

# AnyGrasp: Robust and Efficient Grasp Perception in Spatial and Temporal Domains

Hao-Shu Fang<sup>✉</sup>, Chenxi Wang<sup>✉</sup>, Hongjie Fang<sup>✉</sup>, Minghao Gou<sup>✉</sup>,  
Jirong Liu<sup>✉</sup>, Hengxu Yan<sup>✉</sup>, Wenhai Liu<sup>✉</sup>, Yichen Xie<sup>✉</sup>, Cewu Lu<sup>✉</sup>, *Member, IEEE*

**Abstract**—As the basis for prehensile manipulation, it is vital to enable robots to grasp as robustly as humans. Our innate grasping system is prompt, accurate, flexible, and continuous across spatial and temporal domains. Few existing methods cover all these properties for robot grasping. In this paper, we propose AnyGrasp for grasp perception to enable robots these abilities using a parallel gripper. Specifically, we develop a dense supervision strategy with real perception and analytic labels in the spatial-temporal domain. Additional awareness of objects’ center-of-mass is incorporated into the learning process to help improve grasping stability. Utilization of grasp correspondence across observations enables dynamic grasp tracking. Our model can efficiently generate accurate, 7-DoF, dense, and temporally-smooth grasp poses and works robustly against large depth-sensing noise. Using AnyGrasp, we achieve a 93.3% success rate when clearing bins with over 300 unseen objects, which is on par with human subjects under controlled conditions. Over 900 mean-picks-per-hour is reported on a single-arm system. For dynamic grasping, we demonstrate catching swimming robot fish in the water.

**Index Terms**—General grasping, dynamic grasping, AnyGrasp

## I. INTRODUCTION

VISUAL guided object grasping is important in the robotic community. Other than solely picking objects in the industrial environment, we would foresee robots working closely with humans even in household environments. Thus, a grasping system that can serve most daily manipulation in unstructured environments is attractive.

To achieve that, we first introspect our own grasping ability. For humans, we perceive a partial observation without a full model of the scene. Our visual system processes such input within 100ms and we would then know how to pre-shape our hand for comforting contact during grasping [1]. Such a grasp system is accurate and robust for any object. Moreover, a time-consistent property is guaranteed so that we can recognize and grasp moving objects in dynamic scenes.

To date, researchers have proposed different methods for visual-guided grasping. Some methods [2], [3] assumed full knowledge of the object and contact model, which does not always hold in the real world. Some simplified the grasping perception as a planar detection problem [4], [5] but would

H.-S. Fang, C. Wang, H. Fang, M. Gou, J. Liu, H. Yan, W. Liu, Y. Xie and C. Lu are with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China (email: fhaoshu@gmail.com, {wcx1997, galaxies, gmh2015, jirong, hengxuyan, sjtu-wenhai, xieyichen, lucewu}@sjtu.edu.cn). C.Lu is the corresponding author.

impose restrictions on afterward manipulation. Some proposed sampling-evaluation methodologies sample candidates from the scene and then evaluate their quality [5]–[8]. However, this approach is time-consuming and cannot generate dense predictions. Furthermore, these methods mainly focus on static scene grasp detection, and the dynamic grasp detection problem remains largely unexplored.

In this paper, we present a spatially and temporally unified methodology, AnyGrasp, to bridge the grasp perception ability gap between robots and humans. We focus on grasping with a parallel gripper in this paper. Specifically, a geometry processing module estimates dense 7-DoF grasp configurations for a monocular perceived observation in one feed-forward pass. A temporal association module identifies the grasp correspondence among these tremendous grasp poses across every two observations. To reduce the burden of explicit collision detection during grasping, our model is obstacle-aware and eliminates the grasp candidates that have no space for hand placing. This helps accelerate the selection of grasp poses in afterward manipulation. To improve stability, the awareness of the center of gravity (COG) for objects is also equipped. These two properties are also witnessed in human’s visually guided grasping behavior [1]. Our model can generate accurate, 7-DoF and continuous grasp poses across space and time in 100ms.

A problem during learning is that we lack a dataset for dynamic object grasping in the real world. Previous grasp learning resorts to collecting data in simulation. However, we show in this paper that when using a low-cost commercial depth sensor, an algorithm with simple sim-to-real transferring techniques still performs worse than directly training with real-world data. Thus, we persist in training with real data. To avoid extensive human labor, we exploit the grasp pose correspondence across different observations of a static scene and propose a dense supervision strategy with real perception and analytic labels in the spatial-temporal domain that greatly improves data efficiency.

We have conducted relevant research [9]–[11] before. However, we only considered the grasping problem in static scenes and did not fully evaluate the grasp perception system in different scenarios. Thus, in this paper, we develop the grasping ability in dynamic scenes that is complementary to our previous work. Besides, we did not clearly show the advantages of training with real-world data [9] over simulated data. In this paper, we explicitly demonstrate the superiority

of our grasping system, trained with a relatively small amount of real-world objects (144 objects), on a variety of challenging tasks that are not presented in previous work trained with thousands of simulated objects. Extra analyses of different principles in dataset design are also conducted. We hope our discovery will encourage the community in related areas to prioritize real-world training data.

Specifically, we first test our algorithm in a large-scale bin-picking experiment with over 300 unseen objects with diverse shapes, materials, and sizes. Our results demonstrate an average success rate of over 93% and a completion rate of over 99.8%, which is comparable to the performance of humans using the same end-effector configuration and open-loop grasping strategy. We further carry out experiments of our method on different depth sensing noise to show its robustness. For dynamic scenes, the robot successfully demonstrates catching moving robot fish in the fish tank, which is difficult due to the small friction in the water and the tiny size of the fish. Finally, we discuss the influence of several factors in dataset construction, such as real-world data versus simulated data, the number of grasp annotations on an object, the scene diversity, etc. These analyses may be helpful for future data collection in this area.

The novelty and contribution of this paper include:

- We propose the first unified system for fast, accurate, 7-DoF and temporally-continuous grasp pose detection, using a parallel gripper.
- We incorporate the awareness of object COG for grasp perception and propose a new generation-association methodology for dynamic 7-DoF grasp configuration prediction.
- We demonstrate the robustness of our method, trained on real-world data consisting of *only 144 objects*, through extensive experiments in many challenging scenarios.
- We release a grasping library that has on-par performance with human subjects tested on over 300 unseen objects, with over 900 mean picks per hour (MPPH).
- We provide a detailed analysis of different training factors, such as the selection of real or simulated data, the influence of annotation density, the importance of scene variance, etc.

To support reproducible research, the library and example code of AnyGrasp system is provided in supplementary materials.

## II. RELATED WORK

In this section, we introduce some background material that most relates to the visual grasping topic. The scope is restricted to the two-finger parallel gripper, which we mainly discuss in this paper. This section is divided into two parts, where Sec. II-A focuses on grasp pose detection methods and Sec. II-B focuses on continuous action learning methods.

### A. Grasp Pose Detection

The grasp pose detection problem can be defined as predicting several poses for a given scene, usually in the Cartesian space, such that the robot can successfully lift the objects when

moving its end-effector to those poses. Early methods would assume full 2D or 3D knowledge of the objects [12], [13], or approximate the objects into a set of primitive shapes [14], [15]. These methods would meet limitations in real-world environments where the 3D models of objects are hard to obtain. Learning-based methods help to alleviate this dilemma through large-scale data and automatic feature extraction. Representations for grasp pose detection evolved with the development of learning-based methods, including a point-based representation [16]–[18], a point pair representation [19], a rectangle representation [5], [20]–[25], a grasp quality maps representation [26], etc. These methods mainly generate 4 DoF grasp poses on the camera plane. The limitation in the degree of freedom may neglect some vital grasp poses, *e.g.* grasp poses on the edge of a plate, and result in failure.

To generate the full 6 DoF grasp poses, Ten Pas *et al.* [6] proposed a sampling-evaluation-based method that firstly samples grasp candidates on point cloud and then evaluates their grasp qualities with a neural network. Different sampling or evaluation models are later proposed [8], [27], [28]. A major drawback of the sampling-evaluation methods is that they need to trade off the computation time and the number of generated grasp poses. Thus, they usually took several seconds to run and only generated tens of grasp poses for a scene. Recently, Fang *et al.* [9] proposed an end-to-end network that directly generates abundant grasp poses for an input scene point cloud. A large-scale dataset with real data and analytic labels was also built. Similar end-to-end networks were parallelly developed in [29]–[31]. The major difference between these methods is how they represent the  $SE(3)$  grasp pose. Later, Wang *et al.* [11] proposed a graspsness module that eliminates unfeasible grasp poses in the early stage of the network, which greatly improves the grasp success rate. Other than point cloud, some researchers also explored other representations of a scene, including RGB/D image [10], [32] and neural radiance field [33]. A recent survey [34] provided a full summary of the 6 DoF grasping methods. In this paper, we follow the end-to-end methodology and mainly build our network upon [11]. To improve grasp stability, we further incorporate awareness of the object’s center-of-mass into the learning process, which has been less considered before.

The problem formulation of grasp pose detection constrains them to focus on static scenes. In order to extend to the dynamic scene, classic methods usually require prior information about the objects [35]–[37], or operate on a fixed set of grasp trajectories [38]. For dynamic grasping without prior, existing methods either choose the nearest grasp pose in the next image as the same grasp target [26] or generate possible future candidates and evaluate their grasp quality [39]. Both methods only guarantee the tracked grasp pose has a small distance with respect to the last frame in the image coordinate system, but cannot guarantee a small distance in the object coordinate system. In this paper, we propose a new generation-association methodology for predicting dynamic 7-DoF grasp configurations.

### B. Continuous Action Learning for Grasping

Different from the above methods that detect the grasp poses in the scene and move toward the target grasp pose using a motion planner, this line of research directly maps the observation to a continuous action space. This line of research has a different problem formulation than our paper, so we can only provide a summary of some representative work due to space limitations. Levine *et al.* [40] collected 800K grasp attempts on real robots and learn a CNN grasp predictor that associates image and motor command. Viereck *et al.* [41] proposed to learn with simulated point clouds, such that no real-robot trials are needed. Bousmalis *et al.* [42] investigated the sim-to-real transfer problem in grasp action learning. Song *et al.* [43] collected grasping demonstrations with a low-cost hand-held gripper. Wang *et al.* [44] extended the continuous action space to 6D. A problem of these methods is that they may be prone to the domain shift problem [45], [46] and perform less robust than the detection-based methods.

### C. Training Data for 6-DoF Grasping

For the 6-DoF grasping problem, various simulated object sets have been collected, including DexNet [25], which has over 1000 objects, EGAD [47], which generates over 2000 objects using an evolutionary algorithm, and Acronym [48], which collects over 8000 objects. With each passing year, the number of objects in these datasets continues to increase. However, it remains unclear how many training objects are actually necessary, and the increasing number of objects may make the training process increasingly burdensome. In this paper, we demonstrate that a real-world dataset consisting of only 144 objects can provide a model with grasp performance comparable to that of human subjects. We hope that our findings will encourage the community to identify critical objects for the grasping problem.

## III. ANYGRASP DESIGN PRINCIPLES

In this section, we first describe the problem definition of our spatial-temporal grasp detection. Then we introduce the principles of our system design, from both the algorithm and data perspectives.

