Efficient and Accurate Candidate Generation for Grasp Pose Detection in SE(3)

Andreas ten Pas[1], Colin Keil[1], and Robert Platt[1]

Abstract— **Grasp detection of novel objects in unstructured environments is a key capability in robotic manipulation. For 2D grasp detection problems where grasps are assumed to lie in the plane, it is common to design a fully convolutional neural network that predicts grasps over an entire image in one step. However, this is not possible for grasp pose detection where grasp poses are assumed to exist in SE(3). In this case, it is common to approach the problem in two steps: grasp candidate generation and candidate classification [1], [2], [3], [4]. Since grasp candidate classification is typically expensive, the problem becomes one of efficiently identifying high quality candidate grasps. This paper proposes a new grasp candidate generation method that significantly outperforms major 3D grasp detection baselines.**

I. INTRODUCTION

For many robot manipulation tasks, grasping is a key step. In order to grasp an object successfully, the robot must place its hand in exactly the right position and orientation before attempting to close its fingers. Grasp success depends on the geometry, friction, mass, and deformability of both the robot hand and the object. Possible collisions with nearby objects further complicate the grasping process.

There have been several recent approaches that employ deep learning techniques to predict grasp poses given point cloud data as input [5], [6], [7], [8]. However, while these methods have been shown to enable a robot to grasp a large variety of rigid objects, with many different shapes, sizes, and textures, they are often restricted to grasp detection in the plane or with a shallow out-of-plane orientation. This is a severe limitation because it prohibits the object from being grasped from arbitrary directions in SE(3) and thus restricts how the object can be manipulated.

Some recent work has focused on grasp detection in SE(3) that can exploit all the freedoms of positioning and orienting a robotic hand [1], [2], [3], [9]. Such methods often follow a *generate-and-test* strategy, where a number of grasp candidate poses are first generated and then tested for a grasp quality criterion using a learned classifier. The key question here is how to generate grasp proposals. Many methods [1], [10], [4] either generate large numbers of grasp proposals exhaustively or use heuristics to prune geometrically infeasible proposals. These methods can be slow because they typically perform collision checks and/or grasp stability tests against the observed scene geometry [1], [9]. An important exception to the above is the recent work of Mousavian [3] who uses the decoder portion of a variational autoencoder to propose grasp poses. This is a more flexible approach, but it can be slow.

In this paper, we ask whether it is possible to match or exceed the performance of existing baselines such as [1] and [3] using a relatively simple neural network model architecture to generate grasp proposals from 3D point clouds. Our method first samples a set of points from the cloud. For each sample, it efficiently encodes the geometry of a region about the sample by cropping local rectangular patches from each of several orthographic projections of the global scene. This representation of the local region is input to a neural network model that predicts which orientations about the sampled point are likely to contain a grasp. This method is computationally more efficient than prior methods [1], [3] because both the cropping process and the candidate generation process are very quick. We demonstrate in simulation that both the accuracy and speed of this method compare favorably with previous approaches and we show that the method is effective in practice on a real robot.

II. RELATED WORK

A. Object Proposals in Computer Vision

The ideas for proposing grasp poses algorithmically are analogous to proposing object locations and bounding boxes in computer vision. Girshick et al. laid the groundwork [11] in which a number of candidate object regions is proposed and individually evaluated by a convolutional neural network (CNN). Their work was later extended with region of interest pooling which improves object detection accuracy and reduces computational cost [12]. Ren et al. [13] further improved object detection by predicting candidate object bounding boxes with a region proposal network (RPN), extracting features for each box with region of interest pooling, classifying the box content and regressing the box parameters. Features for the RPN and the box evaluation are shared to further reduce computational inference cost. By learning to predict offsets from fixed reference bounding boxes, called *anchors*, the authors predict bounding boxes of different aspect ratios and sizes. Contrastively, Liu et al. [14] used a single network that makes predictions about a discretized space of default bounding boxes whose aspect ratios and scales are fixed. Similarly, our approach learns to predict 3D rotational "offsets" from a fixed set of reference grasp poses.

[1]All authors are with Khoury College of Computer Sciences, Northeastern University, Boston, MA, USA atp at ccs neu edu

B. 6-DOF Grasp Pose Detection

The state-of-the-art in grasp pose detection are deep learning based methods. Much work has focused on 3- or 4-DOF grasp configurations [5], [6], [7], [8], [15], [16]. Typically, such approaches keep two orientation DOFs fixed and only learn the orientation for one axis, usually the grasp approach axis. This severely restricts the ways in which an object can be grasped and thus restricts the manipulation tasks that such grasps can be used for. Another avenue of recent research explored 6-DOF grasping.

