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

Antonio Alliegro¹, Martin Rudorfer², Fabio Frattin¹, Aleš Leonardis² and Tatiana Tommasi¹

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 strongly 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. Moreover, these methods are generally inefficient with respect to the number of training samples and the time needed during deployment. In this paper, we propose an end-to-end learning solution to generate 6-DOF parallel-jaw grasps starting from the partial view of the object. Our Learning to Grasp (L2G) method takes as input object point clouds and is guided by a principled multi-task optimization objective that generates a diverse set of grasps combining contact point sampling, grasp regression, and grasp evaluation. With a thorough experimental analysis, we show the effectiveness of the proposed method as well as its robustness and generalization abilities.

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 from perception to planning and control. 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 [2], [3], but these are not generally applicable for unknown shapes. Alternative strategies consist in simplifying the task, e.g. by focusing only on planar grasps with a camera observing the scene perpendicularly, and constraining the gripper pose to be parallel to the image plane [4], [5]. The effect of these choices is that the obtained models do not generalize beyond the scenario seen during training. Besides being very sensitive to shape variations (dimension and aspect ratio), existing approaches have also high sample and time complexity: they need a large number of annotated data to be trained and a long prediction time when deployed. 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 3D point clouds are often cumbersome: they are trained through multiple-stage learning strategies and exploit the combination of basic architecture modules that mainly focus on local point neighborhoods.

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 present Learning to Grasp (L2G), an efficient end-to-end learning strategy to generate 6-DOF parallel-jaw grasps starting from a partial point cloud of an object. Our approach does not exploit any geometric assumption, it is instead guided by a principled multi-task optimization objective that generates a diverse set of grasps by combining contact point sampling, grasp regression, and grasp evaluation. 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. Moreover, we go beyond the use of standard backbone architectures discussing how a self-supervised pre-trained encoder that combines local and global information can be easily plugged into the network and provides a further advantage. We can summarize our main contributions as follows:

- We introduce L2G, a new end-to-end method for 6-DOF grasping on object point clouds that jointly learns how to select contact points and generates reliable grasp poses for them.
- We present a thorough experimental evaluation showing that the proposed approach outperforms its competitors. The evaluation is done in terms of success rate and coverage and we also investigate generalization on test data with significant shape variations compared to the training set.
- A study on the robustness of our method shows that it is barely sensitive to its hyperparameters. We also assess the time complexity of L2G empirically demonstrating its low inference time.
- We investigate the role of the feature encoder, highlighting its importance for the grasping task particularly when the number of training samples is limited.
- A set of robotic experiments show how L2G can be effectively deployed in a real-world setting, indicating that it is able to exploit knowledge learned from synthetic scenarios without the need for further fine-tuning.

---

¹ This work was partially supported by the CHIST-ERA BURG project under EPSRC grant no. EP/K033487/1.

² 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

³ 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
II. RELATED WORK

The task of grasping rigid objects with a 2-finger gripper is typically reduced to 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 [5], [6], [7]. Using this setting allows to simplify the definition of the grasping problem, but at the same time limits its applicability. Point clouds allow better reasoning on the geometric properties of the objects and provide more freedom for the gripper pose [1], [8], [9], [10], [11]. The current robot grasping works based on point clouds mainly exploit the basic PointNet [12] and PointNet++ [13] representations which focus on local information and are not able to properly capture the global object shape. More effective representations based on graph convolutions have been rarely considered in relation to grasping [14], [15], and no previous work has analyzed self-supervised learned embeddings [16], [17] for the same task.

Several works have been dedicated to collecting and annotating grasping datasets by exploiting physical grasps in simulation engines [18], [19], [1]. Most recent publications have also proposed larger testbeds, but their simulation environments have not been released yet, which also makes it difficult to consider the methods proposed in the same papers as benchmark reference [10], [20].

Grasping Methods. The earlier grasping approaches were based on handcrafted features [21], [22], while in recent years data-driven methods have gained popularity [23]. They can be categorized as model-based and model-free: the former methods rely on object-specific knowledge such as a 3D model or surface characteristics [2], [3], [24], 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 [4], [20]. On the other hand, generative approaches regress the best grasp poses, however they usually lack in grasp diversity [25], [26]. The most recent approaches combine the two aspects and incorporate both generative and discriminative components. In particular, [10], [11] generate grasp poses by means of a variational autoencoder trained to encode 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 [8], [9], the authors use an approach-based grasp representation and regress for each point of the input point cloud a grasp and the relative graspability score. The number of proposals hence depends on the size of the input point cloud.