### A. Problem Definition

For parallel jaw-based grasping, we represent a grasp pose  $\mathcal{G}$  as

$$\mathcal{G} = [\mathbf{R} \ \mathbf{t} \ w], \quad (1)$$

where  $\mathbf{R} \in \mathbb{R}^{3 \times 3}$  denotes the gripper orientation,  $\mathbf{t} \in \mathbb{R}^{3 \times 1}$  denotes the center of grasp, and  $w \in \mathbb{R}$  denotes the minimum gripper width that is suitable for grasping the target object. This representation covers all the degrees of freedom for a parallel gripper and is also referred to as the 7-DoF grasp configuration. In this paper, We use the notation  $\mathcal{E}$  to represent the environment including the robot and objects,  $\mathcal{P}$  to represent the partial-view point cloud from a depth camera, and  $s(\mathcal{E}, \mathcal{P}, \mathcal{G})$  to denote a binary variable indicating the success or failure of grasp  $\mathcal{G}$  given environment  $\mathcal{E}$  and perception  $\mathcal{P}$ . A grasp is successful if the object is lifted successfully. Our

goal is to find a set of grasp poses  $\mathbf{G} = \{\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_n\}$  that maximizes the grasp success rate given a fixed  $n$ :

$$\{\mathcal{G}_1^*, \mathcal{G}_2^*, \dots, \mathcal{G}_n^*\} = \arg \max_{|\mathbf{G}|=n} \sum_{\mathcal{G}_i \in \mathbf{G}} \text{Prob}(s = 1 | \mathcal{E}, \mathcal{P}, \mathcal{G}_i). \quad (2)$$

This means that we hope our algorithm predicts abundant grasp poses to cover the whole scene so that we can have different candidates for grasp execution.

When introducing the temporal dimension, we denote  $\mathcal{E}^t, \mathcal{P}^t, \mathbf{G}^t$  as the environment, perception and grasp poses at time  $t$ ,  $\text{dist}(\mathcal{G}_k^t, \mathcal{G}_k^{t-1} | \mathcal{E}^t, \mathcal{E}^{t-1})$  as the distance of the paired grasp poses across two moments under the coordinate system of the grasped target. Thus, we would have:

$$\begin{aligned} \{\mathcal{G}_1^*, \mathcal{G}_2^*, \dots, \mathcal{G}_n^*\}^{(t)} &= \arg \max_{|\mathbf{G}^t|=n} \sum_{\mathcal{G}_i \in \mathbf{G}^t} \text{Prob}(s = 1 | \mathcal{E}^t, \mathcal{P}^t, \mathcal{G}_i), \\ \text{s.t. } \text{dist}(\mathcal{G}_k^t, \mathcal{G}_k^{t-1} | \mathcal{E}^t, \mathcal{E}^{t-1}) &\leq \delta, \forall \mathcal{G}_k^{t-1} \in \mathbf{G}^{t-1}, \end{aligned} \quad (3)$$

$\delta$  is the tolerance error, under which two grasp poses could be regarded as the same pose.

### B. Spatial-Continuous Learning

Previous methods [6], [7] adopt the sampling strategy that chooses candidates on point cloud and evaluates their quality. Instead, our geometry processing module directly perceives the single-view point cloud of the scene and estimates the grasp quality across the  $\mathbb{R}^6$  space. We refer to it as “spatial-continuous learning” owing to the dense property of the predicted grasp poses, from which we can query a feasible grasp pose at any target location in most cases.

Compared to the sampling-based methods that classify the quality of a grasp solely based on the cropped local point cloud within the gripper space, our method also considers the geometric structure from neighbor regions that could provide richer information conveying whether a grasp is of good quality or not. Such global geometric features can be easily learned through a 3D convolutional network. By taking the whole scene as input, two extra advantages are witnessed. Firstly, to improve the stability of grasping, human tends to put visual attention on the center of gravity (COG) of the object at the preparing stage of grasping [1]. Since the mass distribution is unknown, humans usually assume the COG is the center of the object. Such intuition can be modeled by our neural network that directly perceives the whole scene. In our network, we encode such information for each grasp pose by predicting a normalized vertical distance from the gripper plane to the COG of the grasped object. Secondly, researchers in cognition found that humans would visually attend to the obstacle during the preparation period of grasping [1], [49]. That is, the visual system would also consider the obstacle and avoid collision for hand pre-shaping. Such ability can only emerge when the network takes the whole scene as input. This aspect has also been emphasized in previous methods that adopt end-to-end networks [9], [29]–[31]. In our network, if there is an obstacle around the grasp pose and leaves no space for gripper pre-shaping, the grasp quality score is directly set to zero. Although obstacle avoidance can also be performed

using mesh and point cloud, the neural-based implicit collision detection helps eliminate a large portion of unreachable grasp poses and reduces the expensive computation time.

### C. Temporal-Continuous Learning

When trying to grasp a moving object, the robot needs to continuously update the target grasp pose before catching the object. The grasp poses between every two frames should have a small  $SE(3)$  distance in the object's coordinate system to enable a smooth, target-consistent movement of the robot gripper. We refer to the learning of such property as "temporal-continuous learning".

Previous sampling-based method [39] added some disturbance to the grasp pose from the previous frame and evaluate these candidates in the current frame's geometry. To ensure computation efficiency, the generated proposals are usually sparse. Thus they cannot cover all the possible movements of the objects.

To avoid the speed-accuracy tradeoff, we propose a new generation-association methodology to ensure dense and consistent grasp poses temporally. Given two observations at different times, our geometry processing module generates dense grasp poses across the scene. A temporal association module takes the grasp poses and their corresponding geometric features encoded by the spatial model as input and produces their many-to-many association score matrix. The association score between each two grasp poses denotes their consistency in the temporal domain, measured by their  $SE(3)$  distance in the grasping object's coordinate system. Contributed to the spatial continuous property of our geometry processing module, we could generate dense grasp poses around a selected target to ensure both temporal continuity and grasp quality.

### D. Training Data

For data collection, most methods for high DoF grasp pose detection directly learn from simulation. The main reason is that obtaining dense grasp pose annotation for data collected in the real world is difficult. However, although turning to simulation lowers the training cost, an expensive and high-precision depth camera is needed [7] to achieve good performance during the inference phase. This is to bridge the sim-to-real gap since simulation produces perfect partial view depth. In contrast, we choose to directly learn from real-world perception, which requires more effort for data collection but enables the algorithm to adapt to real-world noise, especially on low-cost cameras. In our experiments, we show how our grasp detection system can tolerate different sensing noises, and improve the success rate by a large margin over its counterpart trained in simulation.

## IV. METHODS AND MATERIALS

### A. Data Collection and Annotation

For the training data of our algorithm, we mainly adopt the training set of GraspNet-1Billion [9] and follow the same methodology to collect 168 extra scenes composed of 104 new objects. In brief, we obtain objects' 3D mesh models with a

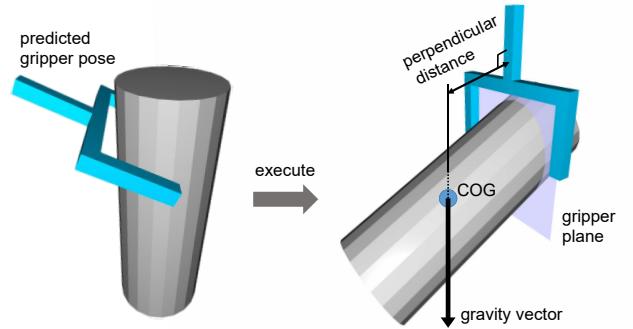


Fig. 1. An illustration of the perpendicular distance from the gripper plane to the COG of the object. Note that we assume the gripper will move to the vertical pose, parallel to the gravity force vector when transporting the object. We observe that this policy provides a larger workspace for our robot.

commercial 3D scanner and calculate dense grasp poses using analytic antipodal scores [5], [6]. For each scene, we randomly choose several objects ( $\sim 10$ ) and randomly placed them on the table. Images are taken at 256 different viewpoints for each scene and we manually annotate the object 6D poses in the scene to ensure the accuracy. Then the grasp poses on the scene can be obtained by projecting the poses on each object with the annotated 6D pose to the scene. Moreover, collision detection is performed by simplifying the gripper model into three cubes and checking whether each grasp pose has an intersection with the object models in the scene. In our pipeline, all annotations except for the 6D poses are automatically labeled by the program.. For more details, we refer readers to [9]. In total, 268 scenes that consist of 144 objects are used to train our network.

In this paper, we annotate three extra labels for the training data, which help improve the stability during grasping and enable the grasp pose tracking ability.

Firstly, the original grasp pose annotation consists of four approach depths, *i.e.*, 1, 2, 3, and 4 centimeters. In this paper, we add an extra depth of 0.5 centimeters to grasp small objects.

Secondly, as illustrated in [1], human attention will bias to the center of gravity of the object during grasping. Inspired by this, we define a stable score for the grasp pose. We assume that the gripper will move to a vertical pose that is parallel to the force of gravity when transporting the object after it is grasped. Thus, we define the stable score as the normalized perpendicular distance between its gripper plane and the COG of the object. The normalization process is conducted by obtaining all the perpendicular distances for an object and then dividing them by the maximum distance among them. In this case, the lower this score is, the more disturbance the grasp pose can tolerate since the gravity moment is more balanced. To obtain the annotation, we first compute the COG of the object. Since the center cannot be precisely computed only from visual perception, the object is assumed to be a solid rigid body with uniform density, which is similar to human intuition. Then, the COG is transformed into the gripper coordinate system according to the grasp pose. The perpendicular distance is the distance from the COG to the gripper plane. Fig. 1

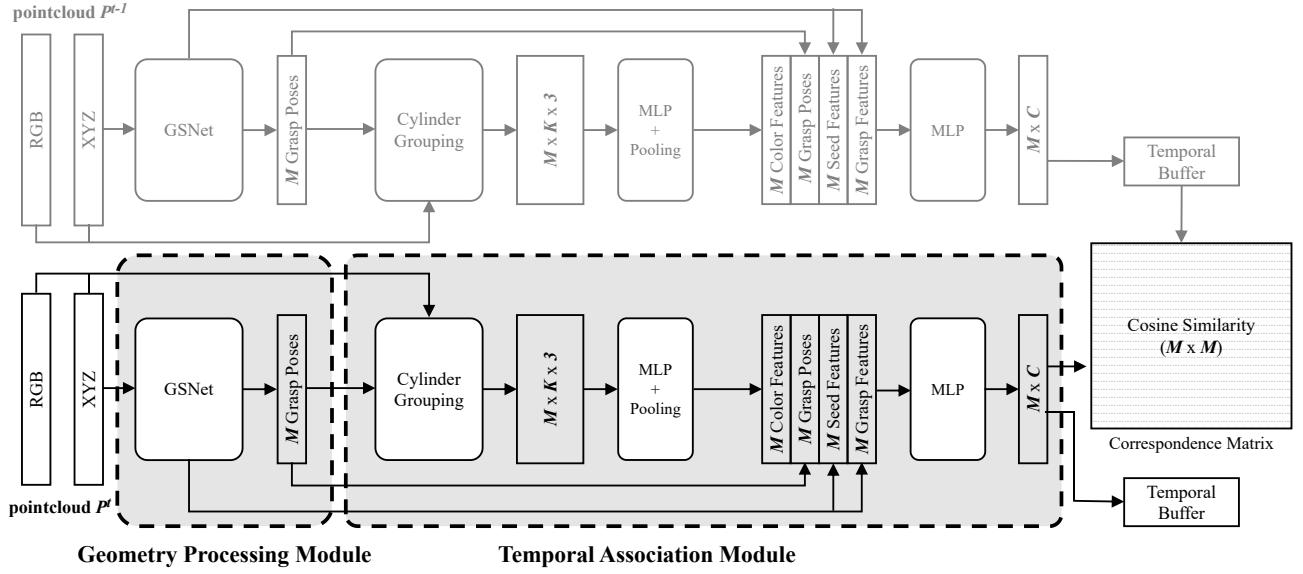


Fig. 2. Illustration of the model architecture for AnyGrasp. For a partial point cloud, the geometry processing module predicts dense grasp poses. The temporal association module generates a feature vector with size  $C$  for each of the  $M$  predicted grasp poses. The feature vector is learned with the objective that for the grasp pose pair with a smaller distance in the object coordinate frame, the higher cosine similarity they have. Thus we can construct the correspondence matrix for the dense grasp pose pairs across two frames. Details of the model structure are given in the text.

gives an example of calculating the perpendicular distance for a grasp pose. Finally, all the distances are collected and normalized to  $[0, 1]$  to get stable scores.

