This CVPR 2020 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

GraspNet-1Billion: A Large-Scale Benchmark for General Object Grasping

Hao-Shu Fang, Chenxi Wang, Minghao Gou. Cewu Lu[1]
Shanghai Jiao Tong University
fhaoshu@gmail.com, {wcx1997,gmh2015,lucewu}@sjtu.edu.cn

Abstract

Object grasping is critical for many applications, which is also a challenging computer vision problem. However, for cluttered scene, current researches suffer from the problems of insufficient training data and the lacking of evaluation benchmarks. In this work, we contribute a large-scale grasp pose detection dataset with a unified evaluation system. Our dataset contains 97,280 RGB-D image with over one billion grasp poses. Meanwhile, our evaluation system directly reports whether a grasping is successful by analytic computation, which is able to evaluate any kind of grasp poses without exhaustively labeling ground-truth. In addition, we propose an end-to-end grasp pose prediction network given point cloud inputs, where we learn approaching direction and operation parameters in a decoupled manner. A novel grasp affinity field is also designed to improve the grasping robustness. We conduct extensive experiments to show that our dataset and evaluation system can align well with real-world experiments and our proposed network achieves the state-of-the-art performance. Our dataset, source code and models are publicly available at www.graspnet.net.

1. Introduction

Object grasping is a fundamental problem and has many applications in industry, agriculture and service trade. The key of grasping is to detect the grasp poses given visual inputs (image or point cloud) and it has drawn many attentions in computer vision community [11, 30].

Though important, there are currently two main hindrances to obtaining further performance gains in this area. Firstly, the grasp poses have different representations including rectangle [36] and 6D pose [41] representation and are evaluated with different metrics [16, 14, 41] correspondingly. The difference in evaluation metrics makes it difficult to compare these methods directly in a unified manner, while evaluating with real robots would dramatically increase the evaluation cost. Secondly, it is difficult to obtain large-scale high quality training data [5]. Previous datasets annotated by human [16, 50, 7] are usually small in scale and only provide sparse annotations. While obtaining training data from the simulated environment [26, 9, 48] can generate large scale datasets, the visual domain gap between simulation and reality would inevitably degrade the performance of algorithms in real-world application.

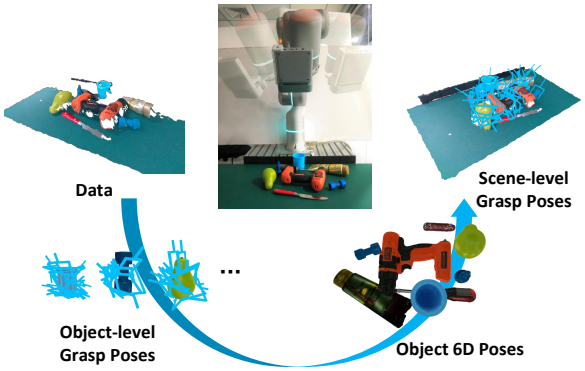


Figure 1. Our methodology for building the dataset. We collect data with real-world sensors and annotate grasp poses for every single object by analytic computation. Object 6D poses are manually annotated to project the grasp poses from object coordinate to the scene coordinate. Such methodology greatly reduces the labor of annotating grasp poses. Our dataset is both densely annotated and visually coherent with real world.

To form a solid foundation for algorithms built upon, it is important for a benchmark to i) provide data that aligns well with the visual perception from real world sensors, ii) be densely and accurately annotated with large-scale grasp pose ground-truth and iii) evaluate grasp poses with different representations in a unified manner. This is nontrivial, especially when it comes to the data annotation. Given an image or scene, it's hard for us to manually annotate endless grasp poses in continuous space. We circumvent this issue by exploring a new direction, that is, collecting data

[1]Cewu Lu is corresponding author, member of Qing Yuan Research Institute and MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, China

from the real world and annotating them by analytic computation in simulation, which leverages the advantages from both sides.

Specifically, inspired by previous literature [41], we propose a two-step pipeline to generate tremendous grasp poses for a scene. Thanks to our automatic annotation process, we built the first large-scale in-the-wild grasp pose dataset that can serve as a base for training and evaluating grasp pose detection algorithms. Our dataset contains **97,280** RGB-D images taken from different viewpoints of over 190 cluttered scenes. For all 88 objects in our dataset, we provide accurate 3D mesh models. Each scene is densely annotated with object 6D poses and grasp poses, bringing over **one billion** grasp poses, which is 5 orders of magnitude larger than previous datasets. Moreover, embedded with an online evaluation system, our benchmark is able to evaluate current mainstream grasping detection algorithms in a unified manner. Experiments also demonstrate that our benchmark can align well with real-world experiments. Fig 1 shows the methodology for building our dataset.

Given such a large scale dataset, we further propose a novel method for learning grasp poses. For better geometric reasoning and context encoding, we propose an end-to-end 3D based grasp pose detection network. Instead of predicting grasp pose matrix directly, our network seeks a more robust learning way that learns approaching direction and operation parameters (e.g. in-plane rotation, grasp width) explicitly under a unified objective. Moreover, to improve the perturbation resistance of the grasp pose, we propose a novel representation called *grasp affinity fields* to make our network being robust to perturbation. Experiments demonstrate the effectiveness and efficiency of our proposed method.

2. Related Work

In this section, we first review deep learning based grasping detection algorithms, followed by related datasets in this area. Point cloud based deep learning methods are also briefly reviewed.