GPNet [1] uses a grid-based heuristics to generate initial 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 proposed learning method follows a four-stage process that 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.

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, thus avoiding unnecessary overhead by sub-optimal heuristics or expensive refinement stages.

III. METHOD

A. Problem Statement

Given a point cloud \( P = \{ p_i \in \mathbb{R}^3 \}_{i=1}^N \) representing the visible surface of an object from a single-view depth image, we indicate a parallel-jaw grasp as \( g = \{ x, \theta \} \in SE(3) \). Here \( x \in \mathbb{R}^3 \) 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
is defined by the two contact points \( u \) and \( v \). A grasp can be alternatively defined by \( g = \{ c_1, c_2, \phi \} \in \mathbb{R}^7 \). Where \( (c_1, c_2) \) are the two contact points on the object surface, which determine the grasp center \( x \) and yaw orientation of \( \theta \), while \( \phi \in [0, \pi] \) is the remaining pitch orientation corresponding to the gripper approach angle.

A physical simulation engine provides us with a set of positive \( G^+ \) (label \( l = 1 \)) and negative \( G^- \) (label \( l = 0 \)) ground truth grasps for \( \mathcal{P} \). They all satisfy the antipodal constraint [27], 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 \( g^M \) grasps that best matches \( G^+ \).

B. Learning to Grasp

Our L2G multi-task deep architecture is optimized to predict all the components (contact points and pitch angle) of a grasp at once. The architecture is composed of Feature Extractor, Contact Point Sampler, Grasp Regressor, and Grasp Classifier as shown in Fig. 3. Each module is described in detail in the next paragraphs.

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++ [13] and DeCo [16]. The latter exploits graph convolutions and combines local information from denosing 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 \( \mathcal{Q} = \{ q_j \in \mathbb{R}^3 \}_{j=1}^M \) 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 [28], [29]. We leverage these approaches to generate \( M \) points that are close to the \( (c_1^+, c_2^+) \in C \) contact points of \( G^+ \) 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

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

and the maximal nearest neighbor loss

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

combined in the following closeness-coverage loss

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

Here the first and last term forces the generated 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 \( q \) we search the set \( p^k \in \mathcal{P}(q) \) of its nearest neighbors from \( \mathcal{P} \) in terms of Euclidean distance \( d_i = \| q - p_i \|_2 \). The \( k \) neighbors are used to evaluate the projection \( r \) of the generated point \( q \) on the object surface, formalized by the following linear combination

\[
r = \sum_{p_i \in \mathcal{N}_k(q)} \omega_i p_i , \tag{4}
\]

where

\[
\omega_i = \frac{\exp\left(-d_i^2/t^2\right)}{\sum_{p_j \in \mathcal{N}_k(q)} \exp\left(-d_j^2/t^2\right)} . \tag{5}
\]

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 \( \mathcal{L}_{sampling} = \alpha \mathcal{L}_{cc} + \mathcal{L}_{proj} \). 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 hypothesis that it corresponds to only one possible successful grasp, and the grasp regression module predicts both its second contact point \( (c_2 \in \mathbb{R}^3 \) 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 \( g_j = (c_1, c_2, \phi) \) and its closest ground truth grasp \( g_j^+ = (c_1^+, c_2^+, \phi^+) \). Specifically, the ground truth contact points \( c_1^+ \) are sorted based on their distance to \( c_1 \), and the closest one also 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 $\mathbf{u}$:

$$\mathcal{L}_{\text{grasping}} = \frac{1}{M} \sum_{j=1}^{M} \left( \| \mathbf{x}_j - \mathbf{x}^*_j \|_2 + \lambda \arccos(\langle \mathbf{u}_j, \mathbf{u}^*_j \rangle) \right),$$

where $\lambda$ weighs the contributions of Euclidean and angular distances, as similarly done in [30].