Thirdly, to ensure temporal consistency of the grasp poses, our algorithm associates the grasp poses that represent the same pose *w.r.t* the target object between two frames. To achieve that, a training dataset that contains the association label is needed. We generate such association labels for our collected training data. Although the original dataset does not contain dynamic scenes with moving objects, it contains images captured from 256 different viewpoints for each scene. The images captured from adjacent viewpoints present subtle differences which share similar patterns when objects are moving. Thus, for each image in the training data, we first find its neighbor images with adjacent viewpoints. For each pair of adjacent images, we annotate the association of grasp poses by defining a distance metric. For two grasp poses on the same object, we transform them to the same coordinate frame (*i.e.*, the object frame) and compute two distances:

$$\Delta\mathbf{R} = \arccos \frac{\text{trace}(\mathbf{R}_1^\top \mathbf{R}_2) - 1}{2}, \quad (4)$$

$$\Delta\mathbf{t} = \|\mathbf{t}_1 - \mathbf{t}_2\|,$$

where  $\mathcal{G}_1 = [\mathbf{R}_1 \mathbf{t}_1 w_1]$  and  $\mathcal{G}_2 = [\mathbf{R}_2 \mathbf{t}_2 w_2]$  are the transformed grasp poses,  $\text{trace}(\cdot)$  is the trace of a matrix,  $\Delta\mathbf{R}$  and  $\Delta\mathbf{t}$  are the rotation distance and the translation distance respectively. The distance between two grasp poses is defined as

$$d(\mathcal{G}_1, \mathcal{G}_2) = \frac{\Delta\mathbf{t}}{w_{\max}} + \gamma \frac{\Delta\mathbf{R}}{\pi}, \quad (5)$$

where  $w_{\max}$  is the maximum gripper width,  $\gamma$  is a distance balancing weight. In practice, we set  $w_{\max} = 0.01\text{m}$  and  $\gamma = 0.1$ . The distance between grasps on different objects is set to infinity.

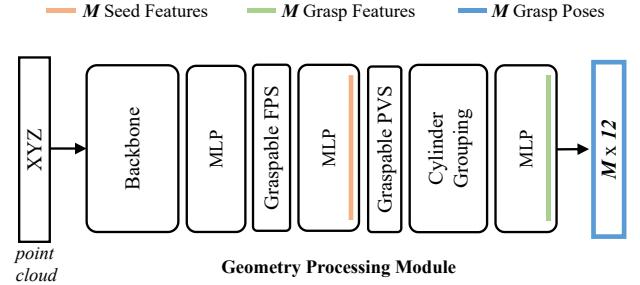


Fig. 3. Detailed illustration of the geometry processing module. The meaning of each block like the “Backbone”, “MLP”, “Graspable FPS”, etc. are specified in Sec.IV-B. The position of the seed features, grasp features, and grasp poses are denoted in different colors.

### B. Grasp Perception Model Details

Next, we illustrate the details of our algorithm, which consists of the geometry processing module and the temporal association module. Fig. 2 illustrates the model structure.

1) *Geometry processing module*: Our geometry processing module is based on GSNet [11], with a minor modification to incorporate the stable score into the network. To explain our approach, we first provide an overview of GSNet, which is also shown in Fig.3. Instead of directly predicting  $\mathbf{R}$ ,  $\mathbf{t}$  and  $w$ , GSNet decomposes these parameters into grasp point, view, in-plane rotation, approach depth, and width. Given a point cloud  $\mathcal{P}$ , a 3D convolutional backbone extracts geometric features for each point. A multi-layer perceptron (MLP) block generates an objectness mask and a heatmap indicating each point’s grasping probability based on the extracted features. Graspable farthest point sampling (Graspable FPS) is performed to sample  $M$  seed points from the scene according to the objectness mask and the grasping probability map ( $M$  is set to 1024). For each sampled point, another MLP block

generates 300 view scores to determine the most suitable view for grasping through graspable probabilistic view selection (Graspable PVS). The cylinder grouping module groups the local geometric features along that view within a cylinder space. For each group, an MLP block predicts the grasp scores for 48 grasp poses composed of 12 in-plane rotations and 4 approach depths along that view, as well as 48 grasp widths for these grasp poses. The in-plane rotation is discretized, but a heuristic search [50] can also be used. We refer readers to [11] for more details of GSNet.

The original GSNet predicts  $12 \times 4 \times 2$  values for each sampled point at the last layer, representing  $12 \times 4$  grasp scores and  $12 \times 4$  grasp widths for the grasp poses along the best view. We modify the last layer and the network now predicts  $12 \times 5 \times 2 + 12$  values. The change from 4 to 5 is because we annotate an extra approach depth for the grasp poses, as stated in Sec. IV-A. The extra 12 predictions denote the stable scores for grasp poses with different angles since the stable score is shared across different grasp depths.

During inference, we multiply the original grasp score by  $(1 - \text{stable\_score})$  as the new score for a grasp pose. Then we reparameterize the predicted grasp point, view, in-plane rotation, approach depth, and width into a 7-DoF grasp pose  $\mathcal{G} = [\mathbf{R} \ \mathbf{t} \ \mathbf{w}]$ . We then rank all predicted grasp poses according to their scores, and select the top- $n$  grasp poses as the set of grasp poses for use in Eqn. 2.

2) *Temporal association module*: On top of the geometry processing module, we develop a temporal association module to enable grasp pose tracking. The key of the temporal association module is to generate a feature vector for each grasp pose so that we can compute the correspondence score between each two grasp poses across time.

We first introduce how we construct the feature vector for each grasp pose. Given an input point cloud, the geometry processing module outputs  $\mathcal{M}$  grasp poses by choosing the pose with the highest score on each of the  $\mathcal{M}$  seed points. Concurrently, we can obtain the seed features and the grasp features of the seed points. These features are extracted before the last layer of two MLP blocks in the GSNet. Fig. 3 illustrates the feature location. These features mainly represent the geometric cues of the grasp poses. Considering that the texture or color information is also a strong cue for tracking, we also extract the color features of each seed point. Using the predicted grasp pose, we group the RGB information along the grasp direction using cylinder grouping, where the input seed features are replaced by seed colors. RGB information of  $\mathcal{K}$  points inside the cylinder space ( $\mathcal{K}$  is set to 16) are grouped for each of the  $\mathcal{M}$  seed points, resulting in a size of  $\mathcal{M} \times \mathcal{K} \times 3$ . The local texture information is forwarded through an MLP block with pooling layers to obtain the color features. Then the color features, seed features, grasp features, and the grasp poses (parameters of shape  $\mathcal{M} \times 12$ , the length 12 is composed of rotation matrix (9) and translation (3)) are concatenated and fed into an MLP block to get the feature vector with size  $\mathcal{C}$  for each predicted grasp ( $\mathcal{C}$  is set to 256).

After obtaining the feature vector of each grasp pose, we can calculate the correspondence score between each two grasp poses. Let  $\mathbf{f}_1$  be the feature vector for grasp  $\mathcal{G}_1$  in the first

point cloud, and  $\mathbf{f}_2$  for grasp  $\mathcal{G}_2$  in the second point cloud, we calculate the correspondence score using cosine similarity:

$$s_{\text{corres}}(\mathcal{G}_1, \mathcal{G}_2) = \frac{\mathbf{f}_1 \cdot \mathbf{f}_2}{\|\mathbf{f}_1\| \cdot \|\mathbf{f}_2\|}. \quad (6)$$

The scores for all  $(\mathbf{f}_1, \mathbf{f}_2)$  pairs are computed and form the correspondence matrix.

During training, two point clouds from adjacent viewpoints are respectively forwarded through the geometry processing module and the temporal association module to obtain their feature vectors for  $\mathcal{M}$  grasp poses, both having a size of  $\mathcal{M} \times \mathcal{C}$ . Then we calculate the correspondence matrix by the cosine similarity. The predicted correspondence matrix with size  $\mathcal{M} \times \mathcal{M}$  is compared with the ground-truth matrix and the loss is back-propagated to the temporal association module. The loss function is introduced in the next subsection.

During inference, only the point cloud at the current time is required to pass the network, and the feature vectors will be stored in a temporal buffer. The correspondence matrix is computed with the current features and those stored in the temporal buffer from the last frame. If we want to track  $n$  certain grasp poses selected in the previous step, we can compare its feature vector of size  $n \times \mathcal{C}$  with the  $\mathcal{M}$  feature vectors in the current frame and generate an association vector of size  $n \times \mathcal{M}$ . Then we pick the feasible grasp poses with the top- $n$  correspondence scores as the next prediction. These grasp poses are the set of grasp poses we want in Eqn. 3.

3) *Loss function*: The loss function for the geometry processing module follows [11]. It contains softmax loss for the objectness classification in the first MLP block, and smooth- $l_1$  loss for the graspable heatmaps and grasp pose parameter regression in the three MLP blocks. We refer readers to [11] for more details.

For the temporal association module, two associated grasp poses should have a high corresponding score, which indicates they have similar features. So we adopt supervised contrastive learning [51] in model training, which pulls together the features from the same class.

For two grasp poses  $\mathcal{G}_1$  and  $\mathcal{G}_2$ , they are treated as the same class if and only if  $d(\mathcal{G}_1, \mathcal{G}_2) \leq \sigma$ . The predicted grasp sets for two point clouds are denoted by  $\mathbf{G}^1 = \{\mathcal{G}_i^1 | i = 1, 2, \dots, \mathcal{M}\}$  and  $\mathbf{G}^2 = \{\mathcal{G}_j^2 | j = 1, 2, \dots, \mathcal{M}\}$  respectively. For a grasp pose  $\mathcal{G}_i^1$  in the first point cloud, we collect all the grasp poses in the second point cloud belonging to the same class of  $\mathcal{G}_i^1$ , which are denoted by  $\mathbf{P}(i) = \{\mathcal{G}_k^2 \in \mathbf{G}^2 | d(\mathcal{G}_i^1, \mathcal{G}_k^2) \leq \sigma\}$ . The loss function is defined as

$$L = \sum_{\mathcal{G}_i^1 \in \mathbf{G}^1} \frac{-1}{|\mathbf{P}(i)|} \sum_{\mathcal{G}_k^2 \in \mathbf{P}(i)} \log \frac{\exp(s_{\text{corres}}(\mathcal{G}_i^1, \mathcal{G}_k^2)/\tau)}{\sum_{\mathcal{G}_j^2 \in \mathbf{G}^2} \exp(s_{\text{corres}}(\mathcal{G}_i^1, \mathcal{G}_j^2)/\tau)}, \quad (7)$$

where  $|\mathbf{P}(i)|$  stands for the cardinality of  $\mathbf{P}(i)$  and  $\tau$  stands for the temperature parameter. In our experiment, we set  $\sigma = 0.1$  and  $\tau = 0.1$ .

### C. Training Details

Input point clouds are down-sampled with a voxel size of 0.005m. We set  $\mathcal{M} = 1024$  for each scene and  $\mathcal{C} = 256$  for each feature vector.  $\mathcal{K}$  is set to 16 in the cylinder grouping of

temporal module. The two modules are trained on the extended GraspNet-1Billion dataset using one Nvidia GTX 2080 Ti GPU with Adam optimizer [52] and an initial learning rate of 0.001. We adopt the “poly” policy with  $power = 0.9$  for learning rate decay, which is used in DeepLab [53].

We first train the geometry processing module from scratch with a batch size of 4. For data augmentation, we randomly flip the scene horizontally, and randomly rotate the points by  $Uniform[-30^\circ, 30^\circ]$  around the Z-axis (in the camera coordinate frame). We also randomly translate the points by  $Uniform[-0.2, 0.2]m$  in X/Y-axis and  $Uniform[-0.1, 0.2]m$  in Z-axis.

After the geometry processing module converged, we freeze its weights and train the temporal association module jointly with it. Each mini-batch contains four pairs of point clouds, where the two point clouds in one pair are captured by neighboring viewpoints from the same scene. Besides random flipping, random rotation, and random translation, we also randomly remove some objects in the scene for data augmentation with a probability of 0.2.

#### D. Detection Post-processing

For the detected grasp poses, we conduct two post-processing steps to improve the stability. The first step is to perform collision detection. Although our network also implicitly learns whether a grasp pose would collide with the scene, such an obstacle-aware property is not a hard constraint and may be prone to noise. Thus, we perform extra collision detection for the top-100 grasp poses among the predicted results. The collision detection is based on the partial-view point cloud by examining whether there are any points within the gripper-occupied grid, where the gripper is simplified into three cubes. This step provides a safety guarantee in most cases.

