward function. The reward function needs to reflect the quality of an executed action. As our final goal is to perform a successful grasping of the given object in the scene at the end of the planning, we first define a *grasp reward* function R_g to measure the grasp quality when the whole process is terminated. Moreover, to further encourage a more natural grasp pose with more gripper components taking part in the grasping and avoiding collision with the scene during the whole process, we define another *reaching reward* function R_c^i per gripper component g_i to provide guidance for each intermediate step.

We measure the grasp quality from two aspects. First is the commonly used execution success, which we use the success signal S obtained from the Pybullet simulator when performing the final grasp. We consider the grasp to be successful if and only if the object can be lifted up by more than 0.2m as in the previous work [Xu et al. 2021]. More details about the simulator setup and how to perform the grasp can be found in the supplementary material. However, this sparse boolean value cannot provide enough guidance for high-DOF grasping, thus we complement it with another well-known geometric measure Q_1 [Ferrari and Canny 1992]. But as the traditional Q_1 measure can only be computed when the gripper touches the object without any collision and the computation becomes unstable during the training, we adopt the generalized Q_1 measure proposed in [Liu et al. 2020b] instead.

As those two grasp quality metrics are only computed when the reach-and-grasp task is completed, to encourage a quick convergence, we set a negative reward r_f for each intermediate step. So the grasp reward function is defined as:

$$R_g = \begin{cases} \omega_s S + \omega_q Q_1, & \text{if the task is completed;} \\ r_f, & \text{otherwise.} \end{cases}$$

where we set $\omega_s = 150$, $\omega_q = 1000$, and $r_f = -3$ in all our experiments. To encourage the contact between the gripper and the object, our planning process terminates requires that not only if a randomly sampled value is smaller than the terminal value but also at least two gripper components contact the object in the current step.

To determine whether a gripper component g_i is contacting with the object, we would like to ensure that there are enough points on the gripper component that are close enough to the object while not colliding with the scene. So we first count the number m_i of IBS points determined together by the inner side of the gripper component g_i ($c_n^p = i, a_g^p \geq 0$) and the object ($b_s^p = 1$) with distance to the object smaller than a given threshold $\delta_d = 0.5cm$ ($d_s^p < \delta_d$), i.e., IBS points with $b_s^p = 1, c_n^p = i, a_h^p \geq 0, d_s^p < \delta_d$. Then we further check if any of the IBS points determined by the gripper component g_i (i.e., IBS points with $c_n^p = i$) is on the inner side of the scene by computing the angle between the corresponding vector v_s^p pointing from the IBS point to the nearest point on the scene and the normal n_s^p of the nearest point p_s. If the angle is smaller than 90 degrees, we consider the IBS point is on the inner side of the scene, and count the number n_i of such IBS points. If $n_i \geq \delta_n$, we consider the gripper component is colliding with the scene. Therefore, if the gripper component g_i is not colliding with the scene ($n_i < \delta_n$) and there are enough contacting points ($m_i \geq \delta_m$), we consider the gripper component is contacting the object.

Thus we define the reaching reward function R_c^i per gripper component g_i as follows to encourage a more effective reaching-and-grasping process with more contacting but not colliding points:

$$R_c^i = \begin{cases} -100, & n_i \geq \delta_n (\text{collide with the scene}); \\ R^i_{\text{contact}}, & \text{otherwise.} \end{cases}$$

$$R^i_{\text{contact}} = \begin{cases} \min\{\eta, m_i\}, & m_i \geq \delta_m (\text{have enough contact}); \\ 0, & \text{otherwise.} \end{cases}$$

In all our experiments, we set $\delta_n = \delta_m = 3$ and $\eta = 40$.

4.4 Network training

To train the network, we adopt the well-known off-policy method Soft Actor-Critic [Haarnoja et al. 2018] and make some small modifications to the Q-network to make the training more effective.

SAC has two networks, i.e., the policy network (also known as the actor network) and the Q-network (also known as the critic network). The policy network outputs a Gaussian distribution $\pi(*|s;\theta)$ to sample action for an input state s, where θ are the parameters of the network. Q-network outputs the evaluation value $Q(s,a;\Phi)$ for given part state s and action a, where Φ are the parameters of the network. SAC uses an addition target Q-network to calculate target value for temporal difference (TD) update, whose parameters are denoted by Φ'. A transition is denoted as a tuple $\{s, a, R, s', d\}$, where R is the reward and d indicates whether state s' is a terminated state. All the transitions will be stored in a replay buffer D and those two networks are trained by the data sampled from D.

The key change to the original SAC networks is that instead of outputting a single scale value to estimate the reward, we let the Q-network output a vector $(Q_g, Q_c^0, \ldots, Q_c^5)$ to estimate both R_g and R_c^i. Accordingly, the reward R in each experience is a vector of $(R_g, R_c^0, \ldots, R_c^5)$. Note that only R_g is accumulated while R_c^i is computed for each single step. We found that this change made the training more stable and prevented the collision more effectively than directly combining all rewards together.

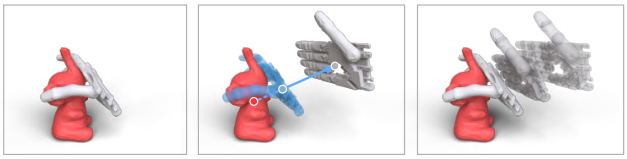
The loss function for training the Q-network is then determined by temporal difference (TD) update:

$$L_Q(\Phi) = [(Q_g(s,a;\Phi) - y_g(R_g, s', d))^2 + \sum_{i=0}^{5} (Q_c^i(s,a;\Phi) - \lambda R_c^i)^2)] \tag{1}$$

with $\lambda = 0.25$ balancing the two type of rewards and target value y_g for R_g defined as:

$$y_g(R_g, s', d) = R_g + \gamma(1-d)\ [Q_g(s', \widetilde{a}'; \Phi')\ - \alpha \log \pi(\widetilde{a}|s';\theta)], \tag{2}$$

where $\widetilde{a}' \sim \pi(*|s';\theta))$ is the sampled action. $\gamma = 0.99$ is the discount factor, and α is the temperature parameter that can be adjusted automatically to match an entropy target in expectation, to balance exploring the environment and maximizing reward.