**Grasp Classifier** ($\mathbb{R}^{M \times 4} \times \mathbb{R}^{M \times F} \rightarrow \mathbb{R}^1$). 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 prune away the predicted grasps that are unlikely to be unsuccessful. Furthermore, having a grasp quality rank is crucial at deployment time since in real-world applications the number of grasping attempts is typically limited. Thus we should be able to identify and execute the most reliable ones. We use a simple binary cross-entropy loss where we indicate the predicted output with $s_j \in \{0, 1\}$ and the ground truth label with $l_j = [0, 1]$:

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

All the loss contributions guide jointly the training process of our L2G model: $\mathcal{L} = \mathcal{L}_{\text{sampling}} + \mathcal{L}_{\text{regression}} + \mathcal{L}_{\text{classifier}}$.

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 basic L2G method 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 shape (i.e. 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 SA1.

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 [16], pre-trained via self-supervision on ShapeNet-Part [31]. In this case, the global feature vector $F_s \in \mathbb{R}^{1024}$ is directly 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 obtained 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 generate (soft sampling at training time) the $q^*_M$ points of the $\mathcal{Q}$ set. We consider $M = 500$ and use a small local context for the projection loss by setting the size of neighborhood $\mathcal{N}_F(q)$ to $k = 10$. For each $q$, we further define $\mathcal{N}_F(q)$ 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 $\mathcal{N}_F(q)$ and predicts both the second contact point and the angle ($c_2, \phi$). The grasp classifier combines $\mathcal{N}_F(q)$ with ($c_2, \phi$). Specifically, an MLP layer takes as input ($c_2, \phi$) to get a 128-dimensional feature vector which is aggregated by summation with $\mathcal{N}_F(q)$ before entering a second MLP layer with 1-dimensional output followed by a sigmoid function. For a detailed ablation of the hyperparameters $M$ and $nn$ we refer to the next section and specifically Fig. 5.

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. Our models are trained on a single NVIDIA Tesla V100 16GB GPU, inference and real-world experiments are performed on an NVIDIA 2080 GPU. The code, pre-trained models, and data will be available at https://github.com/antoalli/L2G.

IV. EXPERIMENTS

A. Datasets, Metrics and Baselines

For our experiments, we consider two datasets. The first is ShapeNetSem-8: it is the one originally presented in [1] and consists of 226 CAD models of 8 object categories (bowl, bottle, mug, cylinder, cuboid, tissue box, soda can and toy.}

![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++ [13] and DeCo [16]. The contact point sampler generates points close to the ground truth contact points while soft-projecting them on the object surface. The grasp regressor predicts the second contact point and the pitch angle for each grasp, while the grasp classifier scores the predicted grasps to identify the most reliable ones.](image-url)
car) from ShapeNetSem [32]. Each object comes with ~100k annotated grasps and associated grasp success or failure label obtained with the Pybullet physics simulator[33]: about 24% of them have a positive annotation corresponding to a grasping success. The point clouds are obtained from RGBD images collected under 1k arbitrary views. The dataset is split into 196 object instances for training and 30 object instances for testing. Using the Pybullet simulator, we also created our second dataset with 76 objects from YCB [34], 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 a testbed 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 as in [1]: simulation-based success rate as well as rule-based success rate and coverage. For the former, we use the same simulation environment as for the creation of the grasp annotations. The rule-based metrics instead compare a predicted grasp \( g \) to the reference annotated grasps \( g^* \in G^+ \) by means of Euclidean \( \delta_g (\mathbf{g}, g^+) = \| \mathbf{x} - \mathbf{x}^+ \|_2 \) and angular \( \delta_\theta (\mathbf{g}, g^+) = \arccos (\langle \mathbf{u}, \mathbf{u}^+ \rangle) \) distances. In order to avoid ambiguities due to symmetry, all grasps are converted to a canonical orientation. A prediction is considered successful if there exists at least one \( g^* \) with \( \delta_g \leq 25\text{mm} \) and \( \delta_\theta \leq 30^\circ \). Conversely, a grasp annotation \( g^* \) is considered covered, if there is a prediction \( g \in G \) close by, using the same distance criterion. Hence, the coverage expresses what fraction of \( G^+ \) is covered by the grasp predictions in \( G \). 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 \( k = \{10, 30, 50, 100\} \) for computation and report the values as \( @k\% \).

As reference baselines we consider GPNet [1] and GraspNet [10]. 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 benefits from the non-maximum suppression 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 [16].