Deep Learning Based Grasping Prediction Algorithms For deep learning based grasping detection algorithms, they can be divided into three main categories. The most popular one is to detect a graspable rectangle based on RGB-D image input [16, 21, 36, 13, 30, 22, 26, 50, 1, 2, 7, 27, 28]. Lenz *et al*. [21] proposed a cascaded method with two networks that first prunes out unlikely grasps and then evaluates the remaining grasps with a larger network. Redmon *et al*. [36] proposed a different network structure that directly regresses the grasp poses in a single step manner, which is faster and more accurate. Mahler *et al*. [26] proposed a grasp quality CNN to predict the robustness scores of grasping candidates. Zhang [50] and Chu [7] extended it to multi-object scenarios. The grasp poses generated by these methods are constrained in 2D plane which limits the degree of freedom of grasp poses. With the rapid development in monocular object 6D pose estimation [17, 45], some researchers [8] predict 6D poses of the objects and project predefined grasp poses to the scene. Such methods have no limitation of grasping orientation, but require a prior knowledge about the object shape. Recently, starting from [42] there is a new line of researches [41, 24, 28, 35] that propose grasping candidates on partial observed point clouds and output a classification score for each candidate using 3D CNN. Such methods require no prior knowledge about the objects. Currently, these methods are evaluated in their own metrics and hard to compare to others.

Grasping Dataset Cornell grasping dataset [16] first proposed rectangle representation for grasping detection in images. Single object RGB-D images are provided with rectangle grasp poses. [7, 50] built datasets with the same protocol but extend to multi-object scenarios. These grasp poses are annotated by human. [30, 22] collect annotations with real robot experiments. These data labeling methods are time consuming and require strong hardware support. To avoid such problem, some recent works explore using simulated environment [26, 9, 48, 28, 4] to anotate grasp poses. They can generate a much larger scale dataset but the domain gap of visual perception is always a hindrance. Beyond rectangle based annotation, GraspSeg [2] provides pixel-wise annotations for grasp-affordance segmentation and object segmentation. For 6D pose estimation, [45] contributes a dataset with 21 objects and 92 scenes. These datasets mainly focus on a subarea of grasp pose detection. In this work, we aim to build a dataset that is much larger in scale and diversity and covers main aspects of object grasping detection.

Point Cloud Based Deep Learning Qi *et al*. first proposed PointNet [33] to directly learn features from raw point cloud inputs. After that, many methods [34, 38, 3, 23, 12, 43, 39, 40, 20, 19, 49, 47, 44, 46, 15] are proposed to perform point cloud classification and segmentation. Beyond that, some recent works [31, 37, 32] extended the PointNet framework to the area of 3D object detection. The most similar network structure to ours is that of Qin *et al*. [35], which also predicted grasp poses based on PointNet. In this work, we design an end-to-end network with a new representation of grasp pose rather than direct regression.

3. GraspNet-1Billion

We next describe the main features of our dataset and how we build it.

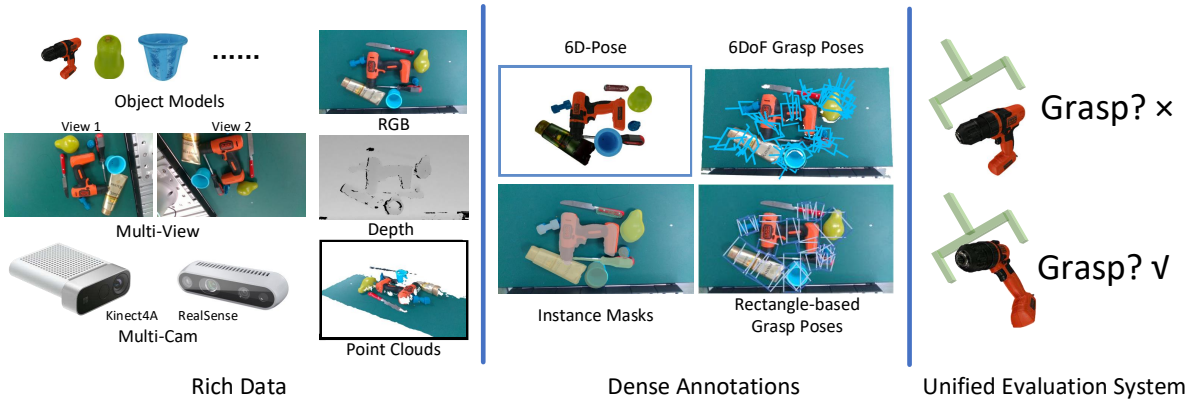


Figure 2. The key components of our dataset. RGB-D images are taken using both RealSense camera and Kinect camera from different views. The 6D pose of each object, the grasp poses, the rectangle grasp poses and the instance masks are annotated. A unified evaluation system is also provided.

3.1. Overview

Previous grasping dataset either focuses on isolated object [16, 26, 9, 48] or only labels one grasp per scene [30, 22]. Few datasets consider multi-object-multi-grasp setting and are small in scale [50, 7] due to the labor of annotation. Moreover, most of the datasets adopt the rectangle based representation [16] of grasp pose, which constrains the space for placing the gripper. To overcome these issues, we propose a large-scale dataset in cluttered scenario with dense and rich annotations for grasp pose prediction named *GraspNet-1Billion*. Our dataset contains 88 daily objects with high quality 3D mesh models. The images are collected from 190 cluttered scenes, each contributes 512 RGB-D images captured by two different cameras, bringing 97,280 images in total. For each image, we densely annotate 6-DoF grasp poses by analytic computation of force closure [29]. The grasp poses for each scene varies from 3,000,000 to 9,000,000, and in total our dataset contains over 1.1 billion grasp poses. Besides, we also provide accurate object 6D pose annotations, rectangle based grasp poses, object masks and bounding boxes. Each frame is also associated with a camera pose, thus multi-view point cloud can be easily fused. Fig 2 illustrates the key components of our dataset.

