DDGC: Generative Deep Dexterous Grasping in Clutter

Jens Lundell, Francesco Verdoja, and Ville Kyrki

Abstract—**Recent advances in multi-fingered robotic grasping have enabled fast 6-Degrees-of-Freedom (DOF) single object grasping. Multi-finger grasping in cluttered scenes, on the other hand, remains mostly unexplored due to the added difficulty of reasoning over obstacles which greatly increases the computational time to generate high-quality collision-free grasps. In this work, we address such limitations by introducing DDGC, a fast generative multi-finger grasp sampling method that can generate high quality grasps in cluttered scenes from a single RGB-D image. DDGC is built as a network that encodes scene information to produce coarse-to-fine collision-free grasp poses and configurations. We experimentally benchmark DDGC against two state-of-the-art methods on 1200 simulated cluttered scenes and 7 real-world scenes. The results show that DDGC outperforms the baselines in synthesizing high-quality grasps and removing clutter. DDGC is also 4-5 times faster than GraspIt!. This, in turn, opens the door for using multi-finger grasps in practical applications which has so far been limited due to the excessive computation time needed by other methods. Code and videos are available at https://irobotics.aalto.fi/ddgc/.**

Index Terms—**Grasping, deep learning in grasping and manipulation, dexterous manipulation.**

I. Introduction

ROBOTIC grasping of unknown objects in cluttered scenes, such as the one shown in Fig. 1, is challenging because the scene geometry is not fully known. To date, the most influential results of grasp sampling in clutter are obtained with methods that sample grasps for parallel-jaw grippers [1]–[4]. Multi-fingered grasping introduces the added difficulty that it requires concurrent reasoning over grasp quality and collision avoidance with multi-DOF dexterous robotic hands and complex scenes. The benefit, however, of multi-finger grippers compared to parallel-jaw grippers is that they can attain more diverse grasp types. These are especially useful for task-specific grasping [5].

Manuscript received February 24, 2021; accepted June 25, 2021. Date of publication July 13, 2021; date of current version July 26, 2021. This letter was recommended for publication by Associate Editor T. Hermans and Editor H. Liu upon evaluation of the reviewers' comments. This work was supported by the Academy of Finland Strategic Research Council under Grant 314180. *(Corresponding author: Jens Lundell.)*

The authors are with the Intelligent Robotics Group, Department of Electrical Engineering and Automation, School of Electrical Engineering, Aalto University, Espoo 02150, Finland (e-mail: jens.lundell@aalto.fi; francesco.verdoja@aalto.fi; ville.kyrki@aalto.fi).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2021.3096239, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3096239

Current methods for multi-finger grasp generation in cluttered scenes either 1) employ a stochastic search process such as simulated annealing to maximize a grasp quality metric to propose good grasps [6], or 2) train a grasping policy in simulation using deep Reinforcement Learning (RL) to directly output the end pose and finger configuration of the gripper [7]. Although a learned deep grasping policy can attain a high grasp success rates on a diverse set of objects, it requires a simulation setup and typically extensive parameter search to work well. Employing a stochastic search procedure, on the other hand, does not require training any model and as such is directly usable in a grasping pipeline; this approach is however limited to known object models and is computationally expensive, with running times in the order of tens of seconds to minutes.

In this work, we present DDGC, a deep network that can generate a set of collision-free multi-finger grasps on unknown objects in cluttered scenes many times faster than state-of-the-art methods. To achieve this, we train a generative deep coarse-to-fine grasp sampler on purely synthetic data of cluttered scenes. In comparison to similar methods where a grasping policy is trained with RL [7], our method does not need any simulation setup which can be brittle to tune.

DDGC was experimentally validated in simulation and in the real-world against two baselines: Multi-FinGAN [8], a recent state-of-the-art single-object multi-finger grasp sampling method, and the simulated-annealing planner in GraspIt! [6]. The simulation experiments evaluated grasp-quality and sampling speed with a *Barrett hand* on three different datasets containing cluttered scenes with one to four objects. The real-world experiments, on the other hand, evaluated the clearance rate in cluttered scenes containing 4 objects using a physical *Franka Emika Panda* equipped with a *Barrett hand*. The results show that DDGC generates higher quality grasps with a higher clearance rate than both baselines. It can also generate grasps 4-5 times faster than the simulated-annealing planner in GraspIt!, which is a significant speedup.