(a) valid grasping (b) init configuration (c) demonstration

Fig. 7. Method for demonstration generation. Given a valid grasping (a), we first push the gripper away from the object along the direction pointing from the object center to the palm center to reach a given distance, and then reset the gripper configuration with some random Gaussian noise to get the initial gripper configuration (b). The final demonstration (c) is generated by first moving the gripper close enough to the final position and then transiting to the final grasping configuration using linear interpolation.

For the policy network, in order to avoid self-collision of the gripper, i.e., the collision among different gripper components, we add a self-collision loss L_{self} adapted from [Liu et al. 2020b] to the original loss:

$$L(\theta) = L_Q(\theta) + \omega L_{\text{self}}(\theta), \tag{3}$$

where $\omega = 100$ is the parameter to balance the two loss terms. The original loss function $L_Q(\theta)$ and the self-collision loss L_{self} for training the policy network are defined as:

$$L_Q(\theta) = [Q_g(s, \widetilde{a}(s;\theta)) + (\sum_{i=0}^{5} Q_c^i(s, \widetilde{a}(s;\theta)) - \alpha \log \pi(\widetilde{a}|s;\theta)], \tag{4}$$

$$L_{\text{self}}(\theta) = \sum_{i=1}^{L}\sum_{j=1}^{N} \max(D(p_j(s, \widetilde{a}(s;\theta)), H_i(s, \widetilde{a}(s;\theta))), 0), \tag{5}$$

where $\widetilde{a} \sim \pi(*|s;\theta))$ is the sampled action based on current state and network parameters, L the number of gripper links, N the number of points $p_j(s;\theta)$ sampled from each link, $H_i(s;\theta)$ the convex hull of each link after performing action $\widetilde{a}$ on state s, and D the signed distance from a point to a convex hull. More details about the self-collision loss L_{self} can be found in [Liu et al. 2020b]. The reason why we add the self-collision loss directly for the policy network instead of defining a corresponding reward function is that the self-collision loss is differentiable and thus can be optimized directly.

Training with demonstration. Note that the searching space of the action is extremely large and to make the training more efficient, we adopt the popular training with demonstration strategy. To generate the demonstrations, as shown in Figure 7, we first generate a valid grasp pose for the given object, and then move the gripper away from the object along the direction pointing from the object center to the palm center until the distance reaches $d = 20cm$, and then we reset the gripper configuration by setting all joint angles to be zero and add some random Gaussian noise scaled based on the rotation limit of each joint to the gripper configuration to generate a random initial configuration. After getting the initial configuration of the gripper, to generate a motion sequence towards the corresponding final grasp pose and use it as the demonstration, we simply first move the gripper to the final position and then transit to the final grasping configuration using linear interpolation.

Note that the demonstration generated in this way is imperfect since the gripper may collide with the scene during the whole process, so unlike previous methods that usually use imitation learning for behavior cloning of the perfect demonstrations, we store the generated imperfect demonstration into the bootstrapping replay buffer and use reinforcement learning only.

In more detail, we use two replay buffers, one for demonstration data with maximal size set to be $n_d = 5.0 \times 10^4$ and the other for self-exploration data with maximal size set to be $n_s = 1.0 \times 10^5$. Before training, we always fill up the demonstration buffer and keep counting the total number n_t of data generated in those two ways. The probability of sampling data from the demonstration buffer is set to be n_d/n_t. Thus, more demonstration data will be sampled in the beginning to guide the network to learn a good initial policy quickly and hence speed up the training process. Following [Vecerik et al. 2017], these samples are included in the update of both the actor and the critic.

5 RESULTS AND EVALUATION

We first explain the experiment setup, the dataset we used, and then evaluate our method both qualitatively and quantitatively.

5.1 Data preparation

We first adopt the dataset provided by Liu et al. [2020b], which consists of 500 objects collected from four datasets. We use the objects from KIT Dataset [2012] and GD Dataset [2015] as training data, and then test objects from YCB Dataset [2017] and BigBIRD dataset [2014]. We then further test our method on more objects from ContactPose Dataset [2020] and from 3DNet Dataset [2012] when compared to baseline methods.

To generate the demonstrations to guide the training of our network, we randomly select 100 grasp poses for each object in the training set from the pose set associated with that object, and synthesize imperfect reach-and-grasp motion as described in Section 4.4. Note that since each object is placed on the table in our setting, we need to first filter out invalid grasp poses which collide with the table.

To generate our self-exploration data, we need to sample the initial configuration of the gripper. For each object, we use its center to create a sphere with radius $r = 20cm$, and sample points on the upper hemisphere as the origin of the local coordinate system of the gripper and rotate the gripper to make its palm face the object center and its thumb point upwards.

As different initial configurations will lead to different reach-and-grasp results, to remove the bias of such initialization, we set a fixed set of initial configurations for each object to test. The initialization details can be found in the supplementation material.