3.2. Data Collection

We select 32 objects that are suitable for grasping from the YCB dataset [6], 13 adversarial objects from DexNet 2.0 [26] and collects 43 objects of our own to construct our object set. The objects have suitable sizes for grasping and are diverse in shape, texture, size, material, etc. We believe that diverse local geometry can bring better generalization ability for the algorithm. To collect data of cluttered scene, we attach the cameras to a robot arm since it can repeat the trajectory precisely and help automatizing the collecting process. Camera calibration is conducted before data collection to obtain accurate camera poses. Considering

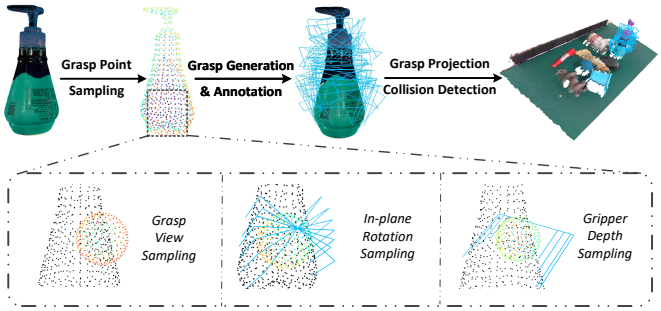


Figure 3. Grasp pose annotation pipeline. The grasp point is firstly sampled from point cloud. Then the grasp view, the in-plane rotation and the gripper depth are sampled and evaluated. Finally, the grasps are projected on the scene using the 6D pose of each object. Collision detection is also conducted to avoid the collision between grasps and background or other object.

different quality of depth image will inevitably affect the algorithms, we adopt two popular RGB-D cameras, Intel RealSense 435 and Kinect 4 Azure, to simultaneously capture the scene and provide rich data. For each scene, we randomly pick around 10 objects from our whole object set and place them in a cluttered manner. The robot arm then moves along a fixed trajectory that covers 256 distinct viewpoints on a quarter sphere. A synchronized image pair from both RGB-D cameras as well as their camera poses will be saved. Note that for camera calibration, we conducted camera extrinsic parameter calibration to avoid the errors from forward kinematics. To be specific, we took photos of a fixed ArUco marker at the 256 data collection points. The camera poses *w.r.t* the marker coordinate system were obtained. These camera poses are pretty accurate and work well for our dataset. The detail setting of our data collection process will be provided in supplementary materials.

3.3. Data Annotation

6D Pose Annotation With 97,280 images in total, it would be labor consuming to annotate 6D poses for each

Dataset	Grasps / scene	Objects / scene	Grasp label	6D pose	Total objects	Total grasps	Total images	Modality	Data source
Cornell [16]	∼8	1	Rect.	No	240	8019	1035	RGB-D	1 Cam.
Pinto *et al*. [30]	1	-	Rect.	No	150	50K	50K	RGB-D	1 Cam.
Levine *et al*. [22]	1	-	Rect.	No	-	800K	800K	RGB-D	1 Cam.
Mahler *et al*. [26]	1	1	Rect.	No	1,500	6.7M	6.7M	Depth	Sim.
Jacquard [9]	∼20	1	Rect.	No	11K	1.1M	54K	RGB-D	Sim.
Zhang *et al*. [50]	∼20	∼3	Rect.	No	-	100K	4683	RGB	1 Cam.
Multi-Object [7]	∼30	∼4	Rect.	No	-	2904	96	RGB-D	1 Cam.
VR-Grasping-101 [48]	100	1	6-DOF	Yes	101	4.8M	10K	RGB-D	Sim.
YCB-Video [45]	None	∼5	None	Yes	21	None	134K	RGB-D	1 Cam.
GraspNet (ours)	**3∼9M**	**∼10**	**6-DOF**	**Yes**	**88**	**∼1.2B**	**97K**	**RGB-D**	**2 Cams.**

Table 1. Summary of the properties of publicly available grasp datasets. "Rect.", "Cam." and "Sim." are short for Rectangle, Camera and Simulation respectively. "-" denotes the number is unknown.

frame. Thanks to the camera poses recorded, we only need to annotate 6D poses for the first frame of each scene. The 6D poses will then be propagated to the remaining frames by:

$$\mathbf{P}_i^j = \mathbf{cam}_i^{-1}\mathbf{cam}_0\mathbf{P}_0^j, \tag{1}$$

where $\mathbf{P}_i^j$ is the 6D pose of object j at frame i and $\mathbf{cam}_i$ is the camera pose of frame i. All the 6D pose annotations were carefully refined and double-checked by several annotators to ensure high quality. Object masks and bounding boxes are also obtained by projecting objects onto the images using 6D poses.

Grasp Pose Annotation Different from labels in common vision tasks, grasp poses distribute in a large and continuous search space, which brings infinite annotations. Annotating each scene manually would be dramatically labor expensive. Considering all the objects are known, we propose a two stage automated pipeline for grasp pose annotation, which is illustrated in Fig. 3.

First, grasp poses are sampled and annotated for each single object. To achieve that, high quality mesh models are downsampled such that the sampled points (called grasp points) are uniformly distributed in voxel space. For each grasp point, we sample V views uniformly distributed in a spherical space. Grasp candidates are searched in a two dimensional grid $D \times A$, where D is the set of gripper depths and A is the set of in-plane rotation angles. Gripper width is determined accordingly such that no empty grasp or collision occurs. Each grasp candidate will be assigned a confidence score based on the mesh model.