Mousavian et al. used a variational auto-encoder to sample a diverse set of grasp poses for an object [3]. These poses are then iteratively evaluated with a separate network and refined using the evaluator network's gradient. All of their networks are based on the PointNet++ architecture [17]. This approach is extended by Murali et al. [18] to grasping a particular object in clutter. Qin et al. [9] used a single-shot grasp proposal network that predicts grasp poses by regression directly from the complete visible scene. Their network is based on the PointNet++ architecture [17]. In prior work, we detected grasp poses by first generating grasp proposals based on local geometry and then classifying multi-view projected features for each of the proposals [1]. Liang et al. extended this approach to directly work with point cloud data by training a network based on the PointNet [4] architecture. Gualtieri and Platt learn to grasp as part of pick and place modeled as an MDP with abstract state and action representations, and solve this MDP with a hierachical method that tells the robot where to look at in the scene [2]. Zhou and Houser [19] learn to predict a grasp score from a depth image and use the score for grasp pose refinement. Yan et al. [20] use generative 3D shape modeling to learn scene reconstruction that is then used to predict grasp outcomes. Merwe et al. [21] learn to reconstruct the object geometry and predict a score for a multi-fingered robot hand configuration that consists of a 6-DOF hand pose and finger joint positions. Lu et al. [22] train a 3D CNN on voxels to predict grasp success for a 16-DOF hand. Wu et al. [23] train a network to iteratively decide whether to zoom in or to execute a grasp for multi-fingered hands. Varley et al. [24] train a 3D convolutional network to perform object shape completion and then find grasps for the completed object. In contrast to the literature, we directly predict grasp poses based on the point cloud geometry and do not try to complete objects. Instead of generating grasp proposals in a heuristic fashion, we use a neural network to evaluate a fixed set of grasp poses.

III. Problem Definition

We define a robot, $\mathcal{R}$, as a robot hand that can move to an arbitrary pose in its workspace. The **robot state** is defined by the pose of the hand, $h \in SE(3)$, and the configuration of the fingers. In this work, we assume the hand is a parallel jaw gripper whose jaws are actuated by a single degree of freedom. The **world state** $\mathcal{W} \in \mathbb{W}$ is a description of the state of the environment and objects around the robot, where $\mathbb{W}$ is the set of all possible world states. A **point cloud**, $\mathcal{C} \in \mathbb{C}$, is a finite set of points in the robot's environment which are obtained by one or more depth sensors, where $\mathbb{C}$ denotes the space of possible point clouds that can be generated by the sensor arrangement of the robot. The world state is a latent variable that is observed only via the point cloud. In particular, we model the depth sensor(s) as a function $\Lambda : \mathbb{W} \rightarrow \mathbb{C}$ that encodes aspects of world state as a point cloud. Given a robot $\mathcal{R}$ and a point cloud $\mathcal{C} = \Lambda(\mathcal{W})$ for some hidden world state $\mathcal{W}$, the problem of **grasp pose detection** is to find a robot hand pose, $h \in SE(3)$, such that if the robot hand is moved to h and the fingers are closed, then some object in the world $\mathcal{W}$ can be grasped.

We approach grasp pose detection as a two step process. First, we generate a set of grasp pose proposals. Second, we evaluate the probability that each proposal is an actual grasp. In this paper, we are mainly concerned with improving the first step. We formulate the problem of generating grasp proposals as learning a function, $f : \mathcal{C} \rightarrow \mathrm{SE}(3)$, that maps from the space of possible point clouds to the space of possible robot hand poses.

IV. Grasp Detection System

Figure 1 shows an overview of the end-to-end grasp detection system. There are two components: the proposal scoring network that generates a large set of grasp candidates quickly and the grasp classification network that makes a high quality binary prediction about each grasp candidate. Since the creation of images for the grasp classification network is computationally expensive [1], the proposal scoring network is essential because it focuses attention on promising grasp candidates.

A. Grasp Candidate Generation

If we were doing grasp detection in the plane (i.e. considering only x, y, θ grasp poses), then the simplest approach to grasp candidate generation would be a single forward pass through a fully convolutional neural network, e.g., as in Mahler et al. [6]. However, we cannot use this approach here because we are doing grasp detection in SE(3). Instead, we do the following. We sample k points from a region of interest in the point cloud. For each of the k points, we will do a single forward pass through the proposal scoring network predicting which hand orientations about the sampled point are likely to be a good grasp.

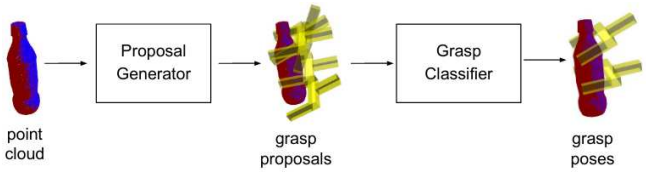


Fig. 1: Overview of our approach. Given a point cloud, we generate grasp pose proposals using a convolutional neural network. We then classify the proposals as actual grasps using another convolutional neural network.