The second step is to perform a gripper-centering process. It comes from an observation that if the two fingertips of the gripper contact the object surface sequentially, the earlier contact may push the object away and cause a failed grasp. The main reason is that the GraspNet-1Billion dataset does not restrict the gripper fingertips to have the same distance to the object. Thus, we perform the gripper centering process by calculating the distance from both fingertips to their contact points and translating the gripper along the connection direction of the fingertips to ensure the same moving distance from both fingertips to their contact points. We define the “contact points” as the outermost points inside the gripper space. We transform the partial-view point cloud to the gripper frame based on the predicted grasp poses. It is possible that the actual contact points are not visible in the partial view. However, we found that this step ensures the gripper is center-located in most real-world cases.

The above two processes are implemented on GPU with matrix computation and take 80 ms for 100 grasp poses. During the execution phase, our method outputs the top-100 ranking grasp poses, and a self-implemented grasp planner is used to select a target grasp pose. Further details are provided in Section V-B.

## V. EXPERIMENTAL SETUP

To verify the performance of our grasp perception system, we embed it with real robot platforms and conduct grasping experiments.

### A. Hardware and Human Subject

For the static scene grasping, we use a UR5 robot arm with an overhead camera. Intel RealSense D415 and D435 are adopted to evaluate the algorithm performance across depth sensors. We use a Robotiq-85 gripper. In [7], the authors used a customized silicone soft fingertip [54] that is designed for robust grasping. Since it is not available to us, we attach a soft table tennis rubber<sup>1</sup> to the fingertips, which is publicly available to any research group.

We also invite human subjects to conduct the bin picking and compare it with the AnyGrasp perception system. Human subjects are required to use a two-finger jaw for a fair comparison. The two-finger jaw has the same opening width as the robot gripper. Note that since we focus on the visual perception for grasping, we ask the volunteers to adopt an open-loop strategy during grasping. It means that they shall not adjust the grasp pose with tactile feedback after contacting the object. To enable fair comparison with human grasping, we attach the same soft rubber to the human hold jaw. Fig. 4 shows the robot and human subject settings.

For dynamic grasping, we use a Flexiv Rizon arm since it can update the servo targets more smoothly. An Intel RealSense L515 camera is attached to the wrist of the robot. We adopt the in-hand setting since the overhead camera might be occluded by the robot during tracking and choose the L515 since it has a larger depth range and can work robustly when the object is close to the camera. The D415 or D435 depth camera cannot generate depth information when the object distance to the camera is less than 15 cm. For safety reasons, we extend the fingertips of the Robotiq-85 gripper by mounting two extra 3D-printed parallel jaws. Fig. 5 illustrates the hardware setting in this experiment.

All the models in the following experiments run on a workstation with Ubuntu 20.04 system, Intel i9-10900K CPU, and Nvidia 2060 GPU. The code is written in Python.

### B. Experimental Procedure

For the parallel-jaw bin picking experiments, we conduct two independent trials for each experiment. The camera extrinsic parameters to the robot are obtained by Aruco marker detection and extra human measurement. In each trial, we randomly pour the objects onto the plate without too much human intervention. For each grasp attempt, our algorithm receives a single-view point cloud and predicts abundant grasp poses for the scene. For safety, we set a workspace limit. The limit includes the restriction on the plate plane (area enclosed by the blue dashed line on the plate, see Fig. 9) and the restriction on grasp pose orientation. Specifically, we restrict the angle between the grasp approach direction and the

<sup>1</sup>[https://www.amazon.com/Rubbers-DHS-Table-Tennis/s?rh=n%3A3419421%2Cp\\_89%3ADHS](https://www.amazon.com/Rubbers-DHS-Table-Tennis/s?rh=n%3A3419421%2Cp_89%3ADHS)

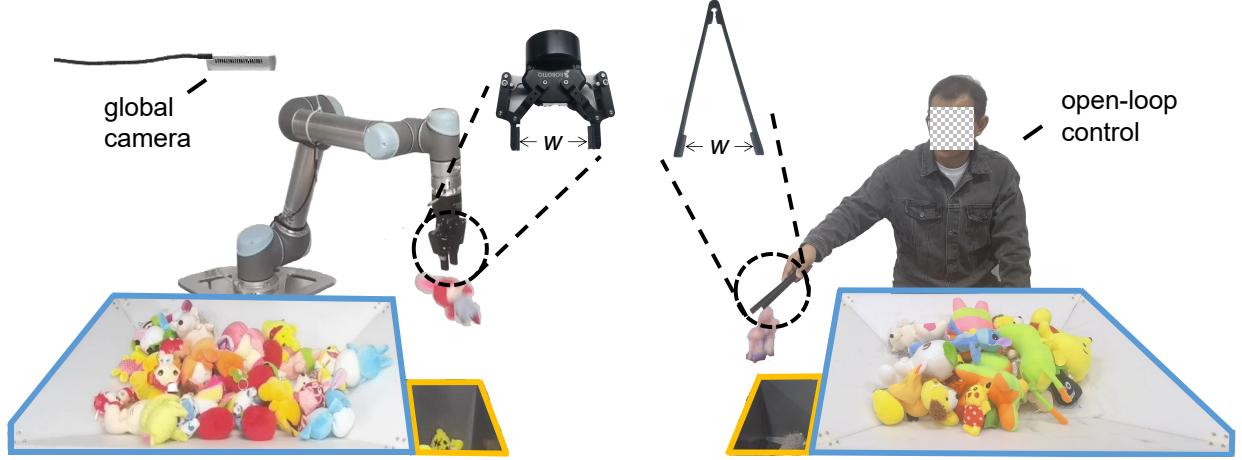


Fig. 4. Illustration of the hardware and volunteer setting in the static scene. Objects are moved from the object plate (surrounded by a blue line) to the target bin (surrounded by a yellow line). The robot is equipped with a global camera. We ensure the same fingertip material and the same opening width of the robot gripper and the human-held jaw. The human subjects are asked to perform an open-loop control, where adjusting the grasp pose with tactile feedback after contacting the object is not allowed.

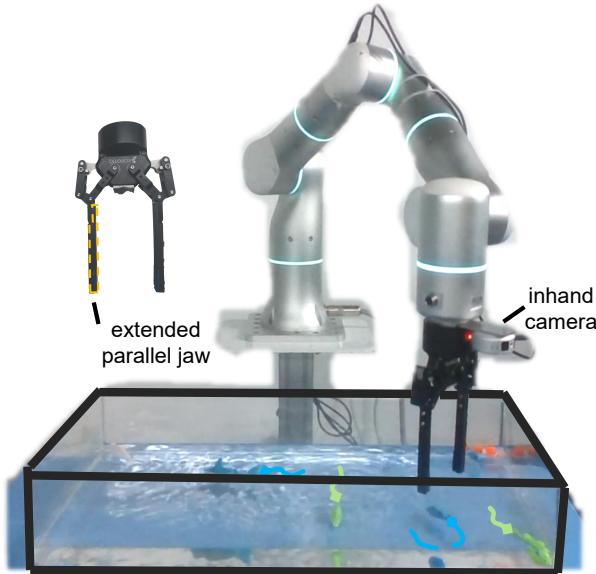


Fig. 5. Illustration of the hardware setting in the fish-catching task. An in-hand camera is attached to the robot's wrist. The moving trajectories of the robot fish are visualized with colored lines. We attach two 3D-printed parallel jaws to the Robotiq gripper.

vertical direction to be less than  $25^\circ$ , as we empirically found that a larger angle may result in an unreachable pose for the UR5 arm. The grasp pose without occlusion, located within the workspace and with the highest grasp score is selected.

To obtain a trajectory without collision with the scene during grasping, we set a waypoint pose by translating the grasp pose 10 cm backward along its approach direction. The waypoint and grasp pose are sent to the UR5 robot arm through UR script command `movels()` via Ethernet and the motion planner of UR5 would move the gripper to these poses sequentially. We found that this simple strategy can avoid most collision situations. The gripper will close after the robot reaches the target pose, grasp the object, lift it, and then move it to the target bin. During the grasping procedure,

the objects may scatter outside of the plane workspace. We would manually push these objects back. The operator will record whether the object is successfully moved to the target bin. Each trial of experiments is regarded as finished if the algorithm cannot estimate grasp poses for the scene ten times.

For the human bin-picking task, two volunteers are invited to conduct the grasping experiments and each is required to clear the plate twice.

For the dynamic fish-catching experiment, we conduct five independent trials. The camera extrinsic is obtained by measuring the customized camera holder. The pseudo-code of our dynamic grasping algorithm is illustrated in Alg. 1. In each fish-catching trial, we first generate abundant grasp poses for the scene (Line. 6 in Alg. 1) and select the best grasp pose  $\mathcal{G}_1^*$  for the first frame (Line. 12 in Alg. 1) using the same procedure in the static scene. For simplicity, we assume that the grasp pose has already been transformed into the robot coordinate system using the camera extrinsic parameters. Then the robot enters the pre-grasp servoing stage. The grasp perception system keeps active during this process and generates updated grasp pose  $\mathcal{G}_*^t$  of time step  $t$  (Line. 14 in Alg. 1) that has a high association score with the previous grasp pose  $\mathcal{G}_{*}^{t-1}$ . Empirically, we found that it is important to predict the future grasp pose when catching the moving fish since it takes some time to close the gripper. Hence, the selected grasp poses during the tracking procedure are saved to a temporary buffer of length 10 for future grasp pose prediction. The system will predict a future grasp pose  $\tilde{\mathcal{G}}_*^t$  (Line. 16 in Alg. 1) by adding the moving momentum calculated from the temporary buffer to the last grasp pose  $\mathcal{G}_*^t$ . In the pre-grasp servoing stage, we do not directly servo to the future grasp pose. Instead, the servoing target pose  $\mathbf{T}_p^t$  during the pre-grasp stage is obtained by translating  $\tilde{\mathcal{G}}_*^t$  along the  $z$ -axis by  $d_{\text{pregrasp}} = 3.5\text{cm}$  backward (Line. 24 in Alg. 1). This is to avoid touching the moving fish and changing their motion status during servoing. Meanwhile, we only preserve the rotation along the  $z$ -axis of the gripper and eliminate the

**Algorithm 1** Dynamic Grasping Algorithm

---

```

1: Initialize: robot.move( $\mathbf{T}_{\text{ready}}$ ), open gripper, empty buffer
2: Mark  $\mathbf{G}^0$  as empty
3: for time step  $t \leftarrow 1$  to  $\infty$  do
4:    $\mathbf{T}_{\text{tcp}}^t \leftarrow \mathbf{robot}.\text{pose}$ 
5:    $\mathcal{P}^t \leftarrow \mathbf{camera}.\text{perception}$ 
6:    $\mathbf{G}^t \leftarrow \text{GetGraspPoses}(\mathcal{P}^t)$ 
7:   if  $\mathbf{G}^t$  is empty then            $\triangleright$  lost tracking of objects
8:     robot.servo( $\mathbf{T}_{\text{ready}}$ ), empty buffer
9:     continue
10:    end if
11:    if  $\mathbf{G}^{t-1}$  is empty then         $\triangleright$  for the first frame
12:       $\mathcal{G}_*^t \leftarrow \arg \max_{\mathcal{G}^t \in \mathbf{G}^t} \text{Prob}(s = 1 \mid \mathcal{P}^t, \mathcal{G}^t)$ 
13:    else                          $\triangleright$  for the following tracking frames
14:       $\mathcal{G}_*^t \leftarrow \arg \max_{\mathcal{G}^t \in \mathbf{G}^t} \text{scores}(\mathcal{G}_*^{t-1}, \mathcal{G}^t)$ 
15:    end if
16:    Add  $\mathcal{G}_*^t$  into buffer and predict  $\tilde{\mathcal{G}}_*^t$  from buffer
17:    Calculate  $\Delta R$ ,  $\Delta t$  and  $\Delta t_{\text{xOy}}$  between  $\tilde{\mathcal{G}}_*^t$  and  $\mathbf{T}_{\text{tcp}}^t$ .
18:    if  $\Delta R \leq \delta_R$  and  $\Delta t \leq \delta_t$  and  $\Delta t_{\text{xOy}} \leq \delta_{t_{\text{xOy}}}$  then
19:       $\triangleright$  grasp when gripper is close enough to the fish
20:      robot.move( $\tilde{\mathcal{G}}_*^t$ ), gripper.close()
21:      Move robot to the throw pose and open gripper
22:      Re-initialize the system.  $\triangleright$  See L1 for details
23:    else            $\triangleright$  continue tracking with pregrasp pose
24:      Calculate pregrasp pose  $\mathbf{T}_p^t$  using  $\tilde{\mathcal{G}}_*^t$  and  $d_{\text{pregrasp}}$ 
25:      robot.servo( $\mathbf{T}_p^t$ )
26:    end if
27: end for