We adopt an analytic computation method to grade each grasp. The force-closure metric [29, 41] has been proved effective in grasp evaluation: given a grasp pose, the associated object and a friction coefficient μ, force-closure metric outputs a binary label indicating whether the grasp is antipodal under that coefficient. The result is computed based on physical rules, which is robust. Here we adopt an improved metric described in [24]. With $\Delta\mu = 0.1$ as interval, we decrease μ gradually from 1 to 0.1 step by step until the grasp is not antipodal. The grasp with lower friction coefficient μ has more probability of success. Thus we define our score s as:

$$s = 1.1 - \mu, \tag{2}$$

such that s lies in $(0, 1]$.

Second, for each scene, we project these grasps to the corresponding objects based on the annotated 6D object poses:

$$\begin{aligned} \mathbf{P}^i &= \mathbf{cam}_0\mathbf{P}_0^i, \\ \mathbb{G}_{(w)}^i &= \mathbf{P}^i \cdot \mathbb{G}_{(o)}^i, \end{aligned} \tag{3}$$

where $\mathbf{P}^i$ is the 6D pose of the i-th object in the world frame, $\mathbb{G}_{(o)}^i$ is a set of grasp poses in the object frame and $\mathbb{G}_{(w)}^i$ contains the corresponding poses in the world frame. Besides, collision check is performed to avoid invalid grasps. Following [41], we adopt the simplified gripper model as shown in Fig. 4 and check whether there are object points in this area. After these two steps we can generate densely distributed grasp set $\mathbb{G}_{(w)}$ for each scene. According to statistics, the ratio of positive and negative labels in our dataset is around 1:2. We conduct real world experiment in Sec. 5 using our robot arm and verify that our generated grasp poses can align well with real world grasping.

3.4. Evaluation

Dataset Split For our 190 scenes, we use 100 for training and 90 for testing. Specifically, we further divide our test sets into 3 categories: 30 scenes with seen objects, 30 with unseen but similar objects and 30 for novel objects. We hope that such setting can better evaluate the generalization ability of different methods.

New Metrics To evaluate the prediction performance of grasp pose, previous methods adopt the rectangle metric that consider a grasp as correct if: i) the rotation error is less than 30° and ii) the rectangle IOU is larger than 0.25.

There are several drawbacks of such metric. Firstly, it can only evaluate rectangle representation of grasp pose.

Secondly, the error tolerance is set rather high since the groundtruth annotations are not exhaustive. It might overestimate the performance of grasping algorithm. Currently, the Cornell dataset [16] has achieved over 99% accuracy. In this work, we adopt an online evaluation algorithm to evaluate the grasp accuracy.

We first illustrate how we classify whether a single grasp pose is true positive. For each predicted grasp pose $\hat{\mathbf{P}}_i$, we associate it with the target object by checking the point cloud inside the gripper. Then, similar to the process of generating grasp annotation, we can get a binary label for each grasp pose by force-closure metric, given different μ.

For cluttered scene, grasp pose prediction algorithms are expected to predict multiple grasps. Since for grasping, we usually conduct execution after the prediction, the percentage of true positive is more important. Thus, we adopt *Precision@k* as our evaluation metric, which measures the precision of top-k ranked grasps. AP_μ denotes the average *Precision@k* for k ranges from 1 to 50 given friction μ. Similar to COCO [25], we report AP_μ at different μ. Specifically, we denote **AP** for the average of AP_μ ranging from $\mu = 0.2$ to $\mu = 1.0$, with $\Delta\mu = 0.2$ as interval.

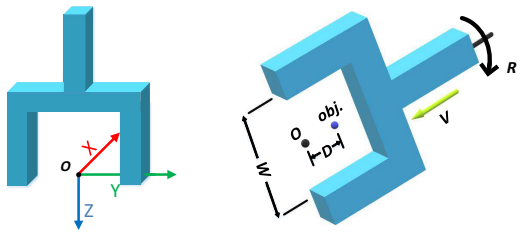
To avoid dominated by similar grasp poses or grasp poses from single object, we run a pose-NMS before evaluation. For the details of pose-NMS please refer to the supplementary file.

3.5. Discussion

In this work, we aim to provide a general benchmark for the problem of object grasping. The grasping problem can be decoupled as: i) predict all possible grasp poses (by CV community) and ii) conduct motion planning for specific robotic setting and grasp (by robotics community). For our benchmark, we focus on the vision problem and decouple the labels from the design choices of the robotic environment as much as possible. We provided multiple cameras and multiple views, simplified the gripper model and the collision detection to improve the generality of the dataset. The motion planning and collisions with real gripper and robot arm are not considered as they are related to the robotic environment and should be solved at run-time. We hope our dataset can facilitate fair comparison among different grasp pose detection algorithms.

We compare our datasets with other publicly available grasp datasets. Table 1 summaries the main differences at several aspects. We can see that our dataset is much larger in scale and diversity. With our two-step annotation pipeline, we are able to collect real images with dense annotations, which leverages the advantages from both sides.

For grasp pose evaluation, due to the continuity in grasping space, there are in fact infinite feasible grasp poses. The previous method that pre-computed ground truth for evaluating grasping, no matter collected by human annotation [16] or simulation [9], cannot cover all feasible solution. In contrast, we do not pre-compute labels for the test set, but directly evaluate them by calculating the quality score using force closure metric [29]. Such evaluation method does not assume the representation of the grasp pose, thus is general in practice. Related APIs is made publicly available to facilitate the research in this area.



(a) (b)

Figure 4. (a) The coordinate frame of the gripper. (b) Our new representation of grasp pose. "obj." denotes object point. Our network needs to predict i) the approaching vector V, ii) the approaching distance from grasp point to the origin of gripper frame D, iii) the in-plane rotation around approaching axis R and iv) the gripper width W.