5.2 Qualitative results

Figure 8 shows the gallery of results obtained with our method, where we show the initial input configuration of the gripper and

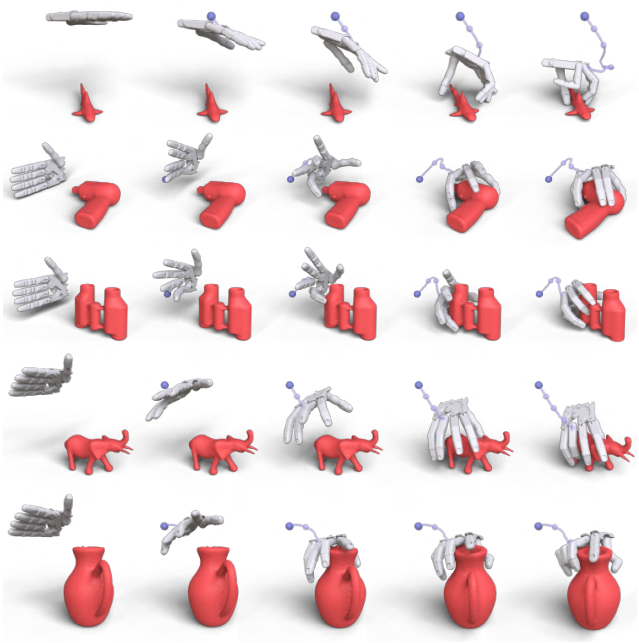


Fig. 8. Gallery of results obtained with our method, where we show the input initial configuration of the gripper and four sampled frames during the approaching process with the final grasping pose on the right. The moving trajectory of the whole reach-and-grasp process is shown with the purple curve for each example.

Fig. 9. Example results where we fix the shape and plan the grasping with different initial gripper configurations. For each example, we show all four different initial configurations on the aligned hemisphere with purple dot on the left, and then their final grasping poses on the right.

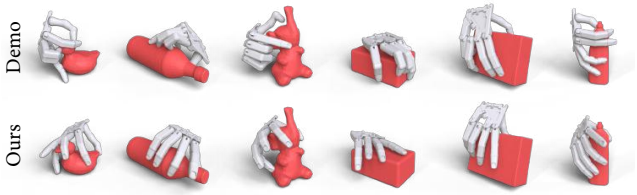


Fig. 10. Visual comparisons of the final grasps generated by our method to the corresponding demonstration with the same object and same initial gripper configurations used for training. Note how our method is able to generate different and more natural grasp poses.

four sampled frames during the approaching process with the final grasp pose on the right. Note that each example is shown with a view we selected to demonstrate the motion sequence more clearly, and we further add a purple curve to visualize moving trajectory together with sampled frames. When inspecting these results, we see that the method is able to handle various shapes given different initial configurations of the gripper. For example, for the small shark model shown in the first row, our method can adjust the gripper pose and joints precisely to pinch the model, while for the model shown in the second row, although the gripper starts from the position close to the table, our method is able to generate the moving sequence with the final grasp on the top of the model to avoid collision with the table. Our method is also able to deal with objects with more complex geometry. For example, to grasp the binoculars shown in the third row and the elephant shown in the fourth row, those four fingers tend to move together to form the gripping grasp with the thumb, while for the pitcher shown in the last row, fingers spread widely to better wrap its top.

Moreover, Figure 9 shows examples of results where we fix the shape and plan the grasping with different initial gripper configurations. For each example, we show four different initial configurations on the aligned hemisphere with purple dots on the left, and then their corresponding final grasp poses on the right. We see that various grasp poses can be generated even for the same shape when given different initial configurations. For example, for the elephant model shown in the first row, some final grasp poses tend to cover the head of the elephant while some others prefer wrapping the back of the model. Similar results can also be observed for the shoe model shown in the second row. For these models with far more complex geometry than the models we have in the training set, the final grasp poses obtained by our method can adapt well to their shapes while avoiding collision with the table at the same time. Overall, our method is quite robust and can successfully plan grasping for different shapes with various geometries when given different initial gripper configurations.

Note that all the objects tested in our experiments are unseen objects from different datasets, which shows that our method can generalize well to other objects instead of just remembering the grasp poses shown in the demonstrations. To further justify this, we compare the grasp pose synthesized by our method from the same initial

Table 1. Ablation studies of our method. S is the success rate of the final grasp and Q_1 is the mean generalized Q_1 value of all successful grasps. The percentages of successful grasps whose process penetration is smaller than different thresholds $\tau = 1mm, 2mm$ and $3mm$ are also reported. More details about the metrics and the settings of different versions of our methods are provided in Section 5.3.

Method	Final grasp		Process penetration		
	S	Q_1	$\leq 1mm$	$\leq 2mm$	$\leq 4mm$
IBS-G	-	-	-	-	-
+ Demo	28.7%	0.217	30.8%	44.2%	54.7%
+ Q-Vec	65.8%	0.215	47.2%	63.3%	79.1%
+ IBS-L (Ours)	**68.3**%	0.207	58.3%	75.7%	89.7%
RGBD image	9.1%	0.173	59.6 %	68.8%	74.3%
Point cloud	17.3%	0.148	55.3%	65.9%	75.0%

configuration of the demonstration grasp in Figure 10. We can see that even starting from the same initial configuration, our method ends up with quite different and generally more natural grasp poses based on the learned policy.

5.3 Quantitative evaluation

To quantitatively evaluate our method, we first conduct ablation studies to justify several key design choices of our method, and then we provide comparisons to several baseline methods to show the superiority of our method.

Ablation studies. As explained in Section 4.3, our goal is to learn a reach-and-grasp planner with high-quality final grasps as well as low penetration during the whole process, thus we use metrics to measure the final grasp and process penetration, respectively. For the final grasp, we first compute the success rate S among the whole testing set, where whether each final grasp is successful or not is tested in the simulator. Then for all successful grasps, we compute the average generalized Q_1 [Liu et al. 2020b] as a complementary metric to get a more detailed grasp quality measure. For the process penetration, we further compute the percentage of the testing objects, of which the penetration of each frame during the whole reach-and-grasp process is smaller than a given threshold τ, where $\tau = 1mm, 2mm$ and $3mm$.