### B. Experiments on ShapeNetSem-8

We started our analysis by running experiments on the training and test data of ShapeNetSem-8. The simulation-based and rule-based results are collected respectively in the left and right parts of Table 1. The reported values are obtained as average on three runs (with different random seeds) and we consider the performance in the case of one as well as five views of each test object. Here both L2G and L2G+DeCo consistently outperform the baselines. When testing on a single view, the effect of the DeCo encoder on top of GPNet provides only a marginal difference. Instead, the advantage of DeCo becomes clear for both GPNet and L2G in simulation experiments with five views, which should be considered the most reliable case. 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 Fig. 4 confirms this behavior. Intuitively, the curve is comparable to a traditional precision-recall curve where the simulation success rate resembles the precision and the coverage is the recall. It was obtained by slowly lowering the confidence threshold applied on the classifier output \( s \) from 1 to 0.5. Only for the predictions with a score above the current threshold we compute the simulation-based success and coverage to define each point of the plot.

![Success-Coverage Curve](image)

**Fig. 4.** This plot is comparable to a traditional precision-recall curve: it shows the success rate in relation to the coverage for different sensitivity settings. The area under the curve is indicative of model performance and demonstrates the improvements of L2G over GPNet as well as the beneficial effect of using the DeCo encoder over PointNet++.

![Robustness Analysis](image)

**Fig. 5.** Robustness analysis with respect to the number \( M \) of sampled contact points and the cardinality \( nn \) of the neighborhood of each sampled contact.
TABLE I
SIMULATION-BASED (LEFT) AND RULE-BASED (RIGHT) EVALUATION RESULTS ON SHAPENETSEM-8. GPNet refers to the top results in [1]. GPNet* indicates the results obtained by re-running the authors’ code. Top: results for a single object view. Bottom: results averaged over 5 different views per object.

| Method          | One view - Simulation Based | One view - Rule Based |
|-----------------|-------------------------------|-----------------------|
|                 | success rate @k% | coverage rate @k% | success rate @k% | coverage rate @k% |
| GPNet [10]      | 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*          | 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 [16]| 91.1 | 89.3 | 81.9 | 67.0 | 89.4 | 85.9 | 82.5 | 7.7  | 15.1 | 24.6 | 33.9 |
| L2G+DeCo [16]   | 94.6 | 93.5 | 91.4 | 82.9 | 95.2 | 94.9 | 94.6 | 94.5 | 20.6 | 29.2 | 35.5 | 41.8 |

| Method          | Five views - Simulation Based | Five views - Rule Based |
|-----------------|--------------------------------|-------------------------|
|                 | success rate @k% | coverage rate @k% | success rate @k% | coverage rate @k% |
| GPNet*          | 87.0 | 85.4 | 79.2 | 59.3 | 89.6 | 87.9 | 83.7 | 68.1 | 7.6  | 15.5 | 24.9 | 34.4 |
| L2G             | 91.0 | 89.0 | 87.2 | 82.4 | 94.9 | 96.0 | 96.2 | 95.8 | 19.1 | 29.1 | 34.0 | 39.8 |
| GPNet*+DeCo [16]| 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 [16]   | 93.0 | 92.1 | 90.7 | 83.2 | 96.4 | 96.5 | 96.4 | 95.4 | 20.3 | 31.0 | 36.5 | 41.5 |

Fig. 6. 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.

TABLE II
SAMPLE AND TIME COMPLEXITY ANALYSIS. THE REPORTED ACCURACY IS SIMULATION-BASED SUCCESS RATE @10%.

| Method          | Ratio of Training Set | Inference Time per shape (s) |
|-----------------|-----------------------|-----------------------------|
|                 | 1/4 | 1/2 | 1 | min | max |
| GPNet*          | 71.9 | 79.6 | 87.0 | 0.909 | 50.861 |
| L2G             | 73.7 | 89.7 | 91.0 | 0.001 | 0.339 |
| GPNet*+DeCo [16]| 80.8 | 90.2 | 91.2 | 0.935 | 46.897 |
| L2G+DeCo        | 78.9 | 91.7 | 93.0 | 0.016 | 0.365 |

C. Robustness, Sample and Time Complexity Analysis

We focus on the simulation-based experiment with five views and success rate @10% of the previous section to analyze 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 $N_F(q)$ centered at each sampled contact $q$. The plots in Fig. 5 show that the performance of L2G has a mild dependence on $M$, with a decrease in success rate for very high values. On the other hand, L2G proves robust to the choice of $nn$. Overall, both L2G and L2G+DeCo maintain their advantages over the GPNet baseline.