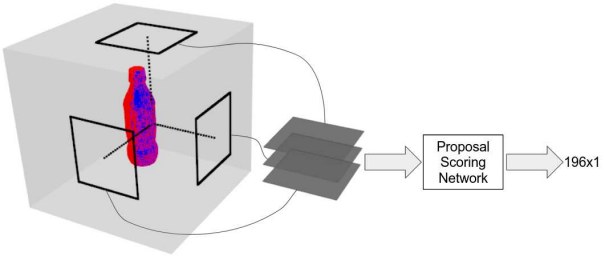


Fig. 2: Grasp proposals are predicted with a proposal scoring network that takes a 3-channel image and produces a score for each grasp pose represented by the image. Each channel is an orthographic representation of a subset of the point cloud.

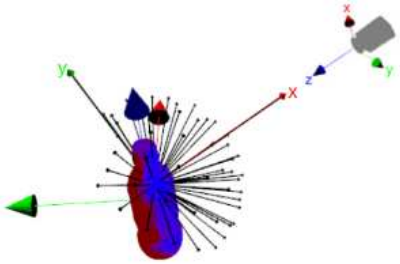


Fig. 3: The robot hand orientations considered by our method. Each black arrow corresponds to a hand approach direction. For each direction, there are four possible orientations about the approach axis.

1) Input to the proposal scoring network: The proposal scoring network will take as input information about the local point cloud near the sampled point by extracting a bounding cube centered on the sampled point. Since we plan to do one forward pass through the proposal scoring network per point sampled (k forward passes), it is critical that this bounding cube is encoded efficiently. We do the following. First, we create a three-view orthographic representation of the entire point cloud expressed in the reference frame of the camera and centered on a point of interest (e.g., the estimated object center). Each of the three orthographic views is associated with a height map that describes the scene when viewed from that direction. These three orthographic projections are illustrated in Figure 2 as the sides to a cube that encloses the point cloud. Second, we crop a fixed-size rectangle in each of the three orthographic projections centered on the sampled point. Each of these crops encodes information about the local neighborhood of the sample (the three orthogonal boxes outlined in black in Figure 2). Finally, the height maps contained in these three rectangles are stacked and input to the grasp proposal network as a three-channel image. Notice that the approach above is very efficient per sampled point. For each point, we simply crop a rectangle from each of three images and copy them into the network input as a three-channel image (one channel for each crop).

2) Output of the proposal scoring network: Our baseline architecture for the proposal scoring network consists of two convolutional layers (no zero padding, stride 1, kernel size 5) followed by two FC layers followed by a sigmoid layer with m outputs. Each output is interpreted as the probability that a grasp exists at a particular orientation when the closing region of the hand is centered on the sample point and the hand is "pushed" forward until some part of it contacts the point cloud. In this paper, we focus on the case where there are $m = 196$ orientations, but the method should generalize to other values of m. Note that since the output layer is a sigmoid instead of a softmax, the network makes predictions about grasps at multiple orientations for each sampled point. Figure 3 illustrates the m orientations about which our network makes predictions. These orientations cover roughly a half dome of orientations pointed in the direction of the camera and are expressed in the reference frame of the camera. To summarize, the proposal scoring network takes as input the visible point cloud geometry in the vicinity of the sample point and outputs predictions about which orientations around the sampled point are likely to be grasps.

3) Loss Function: Learning the parameters of the proposal scoring network is a multi-label classification problem where the labels are multi-hot vectors, $y \in [0,1]^m$, where m is the number of orientations considered by the proposal scoring network. Generally, given some object part, multiple robot hand orientations can usually lead to a successful grasp. We treat this problem as if there are m independent binary classification problems, i.e., the binary cross entropy loss function is calculated separately for each orientation.

B. Grasp Classification

To classify grasp proposals, we use a grasp classification network that is similar to the one presented in our earlier work [1]. The network takes as input information about the local point cloud that would be contained in the closing region of the robot hand at the grasp pose. The points in that region are orthographically projected onto a plane parallel to the hand's approach axis. A height map of these points makes up the first channel and the average surface normal at each of these points makes up three more channels of a four channel image. We use this image type as it is much faster to compute than ones with a larger number of channels [1]. The output of the network is a binary label that is one if the grasp is predicted to be successful and zero otherwise. The architecture for the network is the same as for the proposal scoring network. We use the cross entropy loss to train the network.

V. Network Training

We implemented our networks in PyTorch 1.4 [25]. For point cloud processing, we used Open3D 0.9 [26]. To generate synthetic scenes, we used Pyrender [27].