Using IBS as the dynamic state representation is the key contribution of our method. Based on this, we further propose to use demonstrations, the vector-based representation of Q value, and a combination of the local and global encoder to boost the training and improve the performance. To show the importance of all those design choices, we use the simplest version of our method as the baseline, denoted as "IBS-G", which only uses a global encoder for IBS and a standard scalar Q value and is trained without demonstrations, and then gradually add those key components one by one in the ablation studies to show how performance is boosted accordingly. To further justify the superiority of IBS as the state representation, we also compare different versions of our method with either RGBD images or point clouds as state representation. The evaluation results of the ablation studies are shown in Table 1.

Importance of demonstration. Training with demonstrations is crucial to make our network be able to learn meaningful policy. We can see in the Table 1 that without demonstration, the baseline "IBS-G" cannot learn any meaningful policy to perform valid grasping and thus the evaluation metrics cannot be reported, due to the high complexity of our problem and the corresponding large searching space. When trained with demonstration, denoted as "+ Demo", The network is able to learn a reasonable policy now, although the performance is not satisfactory enough with only 28.7% success rate. Moreover, for the most of those successful grasps, the penetration is higher than other settings.

Importance of vector-based representation of Q value. Making the Q-Network output a vector instead of a single scale value provides better control of each individual gripper component. To justify the benefit of such a design, we further enable this feature in our method and report the results in the third row of Table 1 , denoted as "+ Q-Vec". We see that compared to the results using a scalar Q value, the performance gets highly boosted, including both success rate and penetration avoidance. The main reason is that when combining the grasp quality measure and the contacting measure together into one single value, it's hard for the network to learn which is the main reason to cause the change of the value, while using a vector-based representation can provide a more clear guidance for the network to learn.

Importance of multi-level encoder. To further show the importance of the multi-level encoder used for feature extraction of IBS, i.e., using both global and local encoders and then concatenating the features together, we further add the local encoder to get the full version of our method, and the result is shown in the fourth row of Table 1 , denoted as "+ IBS-L". We can see that the performance is better than other settings, which shows that the component-based IBS partition and the corresponding local encoder can help get more information and give better control of those gripper components.

Importance of IBS as state representation. The introduction of IBS to represent the intermediate state during the whole planning process and encode the interaction between the gripper and the scene is one of our key contributions. To show the importance of our IBS encoding, we compare our method to two alternative ways of interaction representation, i.e., using either RGBD images or point clouds of the scene and the gripper, while all the other design choices are the same to our method, including using demonstrations and vector-based representation of Q value.

For RGBD image presentation, we use visual information captured by a simulated hand-mounted camera in its upright-oriented pose that translates together with the hand but does not rotate. The visual input consists of an RGBD image and the corresponding segmentation information, where each pixel can belong to the object, the gripper or the background. We then replace the encoder with similar architecture to that of the network used in [Jain et al. 2019]. More details about this baseline can be found in the supplementary material.

For point cloud representation, we use the scene point cloud and gripper point cloud directly as input. For the features to be encoded,

Table 2. Quantitative comparison to two-step baseline methods with "grasp synthesis + motion planning". where final grasp poses are synthesized using different methods, including Liu[2020b], GraspIt! [Miller and Allen 2004], and our method, and motions from the same gripper configuration to those final grasp poses are planned using the same method. We conduct the comparison on the YCB dataset, and report the success rates of the final grasp, motion planning, and overall execution in the simulator, where "Avg" refers to the average success rate among all testing samples and " Top-1" refers to the average success rate of all testing object. One test sample means one object with one initial configuration.

Method	Final grasp		Motion Plan		Overall	
	Avg	Top-1	Avg	Top-1	Avg	Top-1
Liu[2020b]	13.6%	24.0%	34.4%	44.0%	8.0%	12.0%
GraspIt!	52.1%	62.0%	36.3%	54.0%	22.6%	42.0%
Ours	**68.3%**	**100.0%**	**58.3%**	**100.0%**	**43.2%**	**90.0%**

we remove the features that contain interaction information from the list of features we defined for IBS points in Section 4.2 and keep all the remaining ones, which include the coordinate $c_s \in R^3$ and foreground/background indicator $b_s \in 0, 1$ for each scene point and coordinate $c_g \in R^3$, gripper component indicate $c_g \in \{0, 1\}^6$ and gripper side indicator $a_g \in [-1, 1]$ for each gripper point. The encoder is similar to that we used for IBS-G. More details about this baseline can also be found in the supplementary material.

The corresponding performances are shown in the last two rows of Table 1, denoted as "RGBD image" and "Point cloud", respectively. Note that our method with either setting performs poorly . We think the main reason is that they cannot provide enough spatial information between the object and the gripper when the demonstrations provided are imperfect and with large variations in initial configurations and final grasp poses, while the IBS we used provides a more informative representation.

Comparison to baselines. Strategies for the reach-and-grasp task can be roughly divided into two categories, one synthesizes the grasp first and then plans the transfer from the initial configuration to the final grasp pose and the other optimizes the whole process directly. To give more detailed comparisons to previous methods, we organize the baseline methods according to those two different types of strategies and analyze the results separately.

For the first type of baselines, we compare the final grasps obtained using our method to those synthesized using the method in [Liu et al. 2020b] and GraspIt! [Miller and Allen 2004]. For the final grasp pose generated via GraspIt!, we select the one closest to the initial gripper configuration from a pre-sampled set of candidate grasp poses based on the distance metric proposed in [Di Gregorio 2008] To further generate the whole reach-and-grasp process, we use the same motion planning method [Kavraki et al. 1996] from The Open Motion Planning Library (OMPL) [Sucan et al. 2012] to plan a collision-free moving trajectory from the initial configuration to the global pose of the final pose. We can execute the whole process in our dynamics simulator to see if the object can be successfully grasped in the end to get an overall success. In more detail, once the gripper in the rest pose reaches its final position and orientation following