```

---

rotation along the other two axes to improve the tracking stability. The servoing target  $\mathbf{T}_p^t$  is sent to the Flexiv arm through Flexiv RDK and the robot would servo to the target pose (Line. 25 in Alg. 1). The ending criterion for the servoing process is that the 3D distance  $\Delta t$  between the predicted future grasp pose  $\tilde{\mathcal{G}}_*^t$  and the robot's current end-effector pose is less than  $\delta_t = 5.5\text{cm}$ , the 2D distance  $\Delta t_{\text{xOy}}$  in the horizontal plane is less than  $\delta_{t_{\text{xOy}}} = 2\text{cm}$ , and the angle  $\Delta R$  between two poses is less than  $\delta_R = 20^\circ$  (Line. 18 in Alg. 1). The robot will then enter the grasping stage which moves to the grasp pose and closes its gripper (Line. 19-21 in Alg. 1). If the tracked pose moves outside of the robot's workspace or camera's view, the robot will return to the initial state, empty the temporary buffer, and choose a new grasp target (Line. 7 in Alg. 1).

**C. Evaluation Metric**

In the area of robotic grasping, there are two different success rates used to evaluate the performance. The first metric, which we refer to as the attempt-centric success rate, is defined as the ratio of the number of successful grasp attempts to the total number of grasp attempts. This metric is commonly used in the literature [7]–[9], as it measures the ability of a grasping method to successfully perform grasps on a per-attempt basis.

The second metric, which we refer to as the object-centric success rate, is defined as the ratio of the number of successfully grasped objects to the total number of objects. This

metric measures the ability of a grasping method to adapt to different objects, which is less strict than the attempt-centric success rate since it does not take into account the number of grasp attempts per object. This metric has been previously adopted by [31] (they allow two attempts per object) and was referred to as “completion rate” in [29], [30].

It is important to note that we use the attempt-centric success rate in our work. However, to provide a clear comparison with previous work that adopted the object-centric success rate, we report the results of our proposed grasping method under both metrics in Table I.

TABLE I  
SUCCESS RATES OF DIFFERENT METHODS ON OUR REAL TEST SET.  
“DEX.” DENOTES DEXNET 4.0 AND “ANY.” DENOTES OUR ANYGRASP.

Object	Attempt-Centric Success Rate (%)			Object-Centric Success Rate (%)		
	Dex.	Any.	Human	Dex.	Any.	Human
Hardware	59.3	81.5	91.4	97.2	100.0	100.0
Snack	52.3	100.0	93.9	93.9	100.0	100.0
Ragdoll	87.4	100.0	96.6	100	100.0	100.0
Toy	72.8	93.1	91.8	99.6	99.6	100.0
Household	64.6	85.5	94.4	98.1	100.0	100.0
All	72.2	93.3	93.9	98.9	99.8	100.0

**VI. EXPERIMENTAL RESULTS**

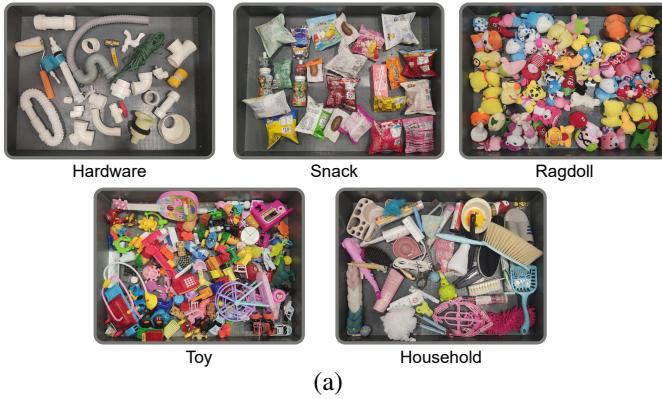
In this section, several experiments are demonstrated to evaluate the following properties of our 7-DoF grasp perception system: (i) generalization ability to different objects and sensors; (ii) accuracy on different kinds of objects compared with humans; (iii) temporal consistency for random moving objects; (iv) efficiency of the whole perception system.

**A. Static Scenes**

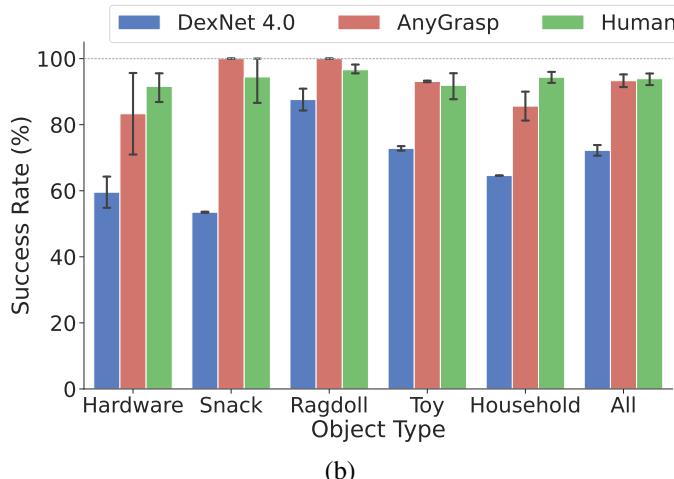
We first conduct grasping experiments in static scenes. To build a representative test set that can fully evaluate the grasp perception system, we collect several common categories of daily-life objects. In total, over 300 objects are collected. Fig. 6 (a) gives an overview of the objects. The size of the objects ranges from  $1.5 \times 1.5 \times 1.5 \text{ cm}^3$  to  $36 \times 4 \times 11.5 \text{ cm}^3$ .

We first compare the grasping system on daily objects with human operators using the same end-effector configuration and a previous method [7]. The overall grasping results on different objects are presented in Fig. 6 (b). The detailed numerical results are given in Tab. I. The videos can be found in Appendix B (Movie S1-S4). From Fig. 6 (b), we can see that our perception system is robust towards different categories of objects. On the large scale benchmark, its accuracy ranges from 81.5% to 100% on different kinds of objects, yielding 93.3% on average. Such grasping accuracy is on-par with human subjects, which is 93.9% on average. However, human subjects give more stable performance on different kinds of objects. The grasp accuracy ranges from 91.4% to 96.6%.

For each trial, the grasp perception system predicts the grasp poses in 100ms and the overall grasp decision time (including collision detection and pose adjustment) is less than 200 ms. Thus, the mean picks per hour (MPPH) only depends on the robot moving and gripper executing speed. With a single UR5



(a)



(b)

Fig. 6. (a) Example objects from the unseen object test set. (b) The grasp success rate of our system compared to a previous grasp pose detection system and the human’s performance using a parallel jaw.

arm that has a maximum end-link speed of 1m/s and a Robotiq gripper, our grasping system achieves over 900 MPPH, which is a dramatic improvement compared to previous state-of-the-art [7], 300 MPPH achieved by a dual-arm system. Such a metric can be further optimized in the industry using a higher-speed robot arm and gripper. We also observe that humans can achieve over 1,500 MPPH with their maximum speed, and on average they can achieve 1,000-1,200 MPPH.

We then show the performance of our algorithm on different depth sensors. We visualize the typical deviation maps of the two cameras we use in Fig. 7 (a). The deviation map is obtained by subtracting the mean depth value of 100 repeatedly captured images from a randomly chosen depth image. We can see that the depth given by the D435 camera presents a larger variance and can achieve up to  $\pm 5$ mm error. Even with this noise, our algorithm can still perform well, as illustrated in Fig. 7 (b). The video of grasping using a D435 camera can be found in Appendix B (Movie S5). The experiments demonstrate the robustness of our algorithm towards depth sensing noise, mainly owing to the training data collected with real sensors.

We further evaluate our grasp perception system on a challenging adversarial object set, which contains the 13 adversarial objects used in DexNet2.0 [5] and the 49 evaluation

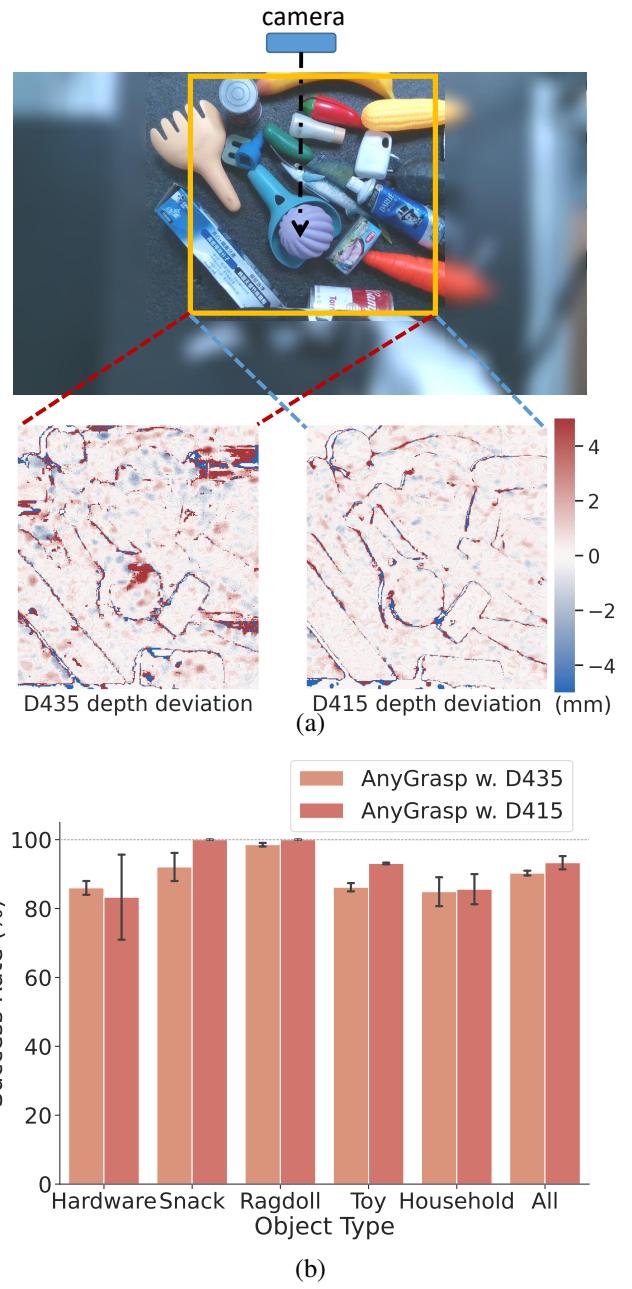


Fig. 7. (a) Typical depth deviation maps of two frequently adopted depth cameras. The distance unit is a millimeter. (b) The grasp success rate comparison of AnyGrasp when using these two cameras.

objects from EGAD [47]. The objects from the EGAD evaluation set are generated by the program to ensure the shape complexity and grasp difficulty. Fig. 8 shows the examples of the adversarial object set and the statistics of the success rate. Full videos can be found in Appendix B (Movie S6-S8, part of S3-S4). We can see that the accuracy decreases for our algorithm and the previous method [7]. However, humans can still perform stably. The main reason is that our grasp perception system would repeat the failed trials on an object without utilizing the feedback from each trial. It deserves more exploration in the future.