A. Ground Truth Grasps

We consider a grasp to be successful if it (1) is collision-free and (2) has force closure. We assume a parallel jaw gripper with soft contacts so that an antipodal grasp is a

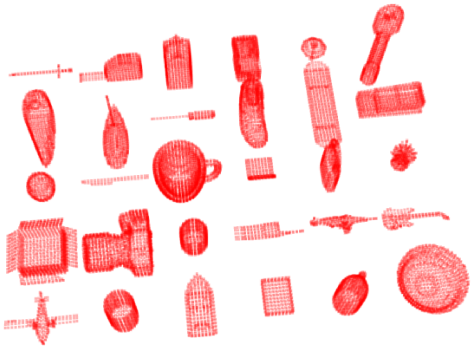


Fig. 4: Example instances from 3DNet object categories used for training.

sufficient and necessary condition for force closure [28]. We consider all points within a fixed distance from each finger as possible contact points. We then check if the surface normals of those points lie within the friction cone with a fixed friction coefficient. This is the same procedure as in our earlier work [1].

B. Generating Data for Training

Our training set consists of a total of 300 objects, ten objects for each of thirty categories in 3DNet [29] (see Figure 4. To fix holes and other issues with the 3DNet object meshes, we apply a preprocessing method that produces watertight meshes with a 2-manifold topology and vertices distributed about uniformly on the object surface [30]. We obtain point clouds from a mesh by placing it at the origin of a fixed world frame and scaling it by a randomly chosen factor such that the object's extents are within $[0.01\text{m}, 0.07m]$ We then obtain single-view, partial point clouds by sampling camera locations uniformly on a sphere that has a radius of 0.5m and that is centered on the world frame's origin. For our training set, we sampled 20 camera locations. From each of the resulting point clouds, we uniformly sampled 100 points and evaluated grasps with each method used for the simulation experiments against the corresponding ground truth mesh. In total, we evaluated about 117 million grasp poses to train the proposal scoring network and 600,000 poses for the grasp classification network.

C. Training the Networks

We use a batch size of 64 and images of width 60 and height 60. All models are trained with stochastic gradient descent with a momentum of 0.9. The learning rate starts from 0.01 and is exponentially decreased after each epoch by a factor of 0.96. We only use the synthetic data (described above) for training.

VI. Antipodal Detection Experiments

In this section, we evaluate how well out method can detect antipodal (i.e. two finger force closure) grasps for objects presented in isolation. We obtain synthetic point clouds in the same way as described in Section V.

A. Object Test Set

We evaluate on novel object instances from the same thirty object categories in 3DNet which were used for training. Objects are randomly scaled and viewpoints are generated in the same way as for the training set (see Section V-B. For each viewpoint, we sample 100 points uniformly from the cloud.

B. Comparisons

We compare our method, called QD, against two ablations, QD:GC and QD:ROT, and one baseline GPD. QD:GC is an ablation of QD that consists of only the grasp classification network (second half of Figure 1). QD:ROT is an ablation that consists of only the proposal scoring network (first half of Figure 1). Both of these ablations are one-stage grasp detectors which evaluate all potential grasp poses. The GPD baseline is the 2-stage grasp detector from [1]. It is structurally similar to the method in this paper, but uses a geometric proposal strategy instead of a learned proposal generator.

C. Evaluation Metrics

We evaluate our approach with three metrics: precision, recall, and detections per second (DPS). Precision and recall have the standard definitions. DPS is the number of grasp predictions produced by our method (both stages of Figure 1) per second. In addition to the standard precision/recall plot (Figure 5) we also show precision/DPS (Figure 6). Precision/DPS shows the tradeoff between precision and the number of predictions the grasp detection system makes. Ideally, our system would achieve a point in the upper right corner of this plot – lots of high precision grasps produced per second.

D. Results: Isolated Objects

Figures 5 and 6 compare QD against QD:GC, QD:ROT, and GPD for the objects in the *3DNet* test set in terms of precision/recall and precision/DPS. First, notice that QD outperforms the GPD baseline by a large margin in both plots. While the QD:GC ablation is very similar to QD in terms of precision/recall, notice that it is much slower than QD, especially at high precision values. (At 90% precision and above, QD:GC is approximately an order of magnitude slower than QD.) We attribute this speedup to the fact that our candidate generation strategy allows us ignore a large number of candidates that the GC network would otherwise need to consider. Whereas QD:GC has to exhaustively calculate grasp images for every possible grasp proposal while QD only considers a subset. Looking at QD:ROT, notice that it is much faster than all other methods, but its precision/recall curve is significantly lower than either QD:GC or QD. This confirms that the proposal scoring network is fast but that it is not as accurate as the grasp classifier. These results emphasize that both components of QD are important for its performance: ROT allows us to subsample the set of grasp proposals to reduce runtime and GC allows us to make more

