End-to-End Learning to Grasp via Sampling From Object Point Clouds

End-to-End Learning to Grasp via Sampling from Object Point Clouds

Antonio Alliegro[1], Martin Rudorfer[2], Fabio Frattin[1], Aleš Leonardis[2] and Tatiana Tommasi[1]

***Abstract*—The ability to grasp objects is an essential skill that enables many robotic manipulation tasks. Recent works have studied point cloud-based methods for object grasping by starting from simulated datasets and have shown promising performance in real-world scenarios. Nevertheless, many of them still rely on ad-hoc geometric heuristics to generate grasp candidates, which fail to generalize to objects with significantly different shapes with respect to those observed during training. Several approaches exploit complex multi-stage learning strategies and local neighborhood feature extraction while ignoring semantic global information. Furthermore, they are inefficient in terms of number of training samples and time required for inference. In this paper, we propose an end-to-end learning solution to generate 6-DOF parallel-jaw grasps starting from the 3D partial view of the object. Our Learning to Grasp (L2G) method gathers information from the input point cloud through a new procedure that combines a differentiable sampling strategy to identify the visible contact points, with a feature encoder that leverages local and global cues. Overall, L2G is guided by a multi-task objective that generates a diverse set of grasps by optimizing contact point sampling, grasp regression, and grasp classification. With a thorough experimental analysis, we show the effectiveness of L2G as well as its robustness and generalization abilities.**

***Index Terms*—Deep Learning in Grasping and Manipulation; Deep Learning for Visual Perception; Grasping**

I. Introduction

GRASPING and manipulating unknown objects in unstructured, real-world environments is a long-standing challenge in robotics research. Ideally, we would like robots to be able to observe 3D objects and propose a variety of reliable grasps, out of which collision-free and kinematically feasible actions can be executed. However, there are many challenges in the whole grasping pipeline that need to be tackled, from perception to planning and control. Some works have simplified the task by focusing on 2D perception and planar grasps prediction: a camera observes the scene perpendicularly and the gripper pose is constrained to be parallel to the image plane [2], [3]. In this way, important geometric information may be disregarded inducing a limit in the grasp quality: the final effect is that the learned models hardly generalize beyond the scenario seen during training. When dealing with 3D perception, a very first bottleneck is due to imprecision and deficiency in sensing: the information acquired from the observed scene is usually noisy and affected by variations in the environmental conditions. Several works have proposed to overcome these issues by exploiting extra sources of geometric and physical information about the observed objects [4], [5], but these are not generally applicable for unknown shapes. Other approaches rely on specific gripper information which makes them less ready to generalize in real-world applications [6]. Besides being very sensitive to shape variations (dimension and aspect ratio), 3D-based existing approaches have high sample and time complexity: they need a large number of densely annotated data to be trained [7], [8] and a long prediction time when deployed [1]. This is mainly due to the use of handcrafted space quantization strategies and other heuristics that need to be progressively adjusted while training. From the implementation point of view, the approaches that take point clouds as input are often cumbersome. Although described as *end-to-end* strategies, they are multi-stage techniques composed of networks trained in cascade or simultaneously with separate losses: each network has its own learning objective with no gradient flow among each other [9], [8]. Furthermore, these works mostly concentrate on feature extraction from local point neighborhoods, with little consideration given to the global appearance of the point cloud.

With our work (see Fig. 1) we aim at pushing deep learning models for robot grasping one step further by overcoming at once the limitations described above. We introduce ***Learning to Grasp (L2G)***, an efficient end-to-end learning strategy to propose 6-DOF parallel-jaw grasps starting from a partial point cloud of an object. Differently from previous work, our approach does not exploit any geometric assumption or impose gripper constraints. **L2G builds on a differentiable sampling procedure. Specifically, it is guided by a multi-task optimization objective that identifies a set of diverse and reliable grasps by solving contact point sampling jointly with grasp regression and grasp classification.** We show how L2G largely improves over its competitors with an advantage that becomes ever more evident when reducing the amount of available training data. It also demonstrates remarkable generalization ability in challenging settings where training and test data present significant shape variations as well differences in the gripper. **Moreover, we go beyond standard backbone architectures discussing how a self-supervised pre-trained encoder that combines local and global cues can be easily plugged into the network and**

Manuscript received: February, 24, 2022; Revised June, 1, 2022; Accepted July, 6, 2022.

This paper was recommended for publication by Editor Markus Vincze upon evaluation of the Associate Editor and Reviewers' comments.

[1]A. Alliegro, F. Frattin and T. Tommasi are with the DAUIN Department at Politecnico di Torino. A. Alliegro and T. Tommasi are also affiliated with the Italian Institute of Technology, Italy. {antonio.alliegro, fabio.frattin,tatiana.tommasi}@polito.it

[2]M. Rudorfer and A. Leonardis are with the School of Computer Science at the University of Birmingham, UK. {m.rudorfer, a.leonardis}@bham.ac.uk

Digital Object Identifier (DOI): see top of this page.

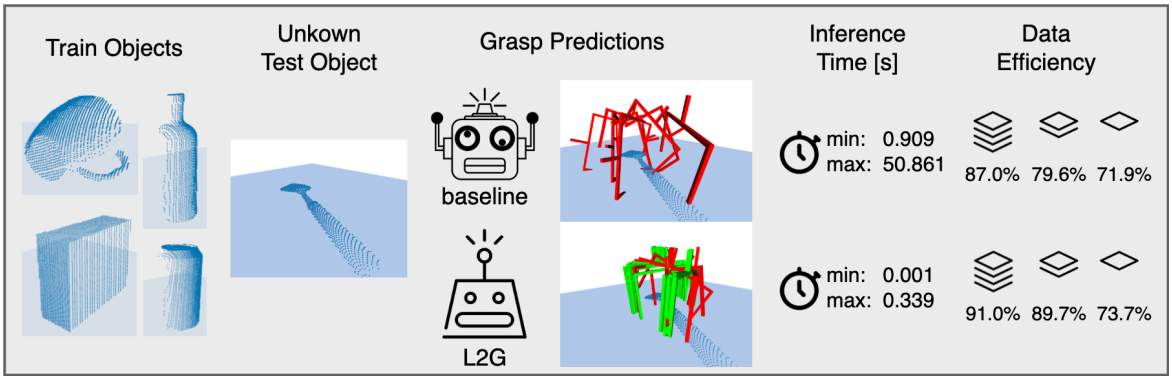


Fig. 1. Overview of the advantages of our L2G model over the GPNet [1] baseline. Trained with the same training dataset, L2G is able to predict a diverse set of reliable grasps even for objects from unseen categories and different shape distributions. The inference times are much shorter and drastically facilitate real-world application. Finally, L2G is showing higher grasp success rates even when trained with only half the amount of data.

provides a further advantage.