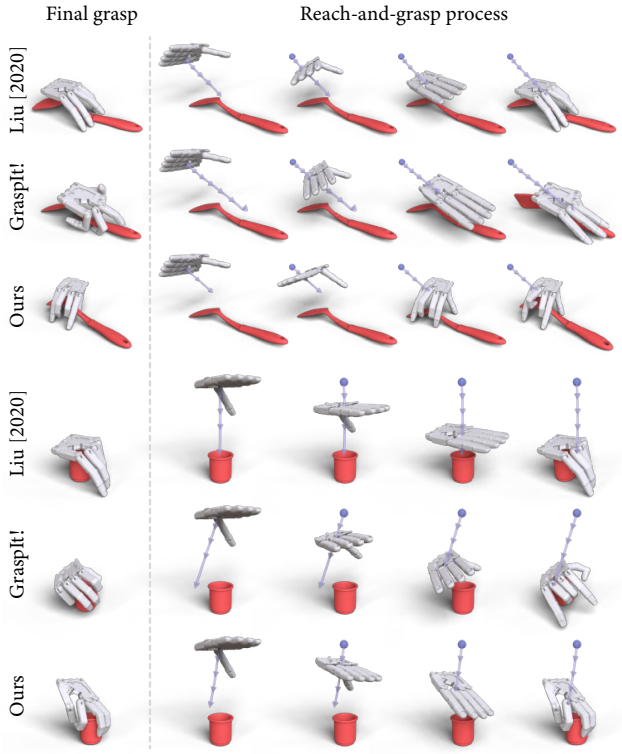


Fig. 11. Visual comparisons to two-step baseline method with "grasp synthesis + motion planning" . For each example, we show the final grasp synthesized by each method on the left and several key frames of the reach-and-grasp process with the offline planned trajectory in purple on the right.

the planned trajectory, we start using the interface provided by the simulator to move fingers towards the target joint states unless it contacts with the scene. When all fingers stop moving, we use the simulator to execute the grasping test. Accordingly, we can compute the success rate for different stages. The first one is "final grasp", where we test whether the gripper with the given final configuration can successfully grasp and lift the object in the dynamics simulator. The second one is "motion plan", where we check whether the given planner can find a feasible trajectory to transit the gripper from the initial configuration to the final configuration. The third one is "overall", where we consider the whole process to be successful if and only if both final grasp and motion plan succeed.

Table 2 reports the success rate of the final grasp, motion plan, and the overall process executed in the dynamic simulator . Note that the success rates are computed for different settings. "Avg" refers to the average success rate among all testing samples, while " Top-1" refers to the average success rate of all testing objects, where each object is tested with a fixed set of initial configurations and is considered to be successfully grasped if any of those initial configurations leads to a successful grasp. We can see that using the final grasp obtained via our method gets consistently better results.

Table 3. Quantitative comparison to the primitive-based method (PBM) on four datasets. We report the overall success rates of execution in the simulator, where “Avg" refers to the average success rate among all testing samples and “ Top-1" refers to the average success rate of all testing object. One test sample means one object with one initial configuration.

Method	YCB*		BIGBIRD*		ContactPose		3DNet*	
	Avg	Top-1	Avg	Top-1	Avg	Top-1	Avg	Top-1
PBM	57.0%	92.0%	47.2%	89.0%	44.5%	84.0%	48.7%	90.0%
Ours	**58.6%**	**100.0%**	**47.7%**	**95.6%**	**48.0%**	**88.0%**	**52.3%**	**100.0%**

Figure 11 shows some visual comparisons of those methods. For each example, we show the final grasp synthesized by each method on the left and several key frames of the reach-and-grasp process with the offline planned trajectory in purple on the right. From the results, the method of Liu et al. [2020b] cannot handle the tabletop well: It tends to generate a grasp pose with finger touching the table and does not form a valid grasp pose, especially for flat-shaped objects shown in the first row. Although GraspIt! can generate much more successful grasps, it usually fails to plan a collision-free moving path for the gripper from the given initial configuration to the final pose, and even with successfully planned path, the gripper configuration is more likely to be changed during the reaching process to avoid collision with the tabletop, which leads to a lower overall success rate. Note how the final grasp pose after the reaching process is different from the planned final poses shown in the fifth row. But for the final grasp pose generated by our method, we have taken the tabletop into the consideration during the whole reaching process, thus on the one hand, our method gets a much higher success rate of the final grasp, one the other hand, even using the same motion planning method instead of the own motion planned by our method, we can still get the best overall success rate.

For the second type of baselines, we compare the whole reach-and-grasp process obtained via our method to a heuristic primitive-based grasping method. Inspired by the work of [Della Santina et al. 2019], we adopt three grasping primitives, including “Pinch", “Top", and “Lateral", and for each testing initial configuration, we select the closest primitive based on the geodesic distance on the sphere to execute the corresponding grasp. Note that “Top" and “Pinch" primitives have the same start gripper configurations, and we choose the one that achieves higher performance to get the final result. More details about this primitive-based method (PBM) can be found in the supplementary material.

Table 3 reports the comparisons of “Avg" and “ Top-1" success rate of overall execution in the simulator on four different datasets. We can see that our method outperforms the primitive-based method on all the datasets. The main reason is the reaching path of each primitive grasp is fixed and cannot adapt well to the various geometry of different objects, while our method can keep targeting the object with the guidance of the encoded IBS. Figure 12 shows some visual comparisons between our method and PBM. Note how the final grasp poses of the primitive-based method deviate from the object due to the shape complexity. For example, neither the final grasps of PBM shown in the first and third examples touches the object.