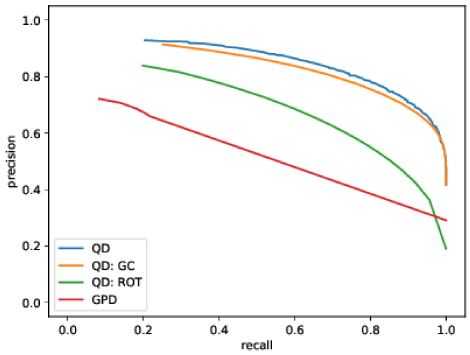


Fig. 5: Precision vs recall comparison on novel 3DNet object instances.

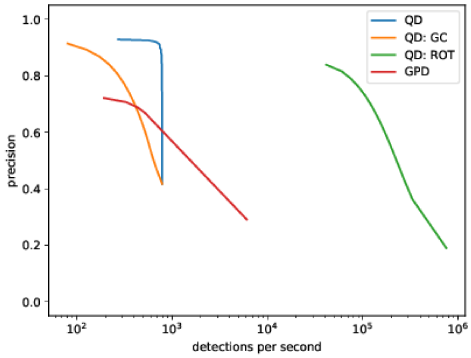


Fig. 6: Precision vs detections per second on novel 3DNet object instances.

accurate predictions because of the more informative grasp descriptor representation.

Figure 7 shows the precision that can be achieved by QD and GPD for each of the 30 object categories considered in our experiments. Notice that while QD outperforms GPD for all categories, the amount of outperformance is significant for some categories, like books. We believe that this simply reflects the difficulty the geometry based GPD proposal generator has for certain objects.

VII. PyBullet Simulation Experiments

In this section, we investigate the performance of our method in a PyBullet physics simulation of robot grasping. While point clouds are still obtained synthetically in the same way as before, we measure the actual performance of grasps in the simulator.

A. Simulating Grasps

After obtaining a point cloud as described in the last section (using Pyrender) and performing grasp detection, we simulate a grasp as follows. This process is designed to mimic the simulation used in 6DoF GraspNet [3], which is not publicly available. We place a free floating Robotiq 2F-85 parallel jaw gripper in a pose relative to the object that corresponds to the selected grasp and close the fingers. After the grasp is formed, we turn on gravity such that it

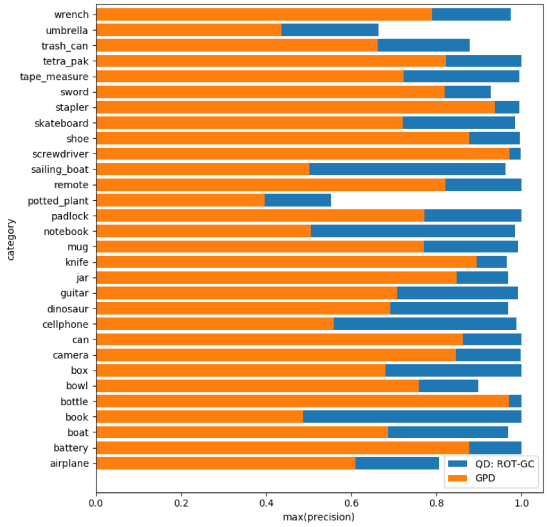


Fig. 7: Per-category maximum precision for GPD and QD on novel, isolated object instances from 3DNet.

points in the direction that the gripper is pointing (it is as if the gripper is pointed down). We then perform a predefined shaking motion. Collision between the hand and the object are checked with FCL [31]. A grasp is considered to be successful if the hand is not in collision, the object's center of mass moves less than 0.5m from the hand, and the z-axis distance between the object and the hand's base is less than 0.3m. During this process, the friction coefficient between the gripper and objects is kept constant, and all objects are simulated with a uniform mass of 0.5kg.

To make the ground truth robust against errors caused by imprecise robot kinematics and sensor noise, we simulate a small number of randomly perturbed poses for each grasp pose (i.e., in addition to the original proposal). The perturbed grasp poses are generated by applying a uniformly distributed random rotation between 0 and 5 degrees about a random axis, and a uniformly distributed random translation between 0 and 3mm. We decide the final label by a majority vote over the labels of the perturbations and the original grasp pose, i.e., if 3 out of the 5 poses are successful, we label the grasp as positive.