4. Method

We then introduce our end-to-end grasp pose detection network, which is illustrated in Fig. 5. Our grasp pose representation is introduced in 4.1. Accordingly, we mainly divide our pipeline into three parts: Approach Network, Operation Network and Tolerance Network.

4.1. Grasp Pose Representation

Similar to previous works [41, 24], we define the frame of the two-finger parallel gripper as Fig. 4(a). With the known gripper frame, grasp pose detection aims to predict the orientation and translation of the gripper under the camera frame, as well as the width of the gripper. We represent the grasp pose $\mathbf{G}$ as

$$\mathbf{G} = [\mathbf{R}\ \mathbf{t}\ w], \tag{4}$$

where $\mathbf{R} \in \mathbb{R}^{3\times3}$ denotes the gripper orientation, $\mathbf{t} \in \mathbb{R}^{3\times1}$ denotes the center of grasp and $w \in \mathbb{R}$ denotes the gripper width that is suitable for grasping the target object. For neural network, directly learning the rotation matrix in $\mathbb{R}^{3\times3}$ is not intuitive. The explicit constraints, such as the determinant of rotation matrix must equal one and the inverse of it is its transpose, are difficult to learn. Instead, we adopt the representation from 6D pose estimation [17] that decouples the orientation as viewpoint classification and in-plane rotation prediction. Our problem is then reformulated as follows without loss of generality: for a grasp point on the surface of objects, we predict the feasible approaching vectors, approaching distance, in-plane rotation along the approaching axis and a tight gripper width. Fig. 4(b) explains

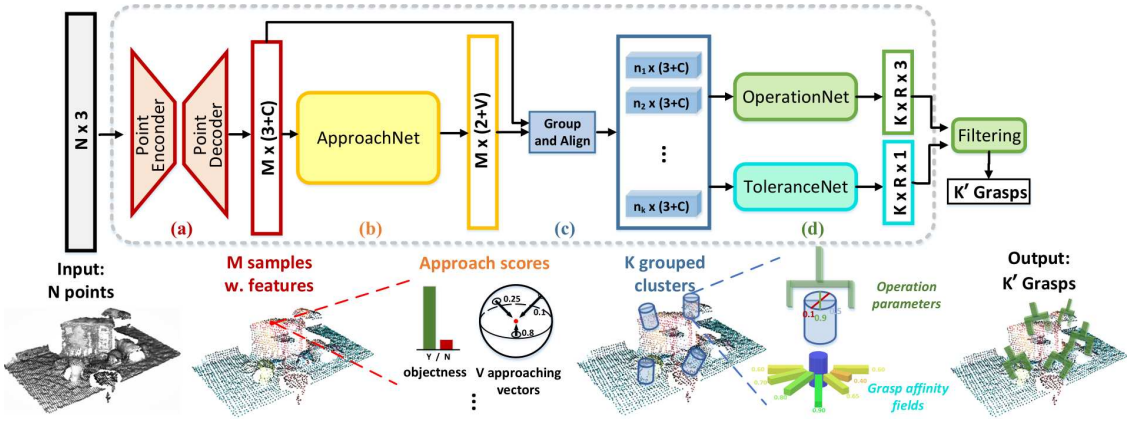


Figure 5. Overview of our end-to-end network. (a) For a scene point cloud with N point coordinates as input, a point encoder-decoder extracts cloud features and samples M points with C-dim features. (b) Approaching vectors are predicted by ApproachNet and are used to (c) grouped points in cylinder regions. (d) OperationNet predicts the operation parameters and ToleranceNet predicts the grasp robustness. See text for more details.

our formulation of grasp pose. Following such formulation, our network design is illustrated as follows.

4.2. Approach Network and Grasp Point Selection

The approaching vectors and feasible grasp points are jointly estimated by our Approach Network, since some directions are not suitable for grasping due to occlusion.

Base Network To build a solid foundation for viewpoint classification, we first use a base network for capturing well point cloud geometric features. In this work, we adopt PointNet++ [34] backbone network. Other networks like VoxelNet [51] can be also adopted. Taking a raw point cloud with size $N \times 3$ as input, our base network outputs a new set of points with C channels features. We subsample M points with farthest point sampling [10] to cover the whole scene.

Output Head We classify feasible approaching vectors into V predefined viewpoints. Meanwhile, for each point, the Approach Network outputs two values to predict its confidence of graspable or not. Therefore, the output of our proposal generation network is $M \times (2 + V)$, where 2 denotes the binary class of graspable or not and V denotes the number of predefined approaching vectors.

Loss Function For each candidate point, we assign it a binary label indicating whether it is graspable or not. First, points which are not on the objects are assigned negative labels. Next, for points on the objects, we found those who have at least one graspable ground-truth within 5mm radius neighbor area. Their graspable scores are assigned as 1. Finally, points on the objects but cannot find reference ground-truth grasps are ignored, which do not contribute to the training objective.

For each graspable point, V virtual approaching vectors are sampled around it under the camera frame. Now, we can define the approaching vector of j^{th} virtual view of i^{th} graspable point as v_{ij}. We then look for its ground-truth reference vector $\hat{v}_{ij}$ on the sphere space of the i^{th} point. Similarly, we only consider reference vectors that are within 5 degree bound. With such definition, our target function for an input point cloud is defined as follows:

$$L^A(\{c_i\},\{s_{ij}\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(c_i, c_i^*) + \lambda_1 \frac{1}{N_{reg}}\sum_i \sum_j c_i^* \mathbf{1}(|v_{ij}, v_{ij}^*| < 5^\circ) L_{reg}(s_{ij}, s_{ij}^*). \quad (5)$$

Here, c_i denotes the binary prediction of graspable or not for point i. c_i^* is assigned 1 if point i is positive and 0 if negative. s_{ij} denotes the predicted confidence score for viewpoint j of point i. s_{ij}^* is the corresponding ground-truth, which is obtained by choosing the maximum grasp confidence (Eqn. 2) from that viewpoint. $|v_{ij}, v_{ij}^*|$ denotes degree difference. Indicator function $\mathbf{1}()$ constrains the loss on approaching vectors that has a nearby groundtruth within 5 degree bound. Here for L_{cls} we use a two class softmax loss, while for L_{reg} we use the smooth L_1 loss.