II. RELATED WORK

The task of grasping rigid objects with a 2-finger gripper consists in identifying the pose of the gripper in which the fingers close, starting from some representation of the object.

Data and representations. A large part of the grasping literature has focused on 2D and 2.5D (images with depth maps) data [3], [10], [11]. This setting simplifies the definition of the grasping problem, but at the same time limits its applicability with gripper forced to approach objects vertically. Dealing with images provides the possibility to exploit supervised Imagenet pre-trained networks as well as large scale self-supervised models reducing the need for data annotation [12], [13], [14]. On the other hand, point clouds allow better reasoning on the geometric properties of the objects and more freedom for the gripper pose [1], [15], [16], [17], [18], but 3D grasping is more challenging and needs densely annotated data. Here transfer learning from supervised or self-supervised pre-trained models [19], [20] is not standard practice, and existing works based on point clouds mainly exploit PointNet [21] and PointNet++ [22] representations which focus on local information, lacking the ability to properly capture the global object shape.

Several works have been dedicated to collecting and annotating grasping datasets by exploiting physical grasps in simulation engines [23], [24], [1]. Most recent publications have also proposed larger testbeds, but their simulation environments have not been released yet, which makes it difficult to consider the methods proposed in the same papers as benchmark references [17], [25], [6].

Grasping Methods. The earlier grasping approaches were based on handcrafted features [26], [27], while in recent years data-driven methods have gained popularity [28]. They can be categorized as model-based and model-free: the former relies on object-specific knowledge such as a 3D model or surface characteristics [4], [5], [29], while the latter assumes that no such explicit information is available. Model-free methods infer grasp poses purely based on the perceived information, and they can be conveniently applied to novel objects for which specific models are not available. Among them, Deep-Learning-based discriminative strategies evaluate a given set of grasp candidates [2], [25]. On the other hand, generative approaches regress the best grasp poses, however they usually lack grasp diversity [30], [31]. The most recent grasping methods combine the two aspects and incorporate both generative and discriminative components. In particular, [17], [18] generate grasp poses by means of a variational autoencoder trained on the distribution of successful grasps obtained from a simulation engine. A subsequent classifier and an iterative grasp refinement are used to rank and improve the candidates. In [15], [16], the authors use a tailored grasp representation and regress a grasp with a relative graspability score for each point of the input point cloud. A coarse grasp for every point is also predicted and then refined in [9]. A combination of sub-networks for point selection and refinement is used in [8]. The recent work [7] relies on dense grasp annotations and predicts the probability of successful grasping for each point which are then subselected via Farthest Point Sampling (FPS), while in [6] FPS is applied upfront to limit memory cost before running grasp prediction for each obtained point. GPNet [1] uses a grid-based heuristics to generate grasp proposals. A discrete set of regular 3D grid points is defined and proposals are obtained by pairwise combination of all input points (as contact points) with all grid points (as grasp centers). The learning process exploits an antipodal classifier to reduce the large number of proposals, a regression module to predict approach angles and offsets to the grasp centers, and a grasp classification module to estimate the success likelihood of the regressed grasps. Finally, the large number of predictions is reduced by Non Maximum Suppression (NMS).

Our L2G method fits in the context of Deep-Learning-based models for grasping from object point clouds. Given a partial observation of an unknown object, we *learn to sample* a set of suitable contact points and proceed to predict a 6-DOF grasp only for these points. Our strategy avoids unnecessary overhead by sub-optimal heuristics, naïve FPS, or expensive refinement stages and post-hoc NMS. Moreover, we show how 3D self-supervision improves data representation with a beneficial effect on grasping performance.

III. METHOD

The core contribution of our L2G model is in the procedure used to gather information from the input point cloud which

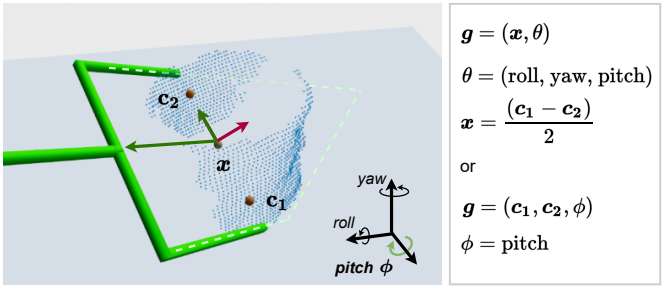


Fig. 2. The world coordinate system has its horizontal plane parallel to the ground and the vertical axis orthogonal to it. The grasp configuration is defined by the two contact points $(\boldsymbol{c}_1, \boldsymbol{c}_2)$ and the pitch angle ϕ which is the angle formed by the plane of the gripper (dashed lines) with the horizontal plane. We indicate with $\boldsymbol{x}$ the center between the contact points, while $\theta \in [-\pi, \pi]^3$ combines the information of yaw, roll, and pitch.

combines the sampling procedure adopted to identify the contact points, and the DeCo feature encoder [19]. Overall L2G is formalized as a multi-task deep architecture as shown in Fig. 3.

A. Problem Statement

Given a point cloud $\mathcal{P} = \{\boldsymbol{p}_i \in \mathbb{R}^3\}_{i=1}^N$ representing the visible surface of an object, we indicate a parallel-jaw grasp as $\boldsymbol{g} = \{\boldsymbol{x}, \theta\} \in SE(3)$. Here $\boldsymbol{x} \in \mathbb{R}^3$ locates the center of the two parallel jaws, and $\theta \in [-\pi, \pi]^3$ is the Euler angle describing the 3D orientation of the gripper, which can be also identified by its unit quaternion representation $\boldsymbol{u} = quat(\theta)$. A grasp can be alternatively defined by $\boldsymbol{g} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \phi\} \in \mathbb{R}^7$. Where $(\boldsymbol{c}_1, \boldsymbol{c}_2)$ are the two contact points on the object surface, which determine the grasp center $\boldsymbol{x}$ as well as the roll and yaw orientation of θ, while $\phi \in [0, \pi]$ is the remaining pitch orientation corresponding to the gripper approach angle[1] (see Fig. 2). A physical simulation engine provides us with a set of positive $\mathcal{G}^+$ (label $l = 1$) and negative $\mathcal{G}^-$ (label $l = 0$) ground truth grasps for $\mathcal{P}$. They all satisfy the antipodal constraint [32], but the negative ones fail to successfully lift the object. This may be due to collisions of the gripper with the object or ground, not making proper contact, or object slipping during lifting. We formalize the task of visual *learning to grasp* as learning the mapping from the object point cloud $\mathcal{P}$ to the set of $\boldsymbol{g}_{j=1}^M \in \mathcal{G}$ grasps that best matches $\mathcal{G}^+$. In order to avoid ambiguities due to symmetry, during evaluation all grasps are mapped into an unambiguous half-space by considering the contact point closer to the ground plane as $\boldsymbol{c}_1$, as done in [1].