Finally, we set a challenging real-world task, where the robot is required to clean the fragments of a broken clay

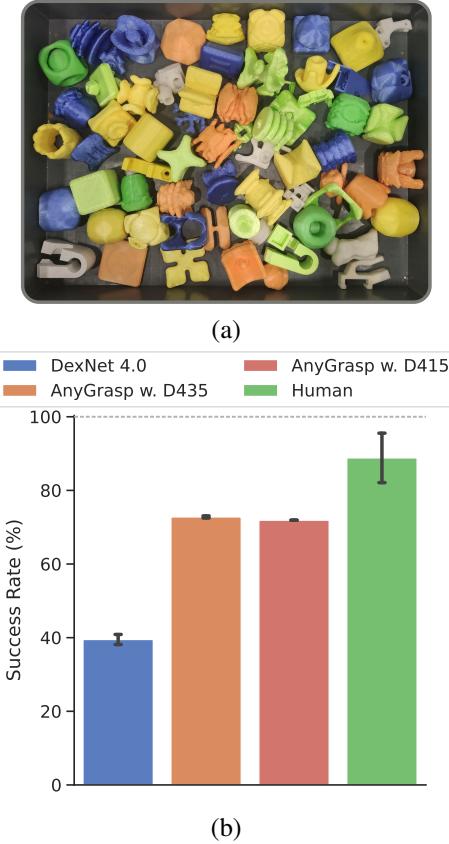


Fig. 8. (a) The 3D-printed adversarial objects. It can be seen that the object's surface is quite smooth. (b) The grasp success rate of different methods on the challenging adversarial object set.

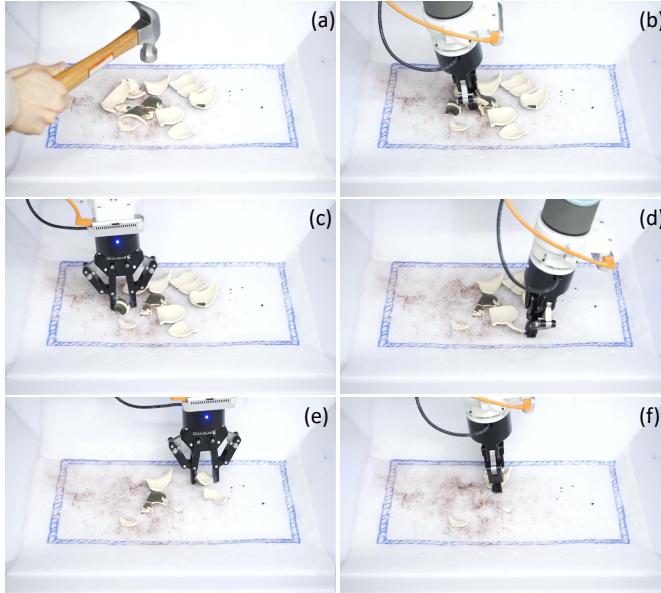


Fig. 9. Snapshots of the fragments cleaning task. It requires the grasp pose detection model to generate accurate poses for the thin pieces. The sequence order is from (a) to (f).

pot. It is challenging since the fragments are pretty thin, usually less than 3mm, which requires an accurate estimation of the grasp pose. The noisy depth perception further increases

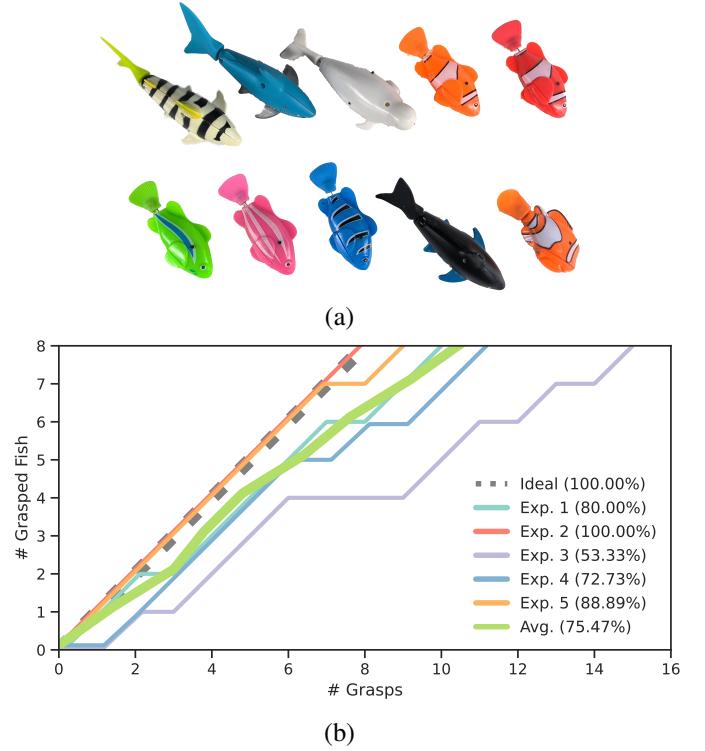


Fig. 10. (a) Different robot fish were used in our experiments. (b) Performance of the fish-catching experiments.

the difficulty. We haven't seen previous grasp pose detection algorithms demonstrate success in this scenario. Fig. 9 shows the snapshots of a robot cleaning the thin fragments under the guidance of the AnyGrasp perception system and the video is given in Appendix B (Movie S9).

### B. Dynamic Scenes

To verify the temporal consistency of the grasp perception system, we conduct a robot fish-catching experiment. A frequently used heuristic baseline [26] that keeps tracking the nearest grasp poses across frames is also implemented and compared with our temporal association module.

Previous literature examines the grasp pose tracking method in a human-robot hand-over setting [39]. For our fish-catching experiment, it poses extra challenges for the tracking system: (i) the underwater environment greatly decreases the contact friction between the gripper and the robot fish; (ii) the fish is pretty small and tolerates small final pose errors, while the handing object is usually larger; (iii) humans tend to stabilize the objects when the robot is about to grasp, while the fish keeps moving in the whole grasping process; (iv) the point cloud would be noisier due to the light reflection and refraction in water.

In our experiment, we randomly place 8 robot fish in the fish tank each time. Fig. 10 (a) shows different robot fish used in our experiments. Note that we have multiple instances for each kind of fish. The robot will try to catch all the swimming fish, and we record the success rate during grasping. The whole procedure is repeated 5 times, and we show the detailed performance in Fig. 10 (b). We illustrate the process of the

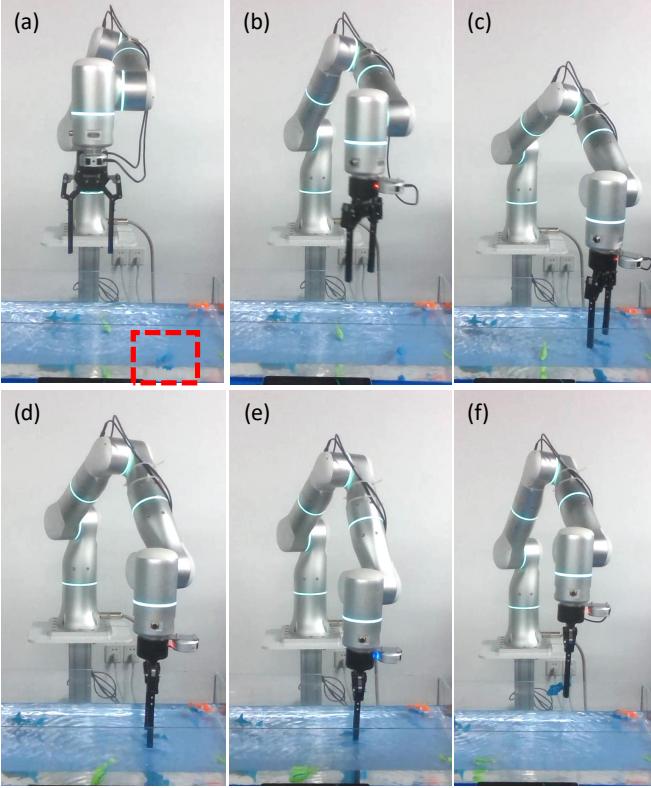


Fig. 11. Snapshots of the robot fish-catching procedure: (a) robot selects a grasp target, denoted by a red dashed line; (b)-(e) robot servos to the target grasp pose while updating the pose; (e) robot closes the gripper; (f) robot lifts the grasped fish.

robot catching a fish in Fig. 11 and the full video can be found in Appendix B (Movie S10). The average success rate achieves 75.5%. We can see that our grasp perception system is robust in this challenging dynamic grasping scenario.

We further conduct a failure analysis and found that the failed cases can be divided into five categories, as shown in Fig. 12. In nearly half of the failure cases, the fish slips away although the grasp pose is good, mainly due to the small friction in the water. The second and fourth reasons are that the predicted future grasp pose falls in front of or behind the fish. The main reason is that the robot fish would change its speed during moving and the historical momentum is outdated. The third reason is that the grasp quality is not good enough, and the gripper finger would push the fish away during closing. The last reason is the correspondence switch, which happens when there are two close and similar fish. The correspondence switch would inject noise into the history pose buffer and lead to a wrong predicted future grasp pose.

On the other hand, the heuristic method achieves an average success rate of 62.5%. During grasping, we found that this nearest target policy is more likely to fall behind the moving target during tracking, and takes a longer time to enter the grasping state. Specifically, it takes 12.7% more time than our method on the successful grasps on average.

Overall, the grasp perception system can run at 7 Hz on an Nvidia 2060 GPU.

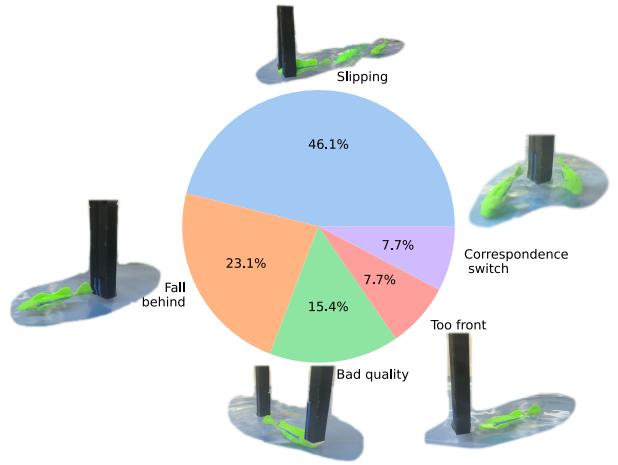


Fig. 12. Failure analysis in the fish catching experiments. We illustrate each failure case with an example image. Zoom in for more details and see the text for more explanation.

## VII. DISCUSSION

### A. Train in Simulation

As most methods and datasets provide training data in simulation, we try to figure out the performance gap between training on real data and simulated data. Since our dataset contains object 3D models and their 6D poses in the scene, we obtain simulated RGBD images by rendering the scene in PyRender [55]. The same model is trained on these simulated depth images. We also try the frequently adopted data augmentation technique that adds Gaussian noise to the depth images [5], [7] for better sim-to-real transfer.

We first analyze the detailed performance on the GraspNet-1Billion benchmark, which directly evaluates the grasp pose's quality based on object mesh using the force-closure analysis. The original metric computes the averaged AP for the Top-50 predicted grasp poses in the scene. To show more details, we compute the averaged AP for grasp poses ranging from Top-1 to Top-50. The performance on different test splits for our network trained with different strategies is shown in Fig. 13. We can see that adding Gaussian noise to the point cloud can improve the performance during evaluation, but still remains a large performance gap with directly trained on real data. Besides, their performance gap widens when the test set contains harder objects (the novel test scenes), especially for the high-score grasp poses.

Then we evaluate the model trained on simulated data with Gaussian noise in the real-world bin-picking experiment. The results are shown in Fig. 14. Movie S11 in Appendix B records the grasping process. We can see that the performance decreases a lot compared to the original model. Besides the lower success rate, we also observe that the network cannot generate grasp poses for a scene when there remain few objects. We conjecture the model trained in the simulation would generate fewer grasp poses with high scores in the real world due to the domain shift.