We compare our method to Graspnet [3], a recently pub-

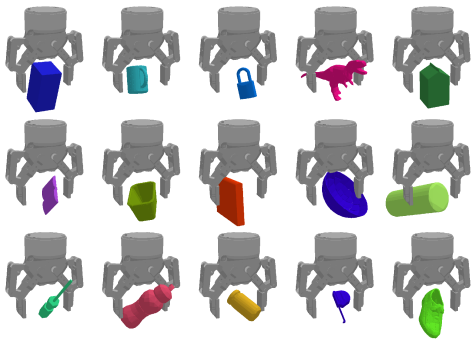


Fig. 8: Examples of grasps simulated in PyBullet.

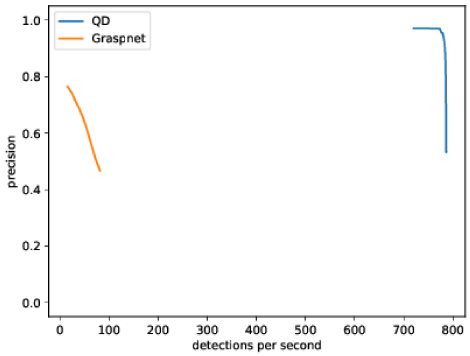


Fig. 9: Precision/DPS comparison between QD and Graspnet. QD was trained using the full 3DNet object set and we use the version of Graspnet pretrained in [3].

(a) (b)

Fig. 10: Robot experiments setup. (a) A point cloud is acquired from a single viewpoint. (b) Object set for isolated grasping.

lished 6-DOF grasp pose detection method. After obtaining a point cloud using Pyrender, we run Graspnet with the following default settings: 200 samples for the generative network and 10 refinement steps using Metropolis-Hastings sampling. We then label the resulting grasp poses using the PyBullet simulator as described above. We evaluate on unseen object instances from the same object categories that Graspnet has been evaluated on: bottles, bowls, boxes, cylinders and mugs (BBBCM). Cylinders and boxes are generated using Open3d with random dimensions. Bowls, bottles, and mugs are taken from the 3DNet object set.

B. Results: Physics Simulation of Grasps

Figure 9 shows the comparison with Graspet [3]. Our method is faster and produces grasp detections with higher precision. We do not perform the precision/recall comparison here because the Graspnet proposal generator does not have a discrete hypothesis set and we therefore cannot calculate recall. However, the precision/DPS shows that our method is much faster with no worse precision. It is important to point out here that Graspnet is at a disadvantage relative to our detector. The problem is that Graspnet cannot be retrained on our training set without access to high-end GPUs. As a result, we are using the version of pretrained Graspnet weights found using the labels generated in [3]. Although both methods were trained using the same object categories, it is likely there are slight differences between the labels in the training set and that this puts Graspnet at a disadvantage. However, given what is publicly available, this is the closest comparison that is possible.

VIII. Robot Experiments

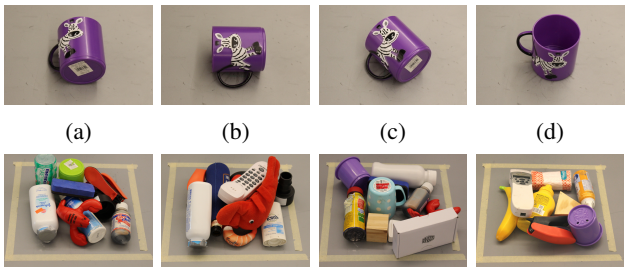
A. Setup

We conducted all our robot experiments on a Universal Robots' *ur5* robot with six DOFs and a Robotiq *2F-85* parallel jaw gripper with one DOF. A *Structure IO* depth camera is mounted on the wrist of the robot arm (see Figure 10a). The robot is mounted to the same table on which the objects to be grasped are placed. Our object set is shown in Figure 10b. We chose objects for which at least one side fits into the robot hand (preferably more than one). All objects weigh less than 0.5kg.

The computer used for these experiments has an Intel i7-7800X 3.50GHz CPU, 64GB system memory, and an Nvidia GeForce GTX 1080 graphics card with 8GB of memory. All robot experiments were run on ROS Melodic [32]. For inverse kinematics, an analytic solution was used [33]. Motion planning and collision checking was done in OpenRAVE [34] and trajopt [35]. We use toppra [36] to produce a trajectory from a path under joint velocity and acceleration constraints.

To generate grasp poses, the robot moves its hand to a single viewpoint and obtains a point cloud from its wrist-mounted depth camera. The viewpoint is chosen such that the system can observe as much as possible of the scene. Our algorithm then takes the point cloud and produces a set of grasp poses. We sample $n = 500$ points from the point cloud. From the $n \times m$ outputs of ROT (n sampled points, m hand orientations), we select the top-k ($k = 20$) scoring orientations for each of the n points, and then uniformly subsample 300, and calculate grasp images for those. Then, we select the top-k ($k = 150$) scores from GC.