B. Learning to Grasp

Our L2G architecture is optimized to predict all the components (contact points and pitch angle) of a grasp at once. It is composed of Feature Extractor, Contact Point Sampler, Grasp Regressor, and Grasp Classifier. Each of the modules is described in detail in the next paragraphs.

[1]Note that by restricting ϕ to $[0, \pi]$ instead of $[-\pi, \pi]$, we eliminate grasps with an approach vector in the lower hemisphere for which the gripper would collide with the ground plane.

Feature Extractor ($\mathbb{R}^{N\times 3} \rightarrow \mathbb{R}^{N\times F}$). The feature extractor learns an F-dimensional representation for each point of the observed point cloud $\mathcal{P}$. We consider two possible encoders: a standard PointNet++ [22] and DeCo [19]. The latter exploits graph convolutions and combines local information from denoising with global information from contrastive learning: the encoder is pre-trained with these two self-supervised tasks and has shown remarkable results when used for shape completion.

Contact Point Sampler ($\mathbb{R}^{N\times F} \rightarrow \mathbb{R}^{M\times F}$). The goal of the sampler is to identify the set of reliable contact points $\mathcal{Q} = \{\boldsymbol{q}_j \in \mathbb{R}^3\}_{j=1}^M$ with $M \leq N$, out of the visible point cloud $\mathcal{P}$, and collect for each of them the corresponding F-dimensional representation vector. The major issue with sampling is that it is a non-differentiable operation, but recent papers have proposed effective workarounds [33], [34]. We leverage these approaches to produce an $M \times 3$ matrix which can be interpreted as a set of M points. Specifically, we learn to produce M points that are close to the set of visible contact points $\mathcal{C}$ of $\mathcal{G}^+$ such that, for each of them, the projection on the object surface soft-matches to a single point of $\mathcal{P}$. The first goal is attained by optimizing the average nearest neighbor loss

$$\mathcal{L}_{nn}(X, Y) = \frac{1}{|X|} \sum_{\boldsymbol{x}\in X} \min_{\boldsymbol{y}\in Y} \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \ , \tag{1}$$

and the maximal nearest neighbor loss

$$\mathcal{L}_{mn}(X, Y) = \max_{\boldsymbol{x}\in X} \min_{\boldsymbol{y}\in Y} \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \ , \tag{2}$$

combined in the following closeness-coverage loss

$$\mathcal{L}_{cc}(\mathcal{Q}, \mathcal{C}) = \mathcal{L}_{nn}(\mathcal{Q}, \mathcal{C}) + \mathcal{L}_{nn}(\mathcal{C}, \mathcal{Q}) + \mathcal{L}_{mn}(\mathcal{Q}, \mathcal{C}) \ . \tag{3}$$

Here the first and last term forces the produced points to stay *close* to the grasp contact points both in average and in the worst case, while the second term ensures the full *coverage* of the grasping input set. Furthermore, for each point $\boldsymbol{q}$ we search the set $\boldsymbol{p}_{i=1}^k \in \mathcal{N}_{\mathcal{P}}(\boldsymbol{q})$ of its nearest neighbors from $\mathcal{P}$ in terms of Euclidean distance $d_i = \|\boldsymbol{q} - \boldsymbol{p}_i\|_2$. The k neighbors are used to evaluate the projection $\boldsymbol{r}$ of the produced point $\boldsymbol{q}$ on the object surface, formalized by the following linear combination

$$\boldsymbol{r} = \sum_{\boldsymbol{p}_i \in \mathcal{N}_{\mathcal{P}}(\boldsymbol{q})} \omega_i \boldsymbol{p}_i \ , \tag{4}$$

where

$$\omega_i = \frac{\exp^{-d_i^2/t^2}}{\sum_{\boldsymbol{p}_i \in \mathcal{N}_{\mathcal{P}}(\boldsymbol{q})} \exp^{-d_i^2/t^2}} \ . \tag{5}$$

These weights can be intended as a probability distribution over the points in $\mathcal{P}$, guided by the temperature parameter t. For high temperature values, the distribution becomes more and more uniform, while for low temperature values the distribution collapses to a Kronecker delta on the closest point. This last condition mimics the desired sampling and can be obtained by minimizing the projection loss:

$$\mathcal{L}_{proj} = t^2 \ . \tag{6}$$

Finally the sampling loss is $\mathcal{L}_{sample} = \alpha \mathcal{L}_{cc} + \mathcal{L}_{proj}$.

Grasp Regressor ($\mathbb{R}^{M\times F} \rightarrow \mathbb{R}^{M\times 4}$). Starting from the features of each selected point, we rely on the simplified

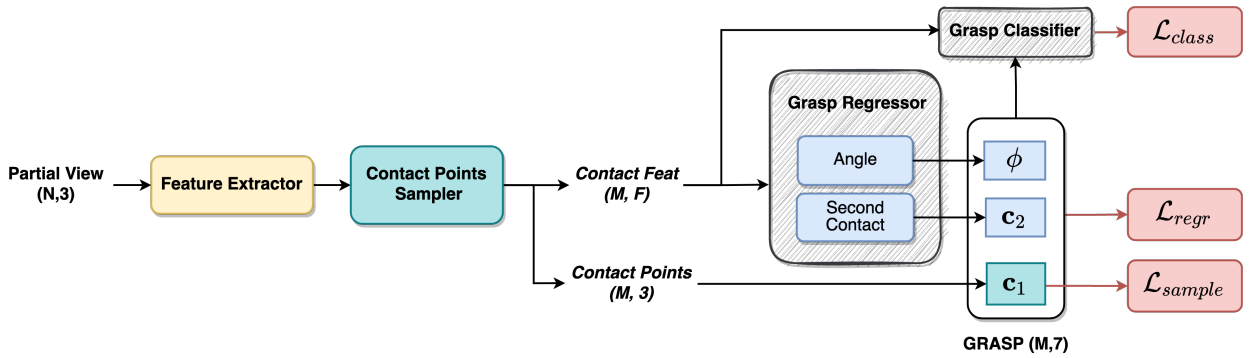


Fig. 3. Schematic overview of our Learning to Grasp (L2G) multi-task network. The *feature extractor* corresponds to our backbone encoder: we use both PointNet++ [22] and DeCo [19]. The *contact point sampler* produces points close to the ground truth contact points while soft-projecting them on the object surface. The *grasp regressor* and the *grasp classifier* are inherited from [1]: the former predicts the second contact point and the pitch angle for each grasp, while the latter scores the predicted grasps to identify the most reliable ones.