In this experiment, we showed that the frequently adopted sim-to-real technology in the grasping community is insufficient. We encourage future researchers to explore different

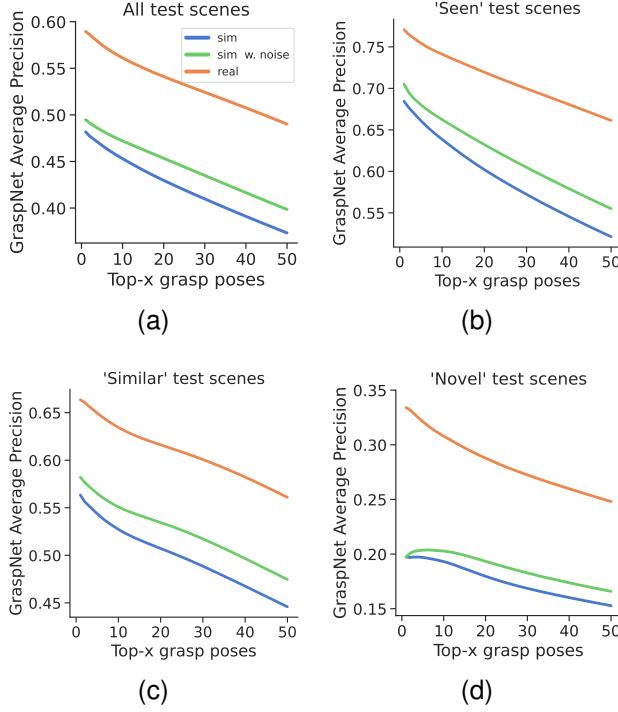


Fig. 13. Evaluation results on the GraspNet-1Billion test set when training with simulated and real-world data. From (a) to (d) we present the average precision (AP) on the whole test set, the test set with seen objects in the training set, the test set with similar objects in the training set, and the test set with novel objects. “sim” denotes the model trained with perfect simulated depth images, “sim w. noise” denotes the model trained with simulated depth with Gaussian noise.

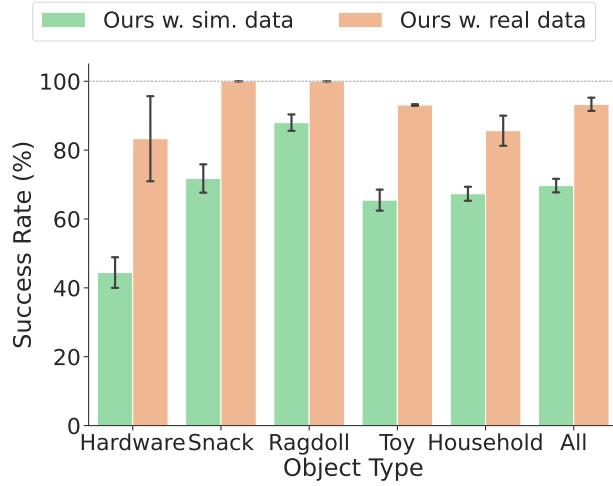


Fig. 14. The success rate of our system when training with real and simulated data. Note that here the “sim. data” denotes the simulated depth images with extra Gaussian noise.

sim-to-real techniques (e.g. [56]) and hope [9] can be used as a benchmark for this direction.

#### B. Influence of the Stable Score

To investigate how awareness of an object’s center of mass can benefit stable grasping, we conducted an analytical experiment. We selected a long, heavy object from our test

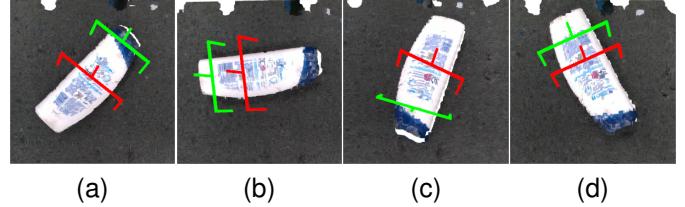


Fig. 15. Visualization of the top-ranking grasp poses with and without considering the stable score. The red grasp pose represents the case where we considered the stable score, while the green pose is the case without it. We see that the red grasp pose is closer to the COG of the object.

set as it better reflects the importance of COG awareness during grasping. We conducted grasping for the object with and without taking the stable score into account separately. In each experiment, we made 25 attempts, and we counted how many times in-hand slippage occurred. In total, 16 in-hand slippages (including 3 failed grasps) occurred when the stable score was not considered, and 11 in-hand slippages (including 2 failed grasps) occurred when the stable score was considered. Figure 15 illustrates some examples of the highest-ranking grasp poses with and without considering the stable score.

#### C. Dense Supervision Strategy

Besides providing data captured in real-world, another difference between GraspNet-1Billion [9] and other datasets [5], [47], [48], [57] is that the grasp pose annotation is much denser in [9]. For examples, Jacquard [57], DexNet 2.0 [5] and EGAD [47] generates 100 grasp poses on each object, ACRONYM [48] generates 2k grasp poses and [8] generates 34k grasp poses. In contrast, the GraspNet-1Billion generates over 10M grasp poses on each object. Here we also evaluate the influence of this factor. We train multiple models on the dataset with different numbers of annotations. The models are evaluated on the GraspNet-1Billion test set with the same metric introduced in the above subsection.

In Fig. 16, we show the model evaluation results with different training configurations. We downsample the training set at different dimensions, namely the grasp pose density on each object, the image amount, and the scene amount. These dimensions are uniformly downsampled by 10 times and 50 times respectively. We can see that when downsampling the grasp poses on objects by 10 times, the performance degradation is similar to decreasing the number of training images. Meanwhile, the performance decreases more in the novel object test scenes. When we downsample the training samples by 50 times, we can see that the degradation is still similar for the grasp pose dimension and the image dimension. These results suggest that the densely annotated grasp poses are as equal importance as the number of training images. Considering that they can be annotated automatically without extra human efforts in collecting images, it is a good trade to annotate dense grasp poses on objects.

Another interesting finding is that when we downsample the scene dimension of the training data, the model performance shows larger degradation on the test sets. When we downsample the scene by 50 times (namely 2 training scenes are

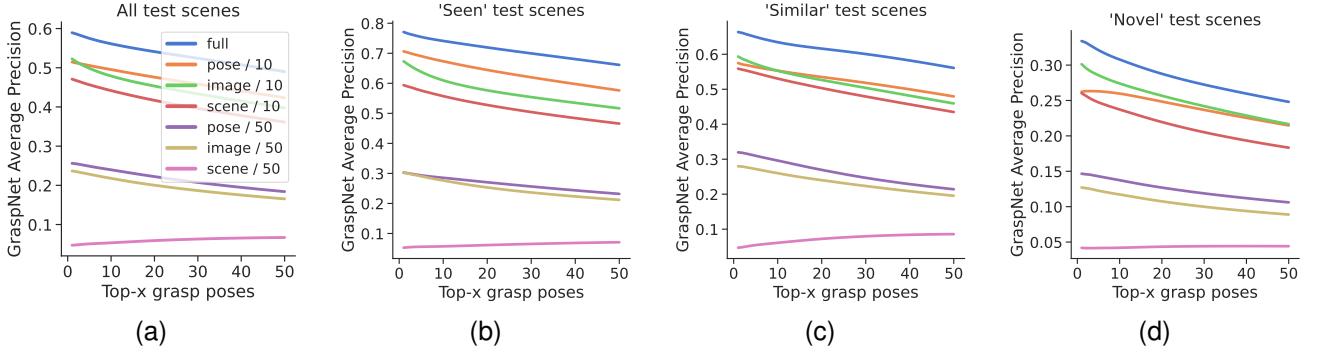


Fig. 16. Evaluation results on the GraspNet-1Billion test set when training with different portions of real-world data. “pose / 10” denotes we downsample the grasp pose density on each object by 1/10. “image / 10” denotes that we downsample the training images by 1/10. “scene / 10” denotes that we downsample the training scene by 1/10. So are the meaning of “pose / 50”, “image / 50” and “scene / 50”.

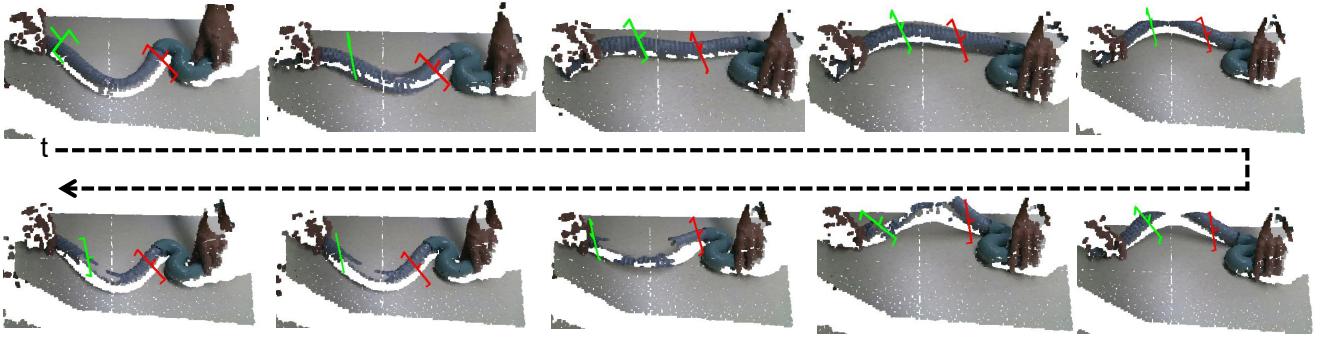


Fig. 17. Illustration of tracking two sampled grasp poses on a deformable, textureless tube. We can see that the grasp poses are stable in this case, even under severe noise input. The object 6D pose tracking algorithm cannot handle this scenario.

used), the model cannot converge. It suggests that the diverse scene might be more important for model training. We hope our analyses would be beneficial to future dataset design.

#### D. Comparison with 6D Pose Tracking

Recently, there is also research focusing on novel object 6D pose tracking [58]. Thus, it may be intuitive to adopt an unseen object 6D pose tracking pipeline for grasp tracking. Here, we explain the advantages of grasp pose tracking versus object 6D pose tracking. Firstly, the 6D pose tracking algorithm cannot handle deformable objects, and thus is not applicable for grasping in such cases. Secondly, the 6D pose tracking algorithm may be prone to large depth noise like incomplete point cloud, since it focuses on the object level features. But the grasp tracking algorithm can work robustly for a target grasp pose, even though other parts of the object are heavily occluded. In Fig. 17 we show an example illustrating these two cases.

#### E. Closed-loop Grasp Adjustment under Occlusion

Humans conduct closed-loop grasping in daily manipulation. Although our algorithm is also temporal continuous and can enable closed-loop grasping, it is fragile to the occlusion induced by the robot. When the robot approaches the object, the gripper would occlude the grasp target. Thus, the visual perception would fail. How to enable the closed-loop grasp

perception against occlusion is an open question. A missing piece here is tactile perception. It aids grasp slip detection and enables in-hand adjustment. This ability is out of the scope of this paper and will be our future work.

## VIII. CONCLUSION

In this paper, we presented AnyGrasp, a visual grasp perception system that can generate spatially dense and temporally smooth grasp poses. We proposed a unified model, wherein the dense prediction of the geometry processing module is the basis to enable smooth grasp pose tracking in the temporal association module. The collisions and object COG were implicitly learned through the supervision signal. Owing to our learning strategy with real-world data, the model performs robustly against different real-world depth sensor noise. Various experiments were conducted to verify the accuracy, robustness, and efficiency of our method. Finally, we analyzed the influence of several factors in the dataset design.

Although our method provides the closed-loop grasping ability, it cannot adjust the grasp pose using visual or tactile feedback. This ability can help the robot to recover from previous errors soon. Meanwhile, we only focused on visual grasping for the two-finger parallel gripper in this section. It would be significant to transfer such ability to different robotic hands. Our future research will be conducted accordingly.

## ACKNOWLEDGMENTS

This work is supported in part by the National Key R&D Program of China, No. 2017YFA0700800, Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), Shanghai Qi Zhi Institute, SHEITC (2018-RGZN-02046). We would like to thank Jiang Zou and Xiaolin Fang for their insightful discussions and comments.