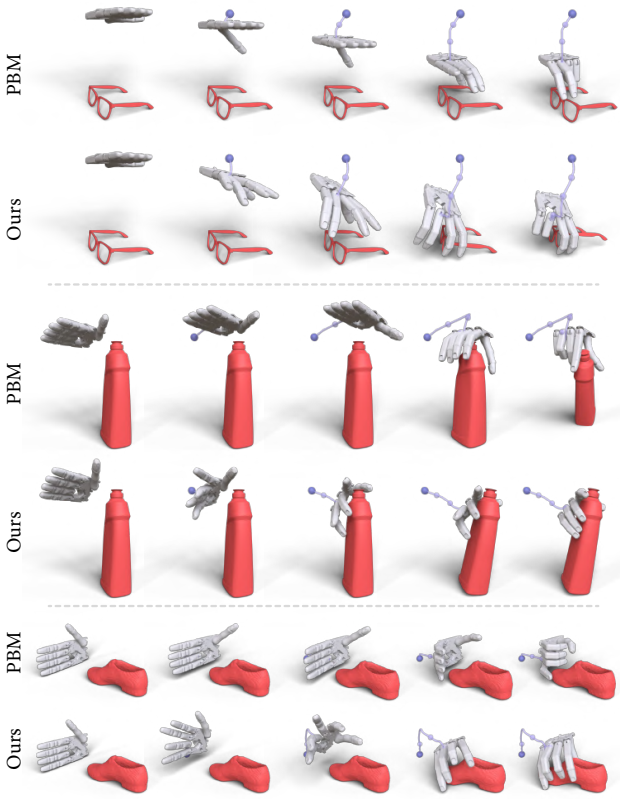


Fig. 12. Visual comparisons to the primitive-based method (PBM). For each example, we show the initial configuration of the gripper on the left, final grasping pose on the right, and three sampled frames during the reaching process in the middle. The purple curve indicates the moving trajectory before the current frame. From top to bottom, the primitives used in these examples for PBM are “Pinch", “Top", and “Lateral", respectively.

One interesting aspect of our method we would like to highlight is that when executing the reach-and-grasp process in the simulator, the pose of the target object may change due to collision with the gripper, and our method is still able to dynamically adapt to its new pose, follow the moving object and grasp it successfully. See the grasp result of the bottle shown in the fourth row of Figure 12. Other than the adaption to dynamic changes of the target object, our method is also robust to partial observations. Examples can be found in the supplementary materials.

6 DISCUSSION AND FUTURE WORK

We have presented an RL-based method to jointly learn grasp and motion planning for high-DOF grippers. We advocate the use of Interaction Bisector Surface to characterize the fine-grained spatial relationship between the gripper and the target object. We found that IBS is surprisingly effective as a state/observation representation of deep RL since it well informs the fine-grained control of each finger with spatial relation against the target object. Together

(a) (b)

Fig. 13. Failure cases. (a) Unnatural grasping poses. (b) Unsuccessful grasps for flat-shaped target objects.

with a few critical designs of the learning model and strategy, our method learns high-quality grasping with smooth reaching motion.

Our method has the following limitations:

- Our RL model adopts success signal in execution as well as Q1 metric for measuring grasp quality and does not explicitly define the naturalness of a grasp. Therefore, there is the case that a generated grasp is successful but looks unnatural. For example, some successful grasps could have an unbent finger that looks implausible; see Figure 13(a).
- The other reward we use is devised to avoid collision during the reach-and-grasp process. This makes our method unable to learn picking up a flat shape lying on the table. See Figure 13(b) for an example.

As future work, we would like to conduct further investigation on the following four aspects:

- To further improve the performance of our method, we can try more complicated feature encoding other than the current PointNet and MPLs, and further, take more dynamic information of each frame such as velocity into account.
- To generate more natural grasps, it is necessary to investigate grasp quality metrics that can better reflect grasp naturalness. This could be learned from human grasping datasets such as ContactPose [Brahmbhatt et al. 2020] .
- To make our method be able to grasp flat-shaped objects, we can relax the collision constraint and even utilize the collision with the environment to help lift the object and achieve a successful grasp as in [Eppner et al. 2015] .
- To be able to conduct real robot implementation, we need to study how to perform sim-to-real policy transfer, overcoming the domain gap between simulated observation and real visual perception, as well as the gap between simplified static environment and real dynamic scenes.

7 ACKNOWLEDGEMENTS

This work was supported in parts by NSFC (61872250, 62132021, U2001206, U21B2023, 62161146005), GD Talent Plan (2019JC05X328), GD Natural Science Foundation (2021B1515020085), DEGP Key Project (2018KZDXM058, 2020SFKC059), National Key R&D Program of China (2018AAA0102200), Shenzhen Science and Technology Program (RCYX20210609103121030, RCJC20200714114435012, JCYJ20210324120213036), and Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ).

REFERENCES

Jacopo Aleotti and Stefano Caselli. 2012. A 3D shape segmentation approach for robot grasping by parts. *Robotics and Autonomous Systems* 60, 3 (2012), 358–366.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. 2020. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research* 39, 1 (2020), 3–20.

Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. 2013. Data-driven grasp synthesis—a survey. *IEEE Transactions on Robotics* 30, 2 (2013), 289–309.

Samarth Brahmbhatt, Chengcheng Tang, Christopher D Twigg, Charles C Kemp, and James Hays. 2020. ContactPose: A dataset of grasps with object contact and hand pose. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*. Springer, 361–378.

Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. 2017. Yale-CMU-Berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research* 36, 3 (2017), 261–268.

I-Ming Chen and Joel W Burdick. 1993. Finding antipodal point grasps on irregularly shaped objects. *IEEE transactions on Robotics and Automation* 9, 4 (1993), 507–512.

Alvaro Collet and Siddhartha S Srinivasa. 2010. Efficient multi-view object recognition and full pose estimation. In *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2050–2055.

Cosimo Della Santina, Visar Arapi, Giuseppe Averta, Francesca Damiani, Gaia Fiore, Alessandro Settimi, Manuel G Catalano, Davide Bacciu, Antonio Bicchi, and Matteo Bianchi. 2019. Learning from humans how to grasp: a data-driven architecture for autonomous grasping with anthropomorphic soft hands. *IEEE Robotics and Automation Letters* 4, 2 (2019), 1533–1540.

Amaury Depierre, Emmanuel Dellandréa, and Liming Chen. 2018. Jacquard: A large scale dataset for robotic grasp detection. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 3511–3516.