hypothesis that it corresponds to only one possible successful grasp, and the grasp regression module predicts both its second contact point ($\boldsymbol{c}_2 \in \mathbb{R}^3$) and the grasp pitch angle ($\phi \in \mathbb{R}^1$). The learning process is guided by a loss that measures the distance between each predicted grasp $\boldsymbol{g}_j = (\boldsymbol{c}_1, \boldsymbol{c}_2, \phi)_j$ and its closest ground truth grasp $\boldsymbol{g}_j^+ = (\boldsymbol{c}_1^+, \boldsymbol{c}_2^+, \phi^+)_j$. Specifically, the ground truth points $\boldsymbol{c}_1^+$ are sorted based on their distance to $\boldsymbol{c}_1$, and the closest one identifies the reference ground truth grasp. The distance between the grasps is measured in terms of the position of their centers and variation of the corresponding angles defined in terms of the quaternion representation $\boldsymbol{u}$:

$$\mathcal{L}_{regr} = \frac{1}{M} \sum_{j=1}^{M} \|\boldsymbol{x}_j - \boldsymbol{x}_j^+\|_2 + \lambda \arccos(|\langle \boldsymbol{u}_j, \boldsymbol{u}_j^+ \rangle|) \ , \quad (7)$$

where λ weighs the contributions of Euclidean and angular distances, as similarly done in [35].

Grasp Classifier ($\mathbb{R}^{M\times 4} + \mathbb{R}^{M\times F} \rightarrow \mathbb{R}^1$). The grasp classifier takes as input the information on the second contact point and angle ($\mathbb{R}^4$) as well as the features of the first contact point ($\mathbb{R}^F$) to finally score the grasp. Its purpose is to sort the predicted grasps and eliminate those that are unlikely to succeed. We use a simple binary cross-entropy loss where we indicate the predicted output with $s_j \in \mathbb{R}^1$ and the ground truth label with $l_j = [0, 1]$:

$$\mathcal{L}_{class} = -\frac{1}{M} \sum_{j=1}^{M} (l_j \log s_j + (1 - l_j) \log(1 - s_j)) \ . \quad (8)$$

All the loss contributions guide jointly the training process of our L2G: $\mathcal{L} = \mathcal{L}_{sample} + \mathcal{L}_{regr} + \mathcal{L}_{class}$.

C. Implementation Details

In the previous section, we provided a high-level intuition about the internal functioning of our approach by referring to a generic F-dimensional feature vector. Here we describe the architecture, the learned intermediate embeddings, and the hyperparameters of our model in more detail.

Our L2G employs the same feature extractor as GPNet [1]: a PointNet++ with four multiscale-grouping Set Abstraction (SA) layers followed by four Feature Propagation (FP) layers. For each observed partial object point cloud we obtain the global feature vector $F_s \in \mathbb{R}^{1024}$ by performing max-pooling on the feature map output of SA_4. The per-point features $F_p \in \mathbb{R}^{128}$ are obtained right after FP_4.

We dub our model *L2G+DeCo* when using as feature extractor the two-branch graph-convolutional backbone presented in [19], pre-trained via self-supervision on ShapeNetPart [36]. In this case, the global feature vector $F_s \in \mathbb{R}^{1024}$ is obtained from the global encoder branch, while the local encoder output is first concatenated with the global vector and then processed through a 128-dimensional convolutional layer. The output defines the per-point features $F_p \in \mathbb{R}^{128}$.

The contact point sampler takes as input the feature vector F_s to produce (soft sampling at training time) the $\boldsymbol{q}_{j=1}^M$ points of the $\mathcal{Q}$ set. The sampler is composed by four Fully Connected layers followed by the soft-projection module which has the same structure described in [34]. We consider $M = 500$ and use a small local context for the projection loss by setting the size of neighborhood $\mathcal{N}_\mathcal{P}(\boldsymbol{q})$ to $k = 10$. For each $\boldsymbol{q}$, we further define $\mathcal{N}_\mathcal{F}(\boldsymbol{q})$ by grouping the per-point feature vectors F_p of the $nn = 100$ nearest points on the input shape. The grasp regressor is fed with input $\mathcal{N}_\mathcal{F}(\boldsymbol{q})$ and predicts both the second contact point and the angle $(\boldsymbol{c}_2, \phi)$. The grasp classifier combines $\mathcal{N}_\mathcal{F}(\boldsymbol{q})$ with $(\boldsymbol{c}_2, \phi)$. Specifically, an MLP layer takes as input $(\boldsymbol{c}_2, \phi) \in \mathbb{R}^4$ to get a 128-dimensional feature vector which is aggregated by summation with $\mathcal{N}_\mathcal{F}(\boldsymbol{q})$ before entering a second MLP layer with 1-dimensional output followed by a sigmoid function. For a detailed analysis of the hyperparameters M and nn we refer to the next section and specifically Fig. 4.

In all our experiments we set the loss weighting parameters to $\alpha = 10$ and $\lambda = 0.1$. For each experiment, we take the average of three runs with different seeds and report the results from the last epoch. Our code is implemented in PyTorch 1.8 with CUDA 11.1. The models are trained on a single NVIDIA Tesla V100 16GB GPU. The code, pre-trained models, and data are available at https://github.com/antoalli/L2G.

IV. EXPERIMENTS

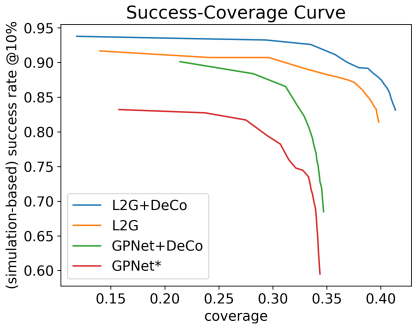
A. Datasets, Metrics and Baselines

For our experiments, we consider two datasets. *ShapeNetSem-8* [1] consists of 226 CAD models of 8 object categories (bowl, bottle, mug, cylinder, cuboid, tissue

TABLE I

LEFT: SIMULATION-BASED AND RULE-BASED EVALUATION RESULTS ON SHAPENETSEM-8. GPNET REFERS TO THE TOP RESULTS IN [1]. GPNET* INDICATES THE RESULTS OBTAINED BY RE-RUNNING THE AUTHORS' CODE. RIGHT: SIMULATION-BASED SUCCESS RATE IN RELATION TO THE COVERAGE FOR DIFFERENT SENSITIVITY SETTINGS.