*Authors' contributions:* H.-S.F. designed the experiments, devised the training method, implemented the static grasping experiments, collected the data and wrote the manuscript. C.W. implemented the neural network training, annotated the grasp label and edited the manuscript. H.F. implemented and conducted the dynamic grasping experiments. M.G. helped the data collection, wrote the API for data parsing and built the robot platform. J.L. generated the simulated data and assisted real-robot experiments. H.Y and W.L assisted real-robot experiments. Y.X. assisted the network training. C.L. supervised the project and provided the environment and funding support for this research.

## APPENDIX

## A. Object Collection

To comprehensively evaluate the algorithm performance in daily scenario, we decide to collect a fairly large scale test object set, as we found that the results varied on different objects. To construct the object set, we go to the supermarket, the hardware store and the toy store for procurement. The principle for selecting objects is that there exists some graspable place smaller than the gripper width on a object. In total, over 300 objects are collected, which is an order of magnitude larger than previous practice [7]. We believe such a large object test can provide a thorough evaluation to algorithms. Note that we do not collect objects with a large portion of transparent or black surface, since current depth sensor cannot give good prediction on these materials. Some recent papers [59], [60] focused on solving this problem with RGB information, but it is not fully addressed yet. The challenging adversarial object set is 3D printed with poly-lactic acid material. This results a smooth surface and makes the grasping even harder.

## B. Supplementary Videos

We record all the real robot experiments conducted in this paper. All the videos are uploaded to YouTube and will be permanently stored in support of this paper. The original length of the videos is over 12 hours. They are speeded up according to their importance to the paper. We honestly report the statistics of each video. Below we provide the links to these supplementary videos.

- S1: “AnyGrasp Demo: RealSense D415 camera on daily objects”, <https://youtu.be/dNnLgAGreec>
- S2: “DexNet4.0 tested on daily objects”, [https://youtu.be/vDqsrl\\_rtk8](https://youtu.be/vDqsrl_rtk8)
- S3: “AnyGrasp experiment: Human subject 1 on daily objects and adversarial objects”, <https://youtu.be/-h0yvFZDfko>
- S4: “AnyGrasp experiment: Human subject 2 on daily objects and adversarial objects”, <https://youtu.be/yItVN-Awjrg>
- S5: “AnyGrasp Demo: RealSense D435 camera on daily objects”, <https://youtu.be/7pgdbyLN0A4>
- S6: “DexNet4.0 tested on adversarial objects”, <https://youtu.be/9vOop28YReg>
- S7: “AnyGrasp Demo: RealSense D435 camera on adversarial objects”, [https://youtu.be/AK\\_nHgH4RBA](https://youtu.be/AK_nHgH4RBA)
- S8: “AnyGrasp Demo: RealSense D415 camera on adversarial objects”, <https://youtu.be/8FztVFRcvMY>
- S9: “AnyGrasp Demo: Cleaning fragments of a broken pot”, <https://youtu.be/s0SUw1vgtr8>
- S10: “AnyGrasp Demo: Robot Fish Catching by A Robot”, <https://youtu.be/2O7UoOxeLlk>
- S11: “AnyGrasp trained solely in simulation, tested in real world”, <https://youtu.be/lBKoa9OAIOA>

## REFERENCES

- [1] D. Baldauf and H. Deubel, “Attentional landscapes in reaching and grasping,” *Vision research*, vol. 50, no. 11, pp. 999–1013, 2010.
- [2] A. Bicchi and V. Kumar, “Robotic grasping and contact: A review,” in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1. IEEE, 2000, pp. 348–353.
- [3] J. Bohg, A. Morales, T. Asfour, and D. Kragic, “Data-driven grasp synthesis—a survey,” *IEEE Transactions on Robotics (T-RO)*, vol. 30, no. 2, pp. 289–309, 2014.
- [4] J. Redmon and A. Angelova, “Real-time grasp detection using convolutional neural networks,” *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, 2014.
- [5] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. Aparicio, and K. Goldberg, “Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics,” in *Proceedings of Robotics: Science and Systems (RSS)*, Cambridge, Massachusetts, July 2017.
- [6] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, “Grasp pose detection in point clouds,” *The International Journal of Robotics Research (IJRR)*, vol. 36, no. 13-14, pp. 1455–1473, 2017.
- [7] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, “Learning ambidextrous robot grasping policies,” *Science Robotics*, vol. 4, no. 26, 2019.
- [8] A. Mousavian, C. Eppner, and D. Fox, “6-dof grapsnet: Variational grasp generation for object manipulation,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2901–2910.
- [9] H.-S. Fang, C. Wang, M. Gou, and C. Lu, “Grasnet-1billion: A large-scale benchmark for general object grasping,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.
- [10] 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,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 13 459–13 466.
- [11] C. Wang, H.-S. Fang, M. Gou, H. Fang, J. Gao, and C. Lu, “Graspsness discovery in clutters for fast and accurate grasp detection,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 964–15 973.
- [12] A. Bicchi and V. Kumar, “Robotic grasping and contact: A review,” in *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 348–353.
- [13] K. B. Shimoga, “Robot grasp synthesis algorithms: A survey,” *The International Journal of Robotics Research*, vol. 15, no. 3, pp. 230–266, 1996.
- [14] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, “Automatic grasp planning using shape primitives,” in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 2. IEEE, 2003, pp. 1824–1829.
- [15] D. L. Bowers and R. Lumia, “Manipulation of unmodeled objects using intelligent grasping schemes,” *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 3, pp. 320–330, 2003.
- [16] A. Saxena, J. Driemeyer, J. Kearns, and A. Ng, “Robotic grasping of novel objects,” *Advances in neural information processing systems*, vol. 19, 2006.
- [17] A. Saxena, J. Driemeyer, and A. Y. Ng, “Robotic grasping of novel objects using vision,” *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 157–173, 2008.
- [18] A. Saxena, L. L. Wong, and A. Y. Ng, “Learning grasp strategies with partial shape information,” in *AAAI*, vol. 3, no. 2, 2008, pp. 1491–1494.
- [19] Q. V. Le, D. Kamm, A. F. Kara, and A. Y. Ng, “Learning to grasp objects with multiple contact points,” in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 5062–5069.
- [20] Y. Jiang, S. Moseson, and A. Saxena, “Efficient grasping from rgbd images: Learning using a new rectangle representation,” in *2011 IEEE International conference on robotics and automation*. IEEE, 2011, pp. 3304–3311.
- [21] I. Lenz, H. Lee, and A. Saxena, “Deep learning for detecting robotic grasps,” *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.
- [22] F.-J. Chu, R. Xu, and P. A. Vela, “Real-world multiobject, multigrasp detection,” *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 3355–3362, 2018.
- [23] J. Redmon and A. Angelova, “Real-time grasp detection using convolutional neural networks,” in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 1316–1322.

- [24] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 3406–3413.
- [25] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg, "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, 2016, pp. 1957–1964.
- [26] D. Morrison, P. Corke, and J. Leitner, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," *arXiv preprint arXiv:1804.05172*, 2018.
- [27] H. Liang, X. Ma, S. Li, M. Görner, S. Tang, B. Fang, F. Sun, and J. Zhang, "Pointnetgp: Detecting grasp configurations from point sets," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3629–3635.
- [28] L. Shao, F. Ferreira, M. Jorda, V. Nambiar, J. Luo, E. Solowjow, J. A. Ojea, O. Khatib, and J. Bohg, "Unigrasp: Learning a unified model to grasp with multifingered robotic hands," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2286–2293, 2020.
- [29] Y. Qin, R. Chen, H. Zhu, M. Song, J. Xu, and H. Su, "S4g: Amodal single-view single-shot se (3) grasp detection in cluttered scenes," in *Conference on robot learning*. PMLR, 2020, pp. 53–65.
- [30] B. Zhao, H. Zhang, X. Lan, H. Wang, Z. Tian, and N. Zheng, "Regnet: Region-based grasp network for end-to-end grasp detection in point clouds," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 13474–13480.
- [31] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox, "Contact-grasnet: Efficient 6-dof grasp generation in cluttered scenes," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 13438–13444.
- [32] Z. Dong, H. Tian, X. Bao, Y. Yan, and F. Chen, "Graspvdn: scene-oriented grasp estimation by learning vector representations of grasps," *Complex & Intelligent Systems*, vol. 8, no. 4, pp. 2911–2922, 2022.
- [33] J. Ichnowski, Y. Avigal, J. Kerr, and K. Goldberg, "Dex-nerf: Using a neural radiance field to grasp transparent objects," *arXiv preprint arXiv:2110.14217*, 2021.
- [34] R. Newbury, M. Gu, L. Chumbley, A. Mousavian, C. Eppner, J. Leitner, J. Bohg, A. Morales, T. Asfour, D. Kragic, D. Fox, and A. Cosgun, "Deep learning approaches to grasp synthesis: A review," 2022.
- [35] S. Kim, A. Shukla, and A. Billard, "Catching objects in flight," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1049–1065, 2014.
- [36] A. Menon, B. Cohen, and M. Likhachev, "Motion planning for smooth pickup of moving objects," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 453–460.
- [37] I. Akinola, J. Xu, S. Song, and P. K. Allen, "Dynamic grasping with reachability and motion awareness," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 9422–9429.
- [38] N. Marturi, M. Kopicki, A. Rastegarpanah, V. Rajasekaran, M. Adjigble, R. Stolkin, A. Leonardi, and Y. Bekiroglu, "Dynamic grasp and trajectory planning for moving objects," *Autonomous Robots*, vol. 43, no. 5, pp. 1241–1256, 2019.
- [39] W. Yang, C. Paxton, A. Mousavian, Y.-W. Chao, M. Cakmak, and D. Fox, "Reactive human-to-robot handovers of arbitrary objects," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 3118–3124.
- [40] 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.
- [41] 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.
- [42] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 4243–4250.
- [43] 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.
- [44] L. Wang, Y. Xiang, W. Yang, A. Mousavian, and D. Fox, "Goal-auxiliary actor-critic for 6d robotic grasping with point clouds," in *Conference on Robot Learning*. PMLR, 2022, pp. 70–80.
- [45] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.
- [46] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.
- [47] D. Morrison, P. Corke, and J. Leitner, "Egad! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4368–4375, 2020.
- [48] C. Eppner, A. Mousavian, and D. Fox, "Acronym: A large-scale grasp dataset based on simulation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6222–6227.
- [49] H. Deubel and W. X. Schneider, "Attentional selection in sequential manual movements, movements around an obstacle and in grasping," in *Attention in action*. Psychology Press, 2004, pp. 85–108.
- [50] F. Spenrath and A. Pott, "Using neural networks for heuristic grasp planning in random bin picking," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, 2018, pp. 258–263.
- [51] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 18661–18673, 2020.
- [52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [53] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.
- [54] M. Guo, D. V. Gealy, J. Liang, J. Mahler, A. Goncalves, S. McKinley, J. A. Ojea, and K. Goldberg, "Design of parallel-jaw gripper tip surfaces for robust grasping," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 2831–2838.
- [55] M. Matl *et al.*, "Pyrender," 2019.
- [56] M. Moosmann, F. Spenrath, J. Rosport, P. Melzer, W. Kraus, R. Bornmann, and M. F. Huber, "Transfer learning for machine learning-based detection and separation of entanglements in bin-picking applications," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 1123–1130.
- [57] A. Depierre, E. Dellandréa, and L. Chen, "Jacquard: A large scale dataset for robotic grasp detection," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3511–3516.
- [58] B. Wen and K. Bekris, "Bundlerec: 6d pose tracking for novel objects without instance or category-level 3d models," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8067–8074.
- [59] S. Sajjan, M. Moore, M. Pan, G. Nagaraja, J. Lee, A. Zeng, and S. Song, "Clear grasp: 3d shape estimation of transparent objects for manipulation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3634–3642.
- [60] H. Fang, H.-S. Fang, S. Xu, and C. Lu, "Transcg: A large-scale real-world dataset for transparent object depth completion and grasping," *arXiv preprint arXiv:2202.08471*, 2022.