During initial testing, we found that grasps found using our method sometimes generated collisions with the object due to small kinematic errors in robot arm positioning. We believe these errors occur because the training set we used does not take proximity to collision into account. In order to correct for this problem on the robotic system, we followed the following procedure. First, we translate each detected grasp pose in the hand closing direction such that the visible points are centered in the grasp. Then, for grasp poses where the hand is further than a threshold distance from the closest point in the hand closing region, we translate the hand forward along its approach direction as far as possible without generating collisions. Next, we find sets of geometrically aligned grasps using a similar procedure as in [10] and generate one additional grasp per cluster that corresponds to the geometric center of the cluster. These centers are often close to object (part) centers. These centers and all grasps produced previously are then checked for collisions with the point cloud. We select a grasp to be



(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 11: Robot experiments setup. (a-d) Examples of an isolated object scene with the four different configurations. (e-f) Examples of dense clutter scenes.

executed in a hierachical manner by considering grasps in the following order: cluster centers, cluster inliers, grasps with a score above 0.5, and grasps with a score below 0.5. For each group, we solve inverse kinematics for all grasps in the group and then compare them using heuristics that take into account the height of the grasp, how vertical the approach direction is and how wide the hand needs to be opened (similar to [1]). The former step reduces the number of inverse kinematics problems to be solved, and the second step can improve grasp success on in cluttered scenes because it mitigates the problem of collisions with objects located below other objects in a pile. If no grasp pose can be reached by the robot arm, the next group of grasps is considered.

B. Isolated Object Grasping

Here, we characterize how well our method is able to grasp objects that are placed in isolation on a table in front of the robot. For each object, we attempt a grasp for up to four distinct configurations:[1] one to three horizontal orientations and one upright (see Figure 11(a-d)). We generate a point cloud using a single depth sensor viewpoint and select a grasp using the procedure described in Section VIII-A. Table I (first column) shows the results. The robot achieved an overall grasp success rate of 90.27% (204 successes out of 226 attempts). Grasps on box corners and the edges of objects were the most common failure modes in this experiment.

C. Dense Clutter Grasping

Here, we evaluate our method in a dense clutter setting similar to the one described in [1]. First, we place ten randomly selected objects from the object set into a box. The object set for this experiment is the same as shown in Figure 10b, except for the three bowls. We then turn the box upside down on a table in a fixed location, shake the box, and remove it. Finally, the robot tries to grasp the objects one by one until no objects remain on the table. Examples of clutter scenes are depicted in Figure 11(e-h).

Results are shown in the second column of Table I. Overall, the robot attempted 176 grasps out of which 144 were successful (81.82% success rate). The most common failure modes were edge grasps and corner grasps on boxes some of which may have failed due to noise in the robot's calibration. Furthermore, the robot removed 144 out of 150 objects from the table (96% success rate). The reason for the six objects which were not removed is that they rolled out of the robot's reach or view because of collisions which occurred while grasping another object.

[1]The bowls have one configuration that is repeated four times, and the wooden cube has two (a and c).

TABLE I: Results of robot grasping experiments.

	Isolated objects	Dense clutter
Num grasp attempts	226	176
Num grasp successes	204	144
Grasp success rate	90.27%	81.82%
Object removal rate	/	96.00%

IX. Conclusion

We proposed a method for predicting grasp poses which were represented as height maps of an orthographically projected point cloud. In simulation experiments, we found that our method is both fast and precise, in particular compared to geometric grasp proposal generation. We also showed that we can learn grasps based on a physics simulation. Finally, we demonstrated that our method can effectively grasp objects in isolation and in dense clutter on a robot.

In future work, we could extend our method to learn collisions with an object's support plane, like a table, and with its surroundings, like in cluttered scenes. Another interesting direction is to learn task-dependent grasps, e.g., to detect grasps on the handle of a drill. Typically, task dependency is learned separately from grasp pose detection but could be learned together with grasp quality.

Acknowledgment

We thank Andrea Baisero and Marcus Gualtieri for reviewing early drafts of this paper.

References

[1] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp pose detection in point clouds," *Int. J. Robot. Res.*, vol. 36, no. 13-14, pp. 1455–1473, 2017.

[2] M. Gualtieri and R. P. Jr., "Learning 6-dof grasping and pick-place using attention focus," in *2nd Annual Conf. on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, ser. Proc. of Machine Learning Research, vol. 87. PMLR, 2018, pp. 477–486.

[3] A. Mousavian, C. Eppner, and D. Fox, "6-dof graspnet: Variational grasp generation for object manipulation," in *Proc. of the 2019 IEEE Int. Conf. on Computer Vision (ICCV)*, Oct. 2019, pp. 2901–2910.