Raffaele Di Gregorio. 2008. A novel point of view to define the distance between two rigid-body poses. In *Advances in robot kinematics: Analysis and design*. Springer, 361–369.

Clemens Eppner, Raphael Deimel, José Alvarez-Ruiz, Marianne Maertens, and Oliver Brock. 2015. Exploitation of environmental constraints in human and robotic grasping. *The International Journal of Robotics Research* 34, 7 (2015), 1021–1038.

Kuan Fang, Yunfei Bai, Stefan Hinterstoisser, Silvio Savarese, and Mrinal Kalakrishnan. 2018. Multi-Task Domain Adaptation for Deep Learning of Instance Grasping from Simulation. *IEEE International Conference on Robotics and Automation (ICRA)* (2018).

Carlo Ferrari and John F Canny. 1992. Planning optimal grasps.. In *ICRA*, Vol. 3. 2290–2295.

F Ficuciello, A Migliozzi, G Laudante, P Falco, and B Siciliano. 2019. Vision-based grasp learning of an anthropomorphic hand-arm system in a synergy-based control framework. *Science robotics* 4, 26 (2019).

Fanny Ficuciello, Damiano Zaccara, and Bruno Siciliano. 2016. Synergy-based policy improvement with path integrals for anthropomorphic hands. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1940–1945.

Joan Fontanals, Bao-Anh Dang-Vu, Oliver Porges, Jan Rosell, and Máximo A Roa. 2014. Integrated grasp and motion planning using independent contact regions. In *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 887–893.

Marcus Gualtieri, Andreas Ten Pas, Kate Saenko, and Robert Platt. 2016. High precision grasp pose detection in dense clutter. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 598–605.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. 2018. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905* (2018).

Kaiyu Hang, Johannes A. Stork, Nancy S. Pollard, and Danica Kragic. 2017. A Framework for Optimal Grasp Contact Planning. *IEEE Robotics and Automation Letters* 2, 2 (2017), 704–711. https://doi.org/10.1109/LRA.2017.2651381

Ruizhen Hu, Chenyang Zhu, Oliver van Kaick, Ligang Liu, Ariel Shamir, and Hao Zhang. 2015. Interaction context (ICON) towards a geometric functionality descriptor. *ACM Transactions on Graphics* 34, 4 (2015), 1–12.

Divye Jain, Andrew Li, Shivam Singhal, Aravind Rajeswaran, Vikash Kumar, and Emanuel Todorov. 2019. Learning Deep Visuomotor Policies for Dexterous Hand Manipulation. In *International Conference on Robotics and Automation (ICRA)*.

Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. 2019. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 12627–12637.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. 2018. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293* (2018).

Daniel Kappler, Jeannette Bohg, and Stefan Schaal. 2015. Leveraging big data for grasp planning. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 4304–4311.

Korrawe Karunratanakul, Jinlong Yang, Yan Zhang, Michael Black, Krikamol Muandet, and Siyu Tang. 2020. Grasping Field: Learning Implicit Representations for Human Grasps. *arXiv preprint arXiv:2008.04451* (2020).

Alexander Kasper, Zhixing Xue, and Rüdiger Dillmann. 2012. The KIT object models database: An object model database for object recognition, localization and manipulation in service robotics. *The International Journal of Robotics Research* 31, 8 (2012), 927–934.

Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation* 12, 4 (1996), 566–580.

Marios Kiatos and Sotiris Malassiotis. 2019. Grasping Unknown Objects by Exploiting Complementarity with Robot Hand Geometry. In *International Conference on Computer Vision Systems*. Springer, 88–97.

Chong-Min Kim, Chung-In Won, Youngsong Cho, Donguk Kim, Sunghoon Lee, Jonghwa Bhak, and Deok-Soo Kim. 2006. Interaction interfaces in proteins via the Voronoi diagram of atoms. *Computer-Aided Design* 38, 11 (2006), 1192–1204.

Kilian Kleeberger, Richard Bormann, Werner Kraus, and Marco F Huber. 2020. A survey on learning-based robotic grasping. *Current Robotics Reports* (2020), 1–11.

Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. 2018. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research* 37, 4-5 (2018), 421–436.

C Karen Liu. 2009. Dextrous manipulation from a grasping pose. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 28, 3 (2009), 59:1–59:6.

Min Liu, Zherong Pan, Kai Xu, Kanishka Ganguly, and Dinesh Manocha. 2019. Generating Grasp Poses for A High-DOF Gripper Using Neural Networks. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 1518–1525.

Min Liu, Zherong Pan, Kai Xu, Kanishka Ganguly, and Dinesh Manocha. 2020b. Deep Differentiable Grasp Planner for High-DOF Grippers. In *Proceedings of the Robotics: Science and Systems 2020*.

Min Liu, Zherong Pan, Kai Xu, and Dinesh Manocha. 2020a. New Formulation of Mixed-Integer Conic Programming for Globally Optimal Grasp Planning. *IEEE Robotics and Automation Letters* 5, 3 (2020), 4663–4670. https://doi.org/10.1109/LRA.2020.3003280

Qingkai Lu, Kautilya Chenna, Balakumar Sundaralingam, and Tucker Hermans. 2020a. Planning multi-fingered grasps as probabilistic inference in a learned deep network. In *Robotics Research*. Springer, 455–472.

Qingkai Lu, Mark Van der Merwe, Balakumar Sundaralingam, and Tucker Hermans. 2020b. Multifingered Grasp Planning via Inference in Deep Neural Networks: Outperforming Sampling by Learning Differentiable Models. *IEEE Robotics & Automation Magazine* (2020).

Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio, and Ken Goldberg. 2017. Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics. In *Proceedings of Robotics: Science and Systems*. https://doi.org/10.15607/RSS.2017.XIII.058

Jeffrey Mahler, Matthew Matl, Xinyu Liu, Albert Li, David Gealy, and Ken Goldberg. 2018. Dex-net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning. In *2018 IEEE International Conference on robotics and automation (ICRA)*. IEEE, 5620–5627.