Test Set from [1] - Simulation Based					Test Set from [1] - Rule Based							
Method	success rate @k%				success rate @k%				coverage rate @k%			
	10	30	50	100	10	30	50	100	10	30	50	100
GraspNet [17]	80.0	59.4	50.8	35.4	86.7	83.3	73.3	53.4	6.3	6.3	12.2	16.8
GPNet [1]	90.0	76.1	72.3	58.8	93.3	93.2	82.0	72.9	6.8	14.4	19.9	30.7
GPNet*	92.2	90.0	82.3	59.7	91.1	89.5	85.0	67.5	7.6	15.2	25.3	34.5
L2G	93.6	90.1	87.9	82.0	94.8	**95.9**	**95.3**	**95.1**	19.2	29.1	34.5	39.9
GPNet*+DeCo [19]	91.1	89.1	81.9	67.0	89.4	85.0	82.5	70.4	7.2	15.1	24.6	33.9
L2G+DeCo [19]	**94.6**	**93.5**	**91.4**	**82.9**	**95.2**	94.9	94.6	94.5	**20.6**	**29.2**	**35.5**	**41.8**
Extended Test Set - Simulation Based					Extended Test Set - Rule Based							
GPNet*	87.0	85.4	79.2	59.5	89.6	87.9	83.7	68.1	7.6	15.5	24.5	34.4
L2G	91.0	89.0	87.2	81.4	94.9	96.0	96.2	**95.8**	19.1	29.1	34.0	39.8
GPNet*+DeCo [19]	91.2	89.5	83.1	68.5	89.2	86.3	82.9	72.3	7.5	15.9	25.0	34.7
L2G+DeCo [19]	**93.0**	**92.1**	**90.7**	**83.2**	**96.4**	**96.5**	**96.4**	95.4	**20.3**	**31.0**	**36.5**	**41.5**



box, soda can and toy car) from ShapeNetSem [37]. Each object comes with ∼100k annotated grasps and associated grasp success or failure label obtained with the Pybullet physics simulator [38]. The dataset is split into 196 object instances for training and 30 object instances for testing. The test set in [1] was composed of only one view per object. To get more statistically meaningful results we also present an *Extended Test Set* of 5 different views per object, with each view serving as an independent test sample.

Using the Pybullet simulator, we also created our second dataset with 76 objects from YCB [39], dubbing it *YCB-76*. Each object is placed in various stable resting poses, totaling 259 distinct grasping scenarios, with point clouds generated from 10 arbitrary views. We consider the described YCB-76 as test set after having trained the grasping models on ShapeNetSem-8. For most of the objects, we only focused on the grasping success in simulation, but for a subset of them (*YCB-8*: cracker box, mug, tomato soup can, potted meat can, mustard bottle, flat screwdriver, large clamp, tennis ball) we also collected ∼100k ground truth grasps in the same fashion as in ShapeNetSem-8 to run a detailed rule-based analysis.

We perform our experimental evaluation by using the same metrics of [1]: simulation-based success rate as well as rule-based success rate and coverage. For the former, we use the same simulation environment used for grasp annotation. The rule-based metrics instead compare a predicted grasp $\boldsymbol{g}$ to the reference annotated grasps $\boldsymbol{g}^+ \in \mathcal{G}^+$ by means of Euclidean $\delta_{\boldsymbol{x}}(\boldsymbol{g}, \boldsymbol{g}^+) = ||\boldsymbol{x} - \boldsymbol{x}^+||_2$ and angular $\delta_\theta(\boldsymbol{g}, \boldsymbol{g}^+) = \arccos|\langle \boldsymbol{u}, \boldsymbol{u}^+ \rangle|$ distances. A prediction is considered successful if there exists at least one $\boldsymbol{g}^+$ with $\delta_{\boldsymbol{x}} \leq 25mm$ and $\delta_\theta \leq 30°$. Conversely, a grasp annotation $\boldsymbol{g}^+$ is considered covered, if there is a prediction $\boldsymbol{g} \in \mathcal{G}$ close by, using the same distance criterion. Hence, the coverage expresses what fraction of $\mathcal{G}^+$ is covered by the grasp predictions in $\mathcal{G}$. The rule-based success rate may be overly optimistic because it only takes into account proximity to successful ground truth grasps but not proximity to unsuccessful ones. For all three metrics, we consider the predictions ranked in the top k $= \{10, 30, 50, 100\}\%$, reporting the values as @k%.

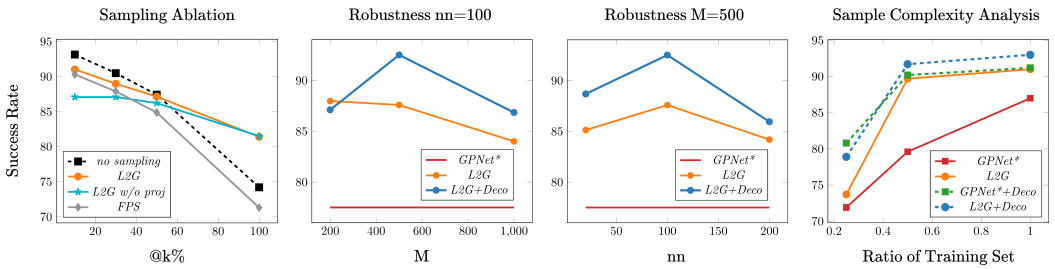
As reference baselines we consider GPNet [1] and GraspNet [17]. In particular the former is our best competitor: we highlight that this approach, besides relying on a heavy initial space quantization to choose the reference contact points, also requires an NMS post-processing stage to refine the predicted grasps. Moreover, to guarantee a fair comparison, we report both the results of GPNet from the original paper and our re-run of the authors' code indicated as GPNet*. With *GPNet+DeCo* we refer to the baseline where we applied the encoder from [19].

B. Experiments on ShapeNetSem-8

We start our analysis by running experiments on the training and test data of ShapeNetSem-8. The results reported in Table I (left) show that L2G and L2G+DeCo consistently outperform the corresponding GPNet and GPNet+DeCo baselines. On the smaller test set of [1], the effect of the DeCo encoder on top of L2G might be marginal. Instead, the advantage of DeCo becomes clear for both GPNet and L2G when focusing on the simulation-based results of the extended test set which should be considered the more reliable testbed. Notably, by increasing @k%, *i.e.* when taking into account predictions with lower confidence, the performances of the baselines drop significantly, whereas L2G maintains high success rates. By combining this information with the increased coverage rate we can conclude that L2G is effectively providing a wide variety of reliable grasp predictions and that DeCo further enhances this effect. The success-coverage plot in the right part of Table I confirms this behavior. It is comparable to a precision-recall curve where the simulation success rate resembles the precision and the coverage is the recall.