4.3. Operation Network

After getting approaching vectors from graspable points, we further predict in-plane rotation, approaching distance, gripper width and grasp confidence, which is important for operation. Here, grasp confidence have 10 levels (Eqn. 2).

Cylinder Region Transformation Before forwarding through the operation network, we build a unified representation for each grasp candidate. Since approaching distance is relatively less sensitive, we divide it into K bins. For each given distance d_k, we sample points inside the cylinder centered along the approaching vectors to a fixed number. For better learning, all the sampled points are transformed into a new coordinate whose origin is the grasp point and z-axis

is $\mathbf{v_{ij}}$. The transformation matrix $\mathbf{O_{ij}}$ is calculated as:

$$\mathbf{O_{ij}} = [\mathbf{o_{ij}^1}, [0, -\mathbf{v_{ij}}^{(3)}, \mathbf{v_{ij}}^{(2)}]^{\mathrm{T}}, \mathbf{v_{ij}}],$$
$$\textit{where } \mathbf{o_{ij}^1} = [0, -\mathbf{v_{ij}}^{(3)}, \mathbf{v_{ij}}^{(2)}]^{\mathrm{T}} \times \mathbf{v_{ij}},$$

$\mathbf{v_{ij}}^{(k)}$ is the k-th element of $\mathbf{v_{ij}}$. After such transformation, candidate grasp poses has a unified representation and coordinate.

Rotation and Width It has been proved in previous literature [17] that for predicting in-plane rotation, classification could achieve better results than regression. Following such setting, our rotation network takes the aligned point cloud as input and predicts classification scores and normalized residuals for each binned rotation, as well as the corresponding grasp width and confidence. It is worth noticing that since gripper is symmetric, we only predict rotations ranging from 0 to 180 degree. The objective function for the network is:

$$\begin{aligned} L^R(R_{ij}, S_{ij}, W_{ij}) = \sum_{d=1}^{K} \Big(& \frac{1}{N_{cls}} \sum_{ij} L_{cls}^d(R_{ij}, R_{ij}^*) \\ & + \lambda_2 \frac{1}{N_{reg}} \sum_{ij} L_{reg}^d(S_{ij}, S_{ij}^*) \\ & + \lambda_3 \frac{1}{N_{reg}} \sum_{ij} L_{reg}^d(W_{ij}, W_{ij}^*) \Big), \end{aligned} \tag{6}$$

where R_{ij} denotes the binned rotation degrees, S_{ij}, W_{ij} and d denote the grasp confidence scores, gripper widths and approaching distance respectively. L^d means loss for the d^{th} binned distance. Here, for L_{cls} we use sigmoid cross entropy loss function for multi-class binary classification.

4.4. Tolerance Network

After previous steps, our end-to-end network can already predict accurate grasp poses. Beyond that, we further propose a representation called *grasp affinity fields*(GAFs) to improve the robustness of our grasp poses prediction. Since feasible grasp poses are infinite, humans tend to pick grasp poses that can tolerate larger errors. Inspired by this, our GAFs learns to predict the tolerance to perturbation for each grasp.

Given a ground truth grasp pose, we search its neighbors in the sphere space to see the farthest distance that the grasp is still robust with grasp score $s > 0.5$ and set it as the target for our GAFs. The loss function is written as:

$$L^F(A_{ij}) = \frac{1}{N_{reg}} \sum_{d=1}^{K} \sum_{ij} L_{reg}^d(T_{ij}, T_{ij}^*), \tag{7}$$

where T_{ij} denotes the maximum perturbation that the grasp pose can resist.

Object	s=1	s=0.5	s=0.1	**Object**	s=1	s=0.5	s=0.1
Banana	98%	67%	21%	Apple	97%	65%	16%
Peeler	95%	59%	9%	Dragon	96%	60%	9%
Mug	96%	62%	12%	Camel	93%	67%	23%
Scissors	89%	61%	5%	Power Drill	96%	61%	14%
Lion	98%	68%	16%	Black Mouse	98%	64%	13%

Table 2. Summary of real world success rate of grasping given different grasp score.

4.5. Training and Inference

During training, the whole network is updated in an end-to-end manner by minimizing the follow objective function:

$$L = L^A(\{c_i\}, \{s_{ij}\}) + \alpha L^R(R_{ij}, S_{ij}, W_{ij}) + \beta L^F(T_{ij}) \tag{8}$$

During inference, we refine our grasp poses by dividing them into 10 bins according to their grasp scores and resort the grasps in each bin according to the perturbation they can resist predicted by our tolerance network. We divide the predicted grasps into 10 bins because our labels have 10 different grasp scores. Experiments demonstrate that such refinement can improve the grasping quality effectively.

5. Experiments

In this section, we first conduct robotic experiments to demonstrate that our ground-truth annotations can align well with real-world grasping. Then we benchmark several representative methods on our dataset and compare them with our methods in a unified evaluation metric (Sec. 3.4). Finally, we conduct ablation studies to show the effectiveness of our network components.

5.1. Ground-Truth Evaluation

To evaluate the quality of our generated grasp poses, we set up a real robotic experiment. Since we need to project grasp poses to the camera frame using objects' 6D poses, we paste ArUco code on the objects and only label their 6D poses once to avoid tedious annotation process.