To investigate the sample complexity, we reduced the amount of training data to 1/4 and 1/2. The results in Table II (left) show that the effect of DeCo becomes more and more evident when reducing the training data support, largely outperforming the PointNet++-based models. This highlights the importance of local+global feature encoding and the ability of self-supervised pre-training to improve generalization and sample efficiency.

Finally, we compare GPNet and L2G in terms of minimum and maximum inference time per input point cloud. The results in Table II (right) are showing a significant advantage of our approach.

D. Generalization on YCB

To assess the generalization abilities of our L2G model, we employ the YCB dataset as test set. It has a much broader range of object categories with respect to ShapeNetSem-8 used for training. Additionally, objects in this YCB dataset
strongly vary in dimension and appear in various 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 appearance domain shift. The results for the simulation-based and rule-based analysis on YCB-8 are in the left part of Table III 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. Interestingly, the DeCo encoder drastically improves the scores of GPNet, leading to top success rates @10%. While DeCo also improves the performance of L2G, the margin is much smaller here. On YCB-76, L2G shows top simulation-based results, demonstrating its generalization abilities with as well as without DeCo. In addition to the quantitative analysis, in Fig. 6 we also compare the grasp predictions of GPNet and L2G qualitatively in simulation. The sponge can be considered an adversarial object: It is very flat and its side length is very 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 no spherical objects were present in the training set. Especially with increasing @k%, GPNet tends to predict many spurious, unreasonable grasps, which is not the case for L2G. For long, elongated objects like the spoon, the grid-based approach employed in GPNet fails entirely. L2G in contrast is more robust to unknown geometries, which we attribute to the fact that it does not rely on any geometric heuristics for the grasp proposal.

E. 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. 7). The point clouds are constructed from a single-view depth image and pre-processed with a 3D bounding box and a plane removal filter to isolate the object. 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 [35]. If no feasible plan can be found, we swap the contact points and plan for this symmetric grasp, which might be more suitable for the robot configuration. If this still does not give a feasible trajectory, we resort to the second or at most the third grasp prediction. The robot approaches the grasp pose in a linear motion, executes the grasp, lifts the object by 30cm, moves back down, and then releases the object. 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 the two sets: one with 28 custom items from the same categories as in ShapeNetSem-8, and one with a set of 20 YCB objects for better experimental reproducibility (see Fig. 7). The results are displayed in Table V 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 depending on the size: 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 models 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 approaches are sensitive to object size. For real shapes much larger or smaller than in the training data the predictions deteriorate or there are no predictions with sufficiently high confidence at all. In particular, the larger bottles and boxes have been affected by this. The chosen setting comes with further challenges introduced by 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 especially crucial when grasping (flat) objects close to the ground, like the 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.
TABLE IV
REAL-WORLD ROBOT GRASPING EXPERIMENTS. WE REPORT THE FRACTION OF SUCCESSFUL TRIALS OUT OF FIVE PERFORMED ON EACH OBJECT AND AVERAGED PER CATEGORY GROUP. IN PARENTHESIS WE INDICATE THE NUMBER OF OBJECT INSTANCES PER CATEGORY.

| Category      | GPNNet | L2G  | Category      | GPNNet* | L2G  |
|---------------|--------|------|---------------|---------|------|
| box (6)       | 0.47   | 0.75 | box (7)       | 0.31    | 0.71 |
| soda can (2)  | 0.50   | 1.00 | mug (1)       | 0.20    | 0.20 |
| cylinder (5)  | 0.52   | 0.76 | bowl (1)      | 0.00    | 0.00 |
| bottle (5)    | 0.25   | 0.48 | cylinder (5)  | 0.72    | 0.64 |
| mug (3)       | 0.27   | 0.40 | sphere (3)    | 1.00    | 1.00 |
| bowl (4)      | 0.15   | 0.10 | bottle (3)    | 0.07    | 0.27 |
| toy car (3)   | 0.33   | 0.27 | average       | 0.38    | 0.47 |

V. CONCLUSIONS

In this paper we introduced L2G, our lightweight end-to-end method for 6-DOF grasping from partial object point clouds by joint learning of contact point sampling, grasp regression, and grasp classification. In our extensive evaluation we thoroughly compared this approach to the main competitor GPNet [1] and could demonstrate the advantages of L2G, namely predicting a larger, more diverse set of reliable grasps. Furthermore, we demonstrated that it better generalizes to unseen objects with significant shape variations. Using the pre-trained feature encoder DeCo [16] instead of a standard PointNet++ encoder 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 to work with more complex scenes containing multiple objects. Moreover, the contact point sampling procedure could be further guided in case of available prior knowledge about the downstream manipulation task to enable task-specific grasping. We plan to investigate both these directions in future work.