C. Sampling Ablation

We run an ablation analysis to assess the role of our contact point sampler. Specifically, we compare L2G to a model that uses the same encoder and grasp heads but lacks the contact points sampler, resulting in a grasp prediction for each point in the observed point cloud (*no sample*). We also consider replacing our sampler with a not-learned *FPS* selection process, as well as testing a variant of our L2G obtained by turning off $\mathcal{L}_{proj}$ (*L2G w/o proj*). As shown by the first plot of Fig. 4, L2G w/o proj performs worse than L2G when considering the top 10, 30 and 50% grasps, while on the entire set (100%) the results are the same. The success rate of



Time Complexity Analysis		
Method	Inference Time per shape (s)	
	min	max
GPNet*	0.909	50.861
L2G	**0.001**	**0.339**
GPNet*+DeCo	0.935	46.897
L2G+DeCo	0.016	0.365

Fig. 4. All the results refer to the simulation-based experiments on the extended test set of ShapeNetSem-8. First plot: ablation analysis to study the role of the contact point sampler. Second and third plot: robustness evaluation when varying M and nn. Fourth plot: sample complexity analysis executed by observing the performance of the grasping methods when changing the amount of training samples. Table: time complexity analysis.

FPS is always lower than L2G: such a naïve sampling process is the core of several 3D grasping approaches [6], [8] and it is significantly less effective than our learned sampling. Finally, the no-sampling strategy is the most expensive: we report it as a reference to show how our efficient L2G approaches its results and becomes equal or better when considering the top 50% predicted grasps and the entire set.

D. Robustness, Sample and Time Complexity Analysis

We evaluate the robustness of L2G to its two hyperparameters: the number M of sampled contact points which also corresponds to the total number of considered grasps, and the cardinality nn of the neighborhood $\mathcal{N}_{\mathcal{F}}(\boldsymbol{q})$ centered at each sampled contact $\boldsymbol{q}$. The second and third plots of Fig. 4 show that the performance of L2G has a mild dependence on M, with a decrease in success rate for very high values, while it is more robust to the choice of nn. The trend is similar also when using the DeCo encoder. Both L2G and L2G+DeCo maintain their advantages over the GPNet baseline.

To investigate the sample complexity, we reduced the amount of training data by a factor 0.25 and 0.50. The results in the fourth plot of Fig. 4 show that L2G outperforms GPNet even in low-shot scenarios. By comparing the accuracy of L2G at 0.25 (89.7) with that of GPNet at 1 (87.0) the low sample complexity of L2G appears even more evident. Similar conclusions can be drawn also comparing L2G+DeCo at 0.5 (91.7) with GPNet+DeCo at 1 (91.2). DeCo well combines with the handcrafted space quantization strategy of GPNet at 0.25 producing the best results.

Finally, the table within Fig. 4 allows to compare GPNet and L2G in terms of minimum and maximum inference time, showing the significant advantage of our approach.

E. Generalization on YCB

To assess the generalization abilities of L2G we employ the YCB dataset as test set. It has more object categories than ShapeNetSem-8 used for training. Additionally, objects in YCB vary in dimension and appear in different resting poses, leading to grasping scenarios with a wide range of difficulty levels. This is a challenging setting due to the need for overcoming both the semantic and the appearance domain shift. The simulation-based and rule-based results on YCB-8 are in the left part of Table II, while the right part presents the simulation-based results on YCB-76.

On YCB-8, L2G outperforms the GPNet baseline by a large margin in terms of both success rates and coverage. L2G+DeCo shows a further advantage over L2G which is particularly evident in the simulation based results. Interestingly, the DeCo encoder provides a significant improvement also to GPNet, with GPNet+DeCo reaching similar or even better success rate in simulation than L2G. Still, the results of GPNet+DeCo remain lower than L2G+DeCo in most of the settings. On the large YCB-76, L2G shows top simulation-based results, demonstrating its generalization abilities. In this case the advantage obtained by using DeCo appears minimal over the already high success rate of L2G. On the other hand, the results of GPNet+DeCo are significantly higher than those of GPNet, despite remaining lower than L2G. These findings suggest that the choice of the feature encoder is important for improving generalization on weaker models. In Fig. 5 we also present a qualitative analysis to compare the grasp predictions of GPNet and L2G in simulation. The sponge can be considered an adversarial object: it is completely flat and its side length is close to the gripper opening width, hence imposing a high risk of the gripper colliding with either the ground or the object and neither method accomplishes to predict successful grasps. On the other hand, the peach can be grasped without failure, although there weren't spherical objects in the training set. Especially with increasing @k%, GPNet tends to predict many spurious, unreasonable grasps, which is not the case for L2G. For elongated objects like the spoon, the grid-based approach of GPNet fails entirely. L2G, in contrast, is more robust to unknown shapes since it does not rely on any geometric heuristics for the grasp proposal.

F. Robot Experiments

To evaluate the performance in real-world scenarios, we conducted experiments with a Franka Emika Panda robot and an Intel RealSense D415 sensor (see Fig. 6). Based on the highest-ranked grasp prediction obtained from the models (trained purely on synthetic data of ShapeNetSem-8), we plan a trajectory using MoveIt [40]. If no feasible plan can be found, we swap the contact points and plan for this symmetric grasp. If this still does not give a feasible trajectory, we resort to the second or at most the third grasp prediction. The trial counts as successful only if the object is continuously in contact with both fingertips from grasp execution until release.

We performed five trials for each object in two sets: one with 28 custom items from the same categories as in ShapeNetSem-8, and one with 20 YCB objects for better experimental

TABLE II
RESULTS OBTAINED WHEN TESTING ON YCB-8 AND YCB-76. FOR THE FIRST, WE COLLECTED GRASP ANNOTATIONS TO ALSO ASSESS THE RULE-BASED PERFORMANCE, WHILE FOR THE SECOND WE SHOW ONLY THE SIMULATION-BASED RESULTS.