We pick 10 objects from our object set and execute grasp poses that has different scores. For each setting we randomly choose 100 grasp poses. For robot arm we adopt a Flexiv Rizon arm and for camera we use the Intel RealSense 435. Table 2 summarizes the success rate of grasping. We can see that for grasp poses with high score, the success rate can achieve 0.96 in average. Meanwhile, the success rate is pretty low for grasp poses with $s = 0.1$. It indicates that our generated grasp poses are well aligned with real world grasping.

5.2. Benchmarking Representative Methods

We benchmark different representative methods on our dataset and compare them with our method.

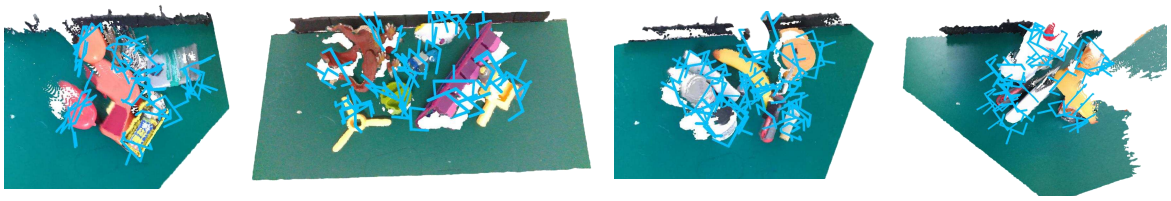


Figure 6. Qualitative results of our predicted grasp poses. Scenes are constructed using the RGB-D images taken by cameras. Grasps are represented by blue lines.

Methods	Seen			Unseen			Novel		
	AP	$AP_{0.8}$	$AP_{0.4}$	**AP**	$AP_{0.8}$	$AP_{0.4}$	**AP**	$AP_{0.8}$	$AP_{0.4}$
GG-CNN[27]	15.48/16.89	21.84/22.47	10.25/11.23	13.26/15.05	18.37/19.76	4.62/6.19	5.52/7.38	5.93/8.78	1.86/1.32
Chu *et al.* [7]	15.97/17.59	23.66/24.67	10.80/12.74	15.41/17.36	20.21/21.64	7.06/8.86	7.64/8.04	8.69/9.34	2.52/1.76
GPD [41]	22.87/24.38	28.53/30.16	12.84/13.46	21.33/23.18	27.83/28.64	9.64/11.32	8.24/9.58	8.89/10.14	2.67/3.16
Liang *et al.* [24]	25.96/27.59	33.01/34.21	15.37/17.83	22.68/24.38	29.15/30.84	10.76/12.83	9.23/10.66	9.89/11.24	2.74/3.21
Ours	**27.56/29.88**	**33.43/36.19**	**16.95/19.31**	**26.11/27.84**	**34.18/33.19**	**14.23/16.62**	**10.55/11.51**	**11.25/12.92**	**3.98/3.56**

Table 3. Evaluation for different methods. The table shows the results on data captured by RealSense/Kinect respectively.

For rectangle based method, we adopt two methods [27, 7] with open implementations. For point cloud proposal method, we adopt [41, 24]. We train these models according to their original implementations.

For our method, rotation angle is divided into 12 bins and approaching distance is divided into 4 bins with the value of 0.01, 0.02, 0.03, 0.04 meter. We set $M = 1024$ and $V = 300$. PointNet++ has four set abstraction layers with the radius of 0.04, 0.1, 0.2, 0.3 in meters and grouping size of 64, 32, 16 and 16, by which the point set is down-sampled to the size of 2048, 1024, 512 and 256 respectively. Then the points are up-sampled by two feature propagation layers to the size 1024 with 256-dim features. ApproachNet, OperationNet and ToleranceNet is composed of MLPs with the size of (256, 302, 302), (128, 128, 36) and (128, 64, 12) respectively. For the loss function, we set $\lambda_1, \lambda_2, \lambda_3, \alpha, \beta = 0.5, 1.0, 0.2, 0.5, 0.1$.

Our model is implemented with PyTorch and trained with Adam optimizer [18] on one Nvidia RTX 2080 GPU. During training, we randomly sample 20k points from each scene. The initial learning rate is 0.001 and the batch size is 4. The learning rate is decreased to 0.0001 after 60 epochs and then decreased to 0.00001 after 100 epochs.

We report the results of different methods in Tab. 3. As we can see, rectangle based method has a lower accuracy among all of the metrics. It denotes that previous rectangle based methods might be over-estimated. Our end-to-end network achieves the state-of-the-art result and outperforms previous methods by a large margin. We show some qualitative results of our predicted grasp poses in Fig. 6.

5.3. Ablation Studies

To evaluate the effectiveness of different components of our network, we conduct ablation studies on the seen test set of Kinect subset. First we evaluate whether different grasp pose representations would affect the results by directly regressing the direction of the approaching vector and degree of in-plane rotation. Then we evaluate the effectiveness of our ToleranceNet by removing it from our inference pipeline. Results are reported in Tab. 4. We can see that classification based learning scheme is indeed better than direct regression. Meanwhile, the drop of performance after removing the ToleranceNet demonstrates effectiveness of the grasp affinity fields.

Method	**AP**	$AP_{0.8}$	$AP_{0.4}$
Full	**29.88**	**36.19**	**19.31**
Replace classification with regression	23.74	33.28	12.15
Remove Tolerance Network	28.53	35.62	16.33

Table 4. Ablation studies of our network. See text for more details.