Jeffrey Mahler, Florian T Pokorny, Brian Hou, Melrose Roderick, Michael Laskey, Mathieu Aubry, Kai Kohlhoff, Torsten Kröger, James Kuffner, and Ken Goldberg. 2016. Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 1957–1964.

Alexis Maldonado, Ulrich Klank, and Michael Beetz. 2010. Robotic grasping of unmodeled objects using time-of-flight range data and finger torque information. In *International Conference on Intelligent Robots and Systems*. IEEE, 2586–2591.

Priyanka Mandikal and Kristen Grauman. 2021. Dexterous robotic grasping with object-centric visual affordances. In *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE.

Andrew T Miller and Peter K Allen. 2000. Graspit!: A versatile simulator for grasp analysis. In *in Proc. of the ASME Dynamic Systems and Control Division*. Citeseer.

Andrew T Miller and Peter K Allen. 2004. Graspit! a versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine* 11, 4 (2004), 110–122.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

Marco Monforte, Fanny Ficuciello, and Bruno Siciliano. 2019. Multifunctional principal component analysis for human-like grasping. In *Human Friendly Robotics*. Springer, 47–58.

Douglas Morrison, Peter Corke, and Jürgen Leitner. 2018. Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach. *arXiv preprint arXiv:1804.05172* (2018).

Van-Duc Nguyen. 1988. Constructing force-closure grasps. *The International Journal of Robotics Research* 7, 3 (1988), 3–16.

Zherong Pan, Duo Zhang, Changhe Tu, and Xifeng Gao. 2022. Planning of Power Grasps Using Infinite Program Under Complementary Constraints. *IEEE Robotics and Automation Letters* 7, 1 (2022), 650–657.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 3803–3810.

Sören Pirk, Vojtech Krs, Kaimo Hu, Suren Deepak Rajasekaran, Hao Kang, Yusuke Yoshiyasu, Bedrich Benes, and Leonidas J Guibas. 2017. Understanding and exploiting object interaction landscapes. *ACM Transactions on Graphics* 36, 3 (2017), 1–14.

Nancy S Pollard and Victor Brian Zordan. 2005. Physically based grasping control from example. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 311–318.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 652–660.

Deirdre Quillen, Eric Jang, Ofir Nachum, Chelsea Finn, Julian Ibarz, and Sergey Levine. 2018. Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 6284–6291.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. 2017. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087* (2017).

Máximo A Roa and Raúl Suárez. 2009. Computation of independent contact regions for grasping 3-d objects. *IEEE Transactions on Robotics* 25, 4 (2009), 839–850.

Anis Sahbani, Sahar El-Khoury, and Philippe Bidaud. 2012. An overview of 3D object grasp synthesis algorithms. *Robotics and Autonomous Systems* 60, 3 (2012), 326–336.

Ashutosh Saxena, Justin Driemeyer, Justin Kearns, and Andrew Y Ng. 2007. Robotic grasping of novel objects. In *Advances in neural information processing systems*. 1209–1216.

Ashutosh Saxena, Lawson Wong, Morgan Quigley, and Andrew Y Ng. 2010. A vision-based system for grasping novel objects in cluttered environments. In *Robotics research*. Springer, 337–348.

Arjun Singh, James Sha, Karthik S Narayan, Tudor Achim, and Pieter Abbeel. 2014. Bigbird: A large-scale 3d database of object instances. In *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 509–516.

Shuran Song, Andy Zeng, Johnny Lee, and Thomas Funkhouser. 2020. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *IEEE Robotics and Automation Letters* 5, 3 (2020), 4978–4985.

Sebastian Starke, He Zhang, Taku Komura, and Jun Saito. 2019. Neural state machine for character-scene interactions. *ACM Trans. Graph.* 38, 6 (2019), 209–1.

Ioan A Sucan, Mark Moll, and Lydia E Kavraki. 2012. The open motion planning library. *IEEE Robotics & Automation Magazine* 19, 4 (2012), 72–82.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

Mark Van der Merwe, Qingkai Lu, Balakumar Sundaralingam, Martin Matak, and Tucker Hermans. 2019. Learning Continuous 3D Reconstructions for Geometrically Aware Grasping. *arXiv preprint arXiv:1910.00983* (2019).

Jacob Varley, Chad DeChant, Adam Richardson, Joaquín Ruales, and Peter Allen. 2017. Shape completion enabled robotic grasping. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2442–2447.

Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. 2017. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817* (2017).

Ulrich Viereck, Andreas Pas, Kate Saenko, and Robert Platt. 2017. Learning a visuomotor controller for real world robotic grasping using simulated depth images. In *Conference on Robot Learning*. PMLR, 291–300.

Lirui Wang, Yu Xiang, and Dieter Fox. 2019. Manipulation trajectory optimization with online grasp synthesis and selection. *arXiv preprint arXiv:1911.10280* (2019).

Walter Wohlkinger, Aitor Aldoma, Radu B Rusu, and Markus Vincze. 2012. 3dnet: Large-scale object class recognition from cad models. In *2012 IEEE international conference on robotics and automation*. IEEE, 5384–5391.

Zhenjia Xu, Beichun Qi, Shubham Agrawal, and Shuran Song. 2021. Adagrasp: Learning an adaptive gripper-aware grasping policy. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 4620–4626.

Xi Zhao, Myung Geol Choi, and Taku Komura. 2017. Character-object interaction retrieval using the interaction bisector surface. *Computer Graphics Forum (Proc. Eurographics)* 36, 2 (2017), 119–129.

Xi Zhao, He Wang, and Taku Komura. 2014. Indexing 3d scenes using the interaction bisector surface. *ACM Trans. on Graphics* 33, 3 (2014), 1–14.