Method	YCB-8												YCB-76			
	Simulation Based				Rule Based								Simulation Based			
	success rate @k%				success rate @k%				coverage rate @k%				success rate @k%			
	10	30	50	100	10	30	50	100	10	30	50	100	10	30	50	100
GPNet*	27.9	28.9	30.1	21.9	54.0	53.7	52.2	37.7	3.0	8.0	14.3	18.6	28.9	29.2	27.2	20.8
L2G	44.6	42.8	42.1	39.0	75.8	76.6	76.3	73.0	16.5	23.6	27.2	33.3	**45.0**	**44.6**	43.8	41.2
GPNet*+DeCo [19]	**54.9**	48.7	42.6	32.7	**80.0**	72.7	63.5	51.0	7.5	15.8	20.5	26.2	40.3	39.2	38.0	34.1
L2G+DeCo [19]	52.5	**53.4**	**51.6**	**46.3**	77.7	**78.7**	**78.4**	**76.5**	**17.3**	**23.6**	**27.5**	**33.8**	43.6	44.0	**43.9**	**42.2**

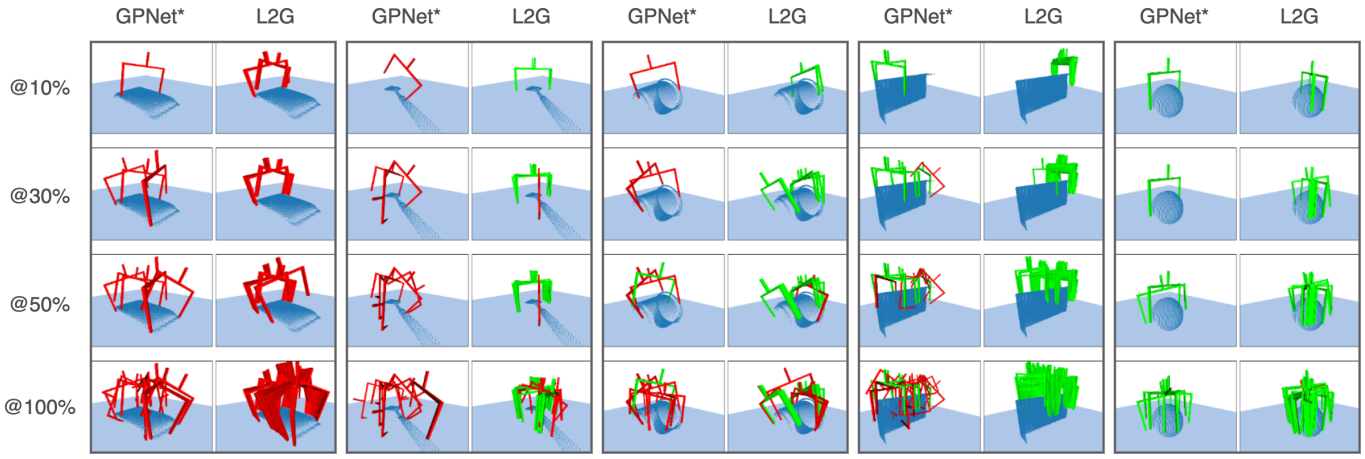


Fig. 5. Visualization of the predicted grasps of GPNet* and L2G for five different objects from YCB-76 (from left to right: sponge, spoon, cup, sugar box, peach). Based on the outcome of the simulation, we color-coded successful grasps in green and unsuccessful ones in red. From top to bottom, we increase the parameter k, i.e. the top row contains only the 10% highest-ranked grasp predictions whereas the bottom row contains all grasp predictions.

Fig. 6. Left: Our real experiments setup with Franka Panda robot and Realsense D415. Center: Custom set with 28 objects from similar categories as in ShapeNetSem-8 (from left to right and top to bottom: 6 boxes, 5 bottles, 2 soda cans, 5 cylinders, 3 toy cars, 3 mugs, 4 bowls). Right: Selection of 20 YCB objects including shapes from unseen categories.

TABLE III
REAL-WORLD ROBOT EXPERIMENTS: FRACTION OF SUCCESSFUL TRIALS OUT OF FIVE PERFORMED ON EACH INSTANCE AND AVERAGED PER CATEGORY. NUMBER OF INSTANCES PER CATEGORY IN PARENTHESIS.

28 custom items		
Category	GPNet*	L2G
box (6)	0.47	**0.73**
soda can (2)	0.80	**1.00**
cylinder (5)	0.52	**0.76**
bottle (5)	0.25	**0.48**
mug (3)	0.27	**0.40**
bowl (4)	**0.15**	0.10
toy car (3)	**0.33**	0.27
average	0.40	**0.53**

20 YCB objects		
Category	GPNet*	L2G
box (7)	0.31	**0.71**
mug (1)	**0.20**	**0.20**
bowl (1)	0.00	0.00
cylinder (5)	**0.72**	0.64
sphere (3)	**1.00**	**1.00**
bottle (3)	0.07	**0.27**
average	0.38	**0.47**

reproducibility (see Fig. 6). The results are displayed in Table III and indicate that the grasping performance varies strongly in relation to the object category. The bowls, which have been grasped with least success, required different modes of grasping: they can be grasped around the circumference only if the diameter is smaller than the gripper opening width, else they must be grasped along the rim. Both GPNet and L2G did not cope well with this mode switch. On the other hand, sphere-like objects and soda cans could be grasped by L2G without failure. Across categories, we observed that both grasping approaches are sensitive to object size. In ShapeNetSem-8, all objects are scaled to have their smallest dimension >60mm and their largest <150mm. Based on this criterion we can separate the 28 custom items into two sub-groups: one with the 10 objects which fit the size constraints, the other with the remaining 18 which do not. It is genuinely hard to find real objects from the categories bottle, toy car, bowl, and cylinder with these dimensions and almost all the instances of those categories are in the second group. As could be expected the results on the 10 object group (GPNet: 0.44, L2G: 0.62) are higher than those on the 18 object group (GPNet: 0.36, L2G: 0.49), but the improvement of L2G over GPNet remains confirmed.

A further challenge originates from the simulation to reality gap to which also the gripper type contributes. The gripper used in the simulation environment was a Robotiq-2F85 gripper, whereas we used a Panda gripper with shorter finger length and slightly smaller opening width. This is crucial when grasping (flat) objects close to the ground, like toy cars. Overall, L2G outperforms GPNet by a significant margin and is also ahead in most category-specific comparisons, with the biggest margins for boxes and bottles.