6. Conclusion

In this paper we built a large-scale dataset for cluttered scene object grasping. Our dataset is orders of magnitude larger than previous grasping datasets and diverse in objects, scenes and data sources. It consists of images taken by real world sensor and has rich and dense annotations. We demonstrated that our dataset align well with real world grasping. Meanwhile, we proposed an end-to-end grasp pose prediction network equipped with a novel representation of grasp affinity fields. Experiments showed the superiority of our method. Our code and dataset will be released.

Acknowledgment This work is supported in part by the National Key R&D Program of China, No. 2017YFA0700800, National Natural Science Foundation of China under Grants 61772332 and Shanghai Qi Zhi Institute

References

[1] Umar Asif, Jianbin Tang, and Stefan Harrer. Ensemblenet: Improving grasp detection using an ensemble of convolutional neural networks. In *BMVC*, page 10, 2018.

[2] Umar Asif, Jianbin Tang, and Stefan Harrer. Graspnet: An efficient convolutional neural network for real-time grasp detection for low-powered devices. In *IJCAI*, pages 4875–4882, 2018.

[3] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *arXiv preprint arXiv:1803.10091*, 2018.

[4] Samarth Brahmbhatt, Ankur Handa, James Hays, and Dieter Fox. Contactgrasp: Functional multi-finger grasp synthesis from contact. *arXiv preprint arXiv:1904.03754*, 2019.

[5] Shehan Caldera, Alexander Rassau, and Douglas Chai. Review of deep learning methods in robotic grasp detection. *Multimodal Technologies and Interaction*, 2(3):57, 2018.

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

[7] Fu-Jen Chu, Ruinian Xu, and Patricio A Vela. Real-world multiobject, multigrasp detection. *IEEE Robotics and Automation Letters*, 3(4):3355–3362, 2018.

[8] Xinke Deng, Yu Xiang, Arsalan Mousavian, Clemens Eppner, Timothy Bretl, and Dieter Fox. Self-supervised 6d object pose estimation for robot manipulation. *arXiv preprint arXiv:1909.10159*, 2019.

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

[10] Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 6(9):1305–1315, 1997.

[11] Kuan Fang, Yuke Zhu, Animesh Garg, Andrey Kurenkov, Viraj Mehta, Li Fei-Fei, and Silvio Savarese. Learning task-oriented grasping for tool manipulation from simulated self-supervision. *arXiv preprint arXiv:1806.09266*, 2018.

[12] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9224–9232, 2018.

[13] Di Guo, Fuchun Sun, Huaping Liu, Tao Kong, Bin Fang, and Ning Xi. A hybrid deep architecture for robotic grasp detection. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1609–1614. IEEE, 2017.

[14] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian conference on computer vision*, pages 548–562. Springer, 2012.

[15] Mingyang Jiang, Yiran Wu, Tianqi Zhao, Zelin Zhao, and Cewu Lu. Pointsift: A sift-like network module for 3d point cloud semantic segmentation. *arXiv preprint arXiv:1807.00652*, 2018.

[16] Yun Jiang, Stephen Moseson, and Ashutosh Saxena. Efficient grasping from rgbd images: Learning using a new rectangle representation. In *2011 IEEE International Conference on Robotics and Automation*, pages 3304–3311. IEEE, 2011.

[17] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1521–1529, 2017.

[18] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[19] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 863–872, 2017.

[20] Truc Le and Ye Duan. Pointgrid: A deep network for 3d shape understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9204–9214, 2018.

[21] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015.

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

[23] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems*, pages 828–838, 2018.

[24] Hongzhuo Liang, Xiaojian Ma, Shuang Li, Michael Görner, Song Tang, Bin Fang, Fuchun Sun, and Jianwei Zhang. Pointnetgpd: Detecting grasp configurations from point sets. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3629–3635. IEEE, 2019.

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision(ECCV)*, pages 740–755. Springer, 2014.

[26] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.

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

[28] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-dof graspnet: Variational grasp generation for object manipulation. *arXiv preprint arXiv:1905.10520*, 2019.

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

[30] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3406–3413. IEEE, 2016.

[31] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. *arXiv preprint arXiv:1904.09664*, 2019.

[32] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018.

[33] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[34] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[35] Yuzhe Qin, Rui Chen, Hao Zhu, Meng Song, Jing Xu, and Hao Su. S4g: Amodal single-view single-shot se (3) grasp detection in cluttered scenes. *arXiv preprint arXiv:1910.14218*, 2019.

[36] Joseph Redmon and Anelia Angelova. Real-time grasp detection using convolutional neural networks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1316–1322. IEEE, 2015.

[37] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. *arXiv preprint arXiv:1812.04244*, 2018.

[38] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2530–2539, 2018.

[39] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. *arXiv preprint arXiv:1703.09438*, 2017.

[40] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2018.

[41] Andreas ten Pas, Marcus Gualtieri, Kate Saenko, and Robert Platt. Grasp pose detection in point clouds. *The International Journal of Robotics Research*, 36(13-14):1455–1473, 2017.

[42] Jacob Varley, Jonathan Weisz, Jared Weiss, and Peter Allen. Generating multi-fingered robotic grasps via deep learning. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4415–4420. IEEE.

[43] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (TOG)*, 36(4):72, 2017.

[44] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018.

[45] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[46] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4606–4615, 2018.

[47] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018.

[48] Xinchen Yan, Jasmined Hsu, Mohammad Khansari, Yunfei Bai, Arkanath Pathak, Abhinav Gupta, James Davidson, and Honglak Lee. Learning 6-dof grasping interaction via deep geometry-aware 3d representations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.

[49] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018.

[50] Hanbo Zhang, Xuguang Lan, Site Bai, Xinwen Zhou, Zhiqiang Tian, and Nanning Zheng. Roi-based robotic grasp detection for object overlapping scenes. *arXiv preprint arXiv:1808.10313*, 2018.

[51] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.