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] P. Brook, M. Ciocarlie, and K. HSiao, “Collaborative grasp planning with multiple object representations,” in ICRA, 2011.
[3] C. Borst, M. Fischer, and G. Hirzinger, “Grasp planning: how to choose a suitable task wrench space,” in ICRA, 2004.
[4] 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 Robotics: Science and Systems (RSS), 2017.
[5] D. Morrison, P. Corke, and J. Leitner, “Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach,” in RSS, 2018.
[6] H. Zhang, X. Lan, S. Bai, X. Zhou, Z. Tian, and N. Zheng, “Roibased robotic grasp detection for object overlapping scenes,” in IROS, 2019.
[7] Y. Zhou, X. Lan, H. Zhang, Z. Tian, Y. Zhang, and N. Zheng, “Fully convolutional grasp detection network with oriented anchor box,” in IROS, 2018.
[8] 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.
[9] 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.
[10] A. Mousavian, C. Eppner, and D. Fox, “6-dof graspnet: Variational grasp generation for object manipulation,” in IEEE CVG, 2019.
[11] A. Murali, A. Mousavian, C. Eppner, C. Paxton, and D. Fox, “6-dof grasping for target-driven object manipulation in clutter,” in ICRA, 2020.
[12] 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.
[13] 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.
[14] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, “Dynamic graph can for learning on point clouds,” ACM Transactions on Graphics (TOG), 2019.
[15] N. Singh, Z. Blum, and N. Renjith, “Point cloud grasp classification for robot grasping,” 2018.
[16] A. Alliegero, D. Valsesia, G. Fraocastoro, E. Magli, and T. Tommasi, “Denoise and contrast for category agnostic shape completion,” in CVPR, 2021.
[17] H. Wang, Q. Liu, X. Yue, J. Lasenby, and M. J. Kusner, “Unsupervised point cloud pre-training via occlusion completion,” in ICCV, 2021.
[18] X. Yan, J. Hou, 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.
[19] A. Depierre, E. Dellandrea, and L. Chen, “Jacquard: A large scale dataset for robotic grasp detection,” in IROS, 2018.
[20] 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.
[21] A. Saxena, L. S. Wong, and A. Y. Ng, “Learning grasp strategies with partial shape information,” in AAAI, 2008.
[22] A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, T. Asfour, and S. Schaal, “Template-based learning of grasp selection,” in ICRA, 2012.
[23] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber, “A survey on learning-based robotic grasping,” Current Robotics Reports, Sep 2020.
[24] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Krüger, 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.
[25] 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.
[26] 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.
[27] V-D. Nguyen, “Constructing force- closure grasps,” The International Journal of Robotics Research, vol. 7, no. 3, pp. 2370–2377, 2018.
[28] O. Dovrat, I. Lang, and S. Avidan, “Learning to sample,” in CVPR, 2019.
[29] I. Lang, A. Manor, and S. Avidan, “SampleNet: Differentiable Point Cloud Sampling,” in CVPR, 2020.
[30] C. Eppner, A. Mousavian, and D. Fox, “A billion ways to grasp - an evaluation of grasp sampling schemes on a dense, physics-based grasp data set,” in ISRR, 2019.
[31] L. Li, Y. 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.
[32] M. Savva, A. X. Chang, and P. Hanrahan, “Semantically-enriched 3d models for common-sense knowledge,” in CVPR Workshops, June 2015.
[33] E. Coumans and Y. Bai, “Pybullet, a python module for physics simulation for games, robotics and machine learning,” http://pybullet.org, 2016–2021.
[34] 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.
[35] I. A. Sucan and S. Chitta, “Moveit!” https://moveit.ros.org/, 2013.