V. CONCLUSIONS

In this paper we introduced L2G, our end-to-end method for 6-DOF grasping from partial object point clouds which leverages a differentiable sampling strategy to identify the visible contact points, and a feature encoder which combines local and global cues. Overall, L2G is guided by a multi-task objective to produce a diverse set of grasps by optimizing contact point sampling, grasp regression, and grasp classification. In our experimental analysis we thoroughly compared L2G to the main competitor GPNet [1] and could demonstrate its advantages: L2G predicts a larger, more diverse set of reliable grasps. Furthermore, it better generalizes to unseen objects with significant shape variations. Using the pre-trained feature encoder DeCo [19] instead of a standard PointNet++ significantly boosts generalization performance. Here we considered single object scenes to maintain the main focus on the robustness and effectiveness of the proposed approach. Still, by its design, L2G can be easily adapted for scenes containing multiple objects. The point sampling procedure could also incorporate prior knowledge about the downstream manipulation task when available. We plan to investigate both these directions in future work.

ACKNOWLEDGMENT

This work was partially supported by the CHIST-ERA BURG project under EPSRC grant no. EP/S032487/1. We also acknowledge the CINECA award IsC87_SS-GRASP under the ISCRA initiative, for the availability of high performance computing resources and support.

REFERENCES

[1] C. Wu, J. Chen, Q. Cao, J. Zhang, Y. Tai, L. Sun, and K. Jia, "Grasp proposal networks: An end-to-end solution for visual learning of robotic grasps," in *NeurIPS*, 2020.

[2] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *RSS*, 2017.

[3] D. Morrison, P. Corke, and J. Leitner, "Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach," in *RSS*, 2018.

[4] P. Brook, M. Ciocarlie, and K. Hsiao, "Collaborative grasp planning with multiple object representations," in *ICRA*, 2011.

[5] C. Borst, M. Fischer, and G. Hirzinger, "Grasp planning: how to choose a suitable task wrench space," in *ICRA*, 2004.

[6] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox, "Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes," in *ICRA*, 2021.

[7] C. Wang, H.-S. Fang, M. Gou, H. Fang, J. Gao, and C. Lu, "Graspness discovery in clutters for fast and accurate grasp detection," in *ICCV*, 2021.

[8] 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 *ICRA*, 2021.

[9] W. Wei, Y. Luo, F. Li, G. Xu, J. Zhong, W. Li, and P. Wang, "Gpr: Grasp pose refinement network for cluttered scenes," in *ICRA*, 2021.

[10] H. Zhang, X. Lan, S. Bai, X. Zhou, Z. Tian, and N. Zheng, "Roi-based robotic grasp detection for object overlapping scenes," in *IROS*, 2019.

[11] X. Zhou, X. Lan, H. Zhang, Z. Tian, Y. Zhang, and N. Zheng, "Fully convolutional grasp detection network with oriented anchor box," in *IROS*, 2018.

[12] X. Zhu, D. Wang, O. Biza, G. Su, R. Walters, and R. Platt, "Sample efficient grasp learning using equivariant models," *RSS*, 2022.

[13] B. Huang, S. D. Han, A. Boularias, and J. Yu, "Dipn: Deep interaction prediction network with application to clutter removal," in *ICRA*, 2021.

[14] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," in *IROS*, 2018.

[15] 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 *CoRL*, 2020.

[16] P. Ni, W. Zhang, X. Zhu, and Q. Cao, "Pointnet++ grasping: Learning an end-to-end spatial grasp generation algorithm from sparse point clouds," in *ICRA*, 2020.

[17] A. Mousavian, C. Eppner, and D. Fox, "6-dof graspnet: Variational grasp generation for object manipulation," in *ICCV*, 2019.

[18] A. Murali, A. Mousavian, C. Eppner, C. Paxton, and D. Fox, "6-dof grasping for target-driven object manipulation in clutter," in *ICRA*, 2020.

[19] A. Alliegro, D. Valsesia, G. Fracastoro, E. Magli, and T. Tommasi, "Denoise and contrast for category agnostic shape completion," in *CVPR*, 2021.

[20] H. Wang, Q. Liu, X. Yue, J. Lasenby, and M. J. Kusner, "Unsupervised point cloud pre-training via occlusion completion," in *ICCV*, 2021.

[21] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *CVPR*, 2017.

[22] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *NeurIPS*, 2017.

[23] X. Yan, J. Hsu, M. Khansari, Y. Bai, A. Pathak, A. Gupta, J. Davidson, and H. Lee, "Learning 6-dof grasping interaction via deep geometry-aware 3d representations," in *ICRA*, 2018.

[24] A. Depierre, E. Dellandréa, and L. Chen, "Jacquard: A large scale dataset for robotic grasp detection," in *IROS*, 2018.

[25] H. Liang, X. Ma, S. Li, M. Görner, S. Tang, B. Fang, F. Sun, and J. Zhang, "PointNetGPD: Detecting grasp configurations from point sets," in *ICRA*, 2019.

[26] A. Saxena, L. L. S. Wong, and A. Y. Ng, "Learning grasp strategies with partial shape information," in *AAAI*, 2008.

[27] A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, T. Asfour, and S. Schaal, "Template-based learning of grasp selection," in *ICRA*, 2012.

[28] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber, "A survey on learning-based robotic grasping," *Curr Robot Rep*, vol. 1, p. 239–249, 2020.

[29] 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 *ICRA*, 2016.

[30] P. Schmidt, N. Vahrenkamp, M. Wächter, and T. Asfour, "Grasping of unknown objects using deep convolutional neural networks based on depth images," in *ICRA*, 2018.

[31] C. Choi, W. Schwarting, J. DelPreto, and D. Rus, "Learning object grasping for soft robot hands," *IEEE RAL*, vol. 3, no. 3, pp. 2370–2377, 2018.

[32] V.-D. Nguyen, "Constructing force- closure grasps," *The International Journal of Robotics Research*, vol. 7, no. 3, pp. 3–16, Jun 1988.

[33] O. Dovrat, I. Lang, and S. Avidan, "Learning to sample," in *CVPR*, 2019.

[34] I. Lang, A. Manor, and S. Avidan, "SampleNet: Differentiable Point Cloud Sampling," in *CVPR*, 2020.

[35] C. Eppner, A. Mousavian, and D. Fox, "A billion ways to grasps - an evaluation of grasp sampling schemes on a dense, physics-based grasp data set," in *ISRR*, 2019.

[36] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas, "A scalable active framework for region annotation in 3d shape collections," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, Nov. 2016.

[37] M. Savva, A. X. Chang, and P. Hanrahan, "Semantically-enriched 3d models for common-sense knowledge," in *CVPR-W*, 2015.

[38] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

[39] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the yale-CMU-berkeley object and model set," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 36–52, Sep 2015.

[40] I. A. Sucan and S. Chitta, "Moveit!" https://moveit.ros.org/, 2013.