[4] 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 *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2019.

[5] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *Int. J. Robot. Res.*, vol. 34, no. 4–5, pp. 705–724, Apr. 2015.

[6] 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.

[7] D. Kalashnikov, *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Proc. of The 2nd Conf. on Robot Learning*, vol. 87, 2018, pp. 651–673.

[8] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *Int. J. Robot. Res.*, vol. 37, no. 4-5, pp. 421–436, 2018.

[9] Y. Qin, R. Chen, B. Zhu, M. Song, J. Xu, and H. Su, "S4g: Amodal single-view single-shot se(3) grasp detection in cluttered scenes," in *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, ser. Proceedings of Machine Learning Research, vol. 100. PMLR, 2019, pp. 53–65.

[10] A. ten Pas and R. Platt, "Using geometry to detect grasp poses in 3d point clouds," in *Robotics Research: Volume 1*, 2018, pp. 307–324.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. of the 2014 IEEE Conf. on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[12] R. Girshick, "Fast r-cnn," in *Proc. of the 2015 IEEE Int. Conf. on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28*, Dec. 2015, pp. 91–99.

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," in *ECCV*, vol. 9905, 2016, pp. 21–37.

[15] V. Satish, J. Mahler, and K. Goldberg, "On-policy dataset synthesis for learning robot grasping policies using fully convolutional deep networks," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1357–1364, 2019.

[16] C. Bodnar, A. Li, K. Hausman, P. Pastor, and M. Kalakrishnan, "Quantile qt-opt for risk-aware vision-based robotic grasping," in *Proc. of Robotics: Science and Systems*, 2020.

[17] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. of the 31st Int. Conf. on Neural Information Processing Systems*, Dec. 2017, p. 5105–5114.

[18] A. M. A. Murali, C. Eppner, C.Paxton, and D. Fox, "6-dof grasping for target-driven object manipulation in clutter," in *Int. Conf. on Robotics and Automation (ICRA)*, 2020.

[19] Y. Zhou and K. Hauser, "6dof grasp planning by optimizing a deep learning scoring function," in *Robotics: Science and Systems (RSS) Workshop on Revisiting Contact-Turning a Problem into a Solution*, vol. 2, 2017.

[20] 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 *2018 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2018, pp. 1–9.

[21] M. Van der Merwe, Q. Lu, B. Sundaralingam, M. Matak, and T. Hermans, "Learning continuous 3d reconstructions for geometrically aware grasping," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2020.

[22] Q. Lu, M. Van der Merwe, B. Sundaralingam, and T. Hermans, "Multifingered grasp planning via inference in deep neural networks: Outperforming sampling by learning differentiable models," *IEEE Robot. Automat. Mag.*, vol. 27, no. 2, pp. 55–65, 2020.

[23] B. Wu, I. Akinola, A. Gupta, F. Xu, J. Varley, D. Watkins-Valls, and P. Allen, "Generative attention learning: a "general" framework for high-performance multi-fingered grasping in clutter," *Autonomous Robots*, 2020.

[24] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen, "Shape completion enabled robotic grasping," in *2017 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2017, pp. 2442–2447.

[25] A. Paszke, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[26] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3d: A modern library for 3d data processing," *arXiv preprint arXiv:1801.09847*, 2018.

[27] M. Matl. (2019) Pyrender. [Online]. Available: https://github.com/mmatl/pyrender

[28] V.-D. Nguyen, "Constructing force-closure grasps," *Int. J. Robot. Res.*, vol. 7, no. 3, pp. 3–16, 1988.

[29] W. Wohlkinger, A. Aldoma, R. B. Rusu, and M. Vincze, "3dnet: Large-scale object class recognition from cad models," in *2012 IEEE Int. Conf. on Robotics and Automation*, 2012, pp. 5384–5391.

[30] J. Huang, H. Su, and L. J. Guibas, "Robust watertight manifold surface generation method for shapenet models," *CoRR*, vol. abs/1802.01698, 2018. [Online]. Available: http://arxiv.org/abs/1802.01698

[31] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *2012 IEEE Intl. Conf. on Robotics and Automation*, 2012, pp. 3859–3866.

[32] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," 2009.

[33] K. P. Hawkins, "Analytic inverse kinematics for the universal robotsur-5/ur-10 arms," Georgia Institute of Technology, Tech. Rep., 2013.

[34] R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34, July 2008.

[35] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *Int. J. Robot. Res.*, vol. 33, no. 9, pp. 1251–1270, 2014.

[36] H. Pham and Q. Pham, "A new approach to time-optimal path parameterization based on reachability analysis," *IEEE Trans. Robot. Automat.*, vol. 34, no. 3, pp. 645–659, 2018.