The main contributions of this work are:

1) a novel generative grasp sampling method that enables fast sampling of multi-finger collision-free grasps in cluttered scenes;
2) new datasets with high-quality grasps for benchmarking multi-finger grasping; and
3) an empirical evaluation of the proposed method against state-of-the-art, presenting, both in simulation and on real hardware, improvements in terms of running time, grasp ranking, and clearance rate.

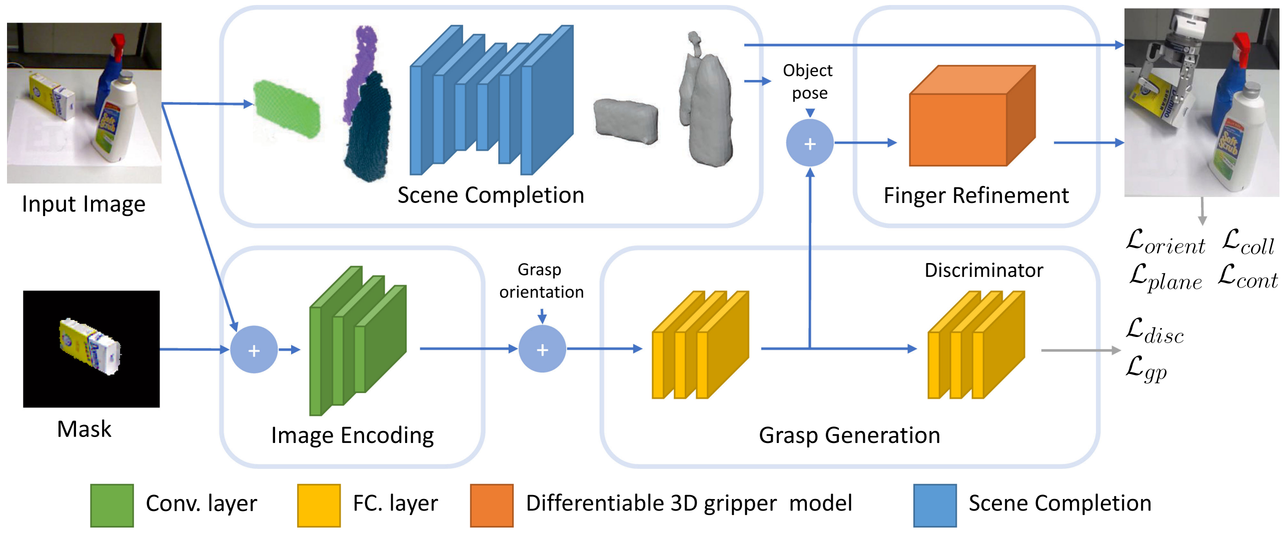

Fig. 1. Using a single RGB-D image of a cluttered scene as input, our proposed generative grasp planner can produce up to 10 collision-free multi-fingered grasps with various grasp types in less than a second.

II. Related Work

To date, the majority of recent research on grasping in clutter has focused on parallel-jaw grippers [1]–[4], [9]–[13], as the kinematic structure of such grippers simplifies the overall problem. Only a few works [7], [14], [15] consider multi-finger hands. Because of this division, we will review multi-finger grasping and grasping in clutter separately.

A. Multi-Finger Grasping

Early work on multi-finger grasping focused on constructing force-closure grasps that are dexterous, in equilibrium, stable, and exhibit a certain dynamic behavior by casting the problem as an optimization problem over one or all of these criteria [15], [16]. However, one of the core issues of such methods are that they assume precise and known geometric and physical models of the objects to grasp [17]. Moreover, analytical methods are typically computationally expensive and as such generate only a few grasps.

To lift some of the requirements of analytical methods and to make grasp sampling faster, researchers resorted to data-driven methods. Early data-driven methods are based heavily on the development and release of GraspIt! in 2004 [6] and focused mainly on the issue of sampling grasps from the infinite space of solutions [18]–[22]. Out of these works, the *eigengrasp* formulation by Ciocarlie and Allen [19]—where grasp planning is performed in a hand posture subspace of highly reduced dimensionality—has proven effective as it is still used today as a grasp sampling component in many grasping pipelines [23]–[25]. Although these methods greatly sped up grasp sampling in comparison to analytical methods, they still operate in the order of tens of seconds to minutes and require full object models.

To enable data-driven grasp sampling methods to work on unknown objects, recent works have used deep learning to shape complete unknown objects and then plan grasps on the completed shapes [23]–[26]. These methods generalize across many different objects but still rely on an inherently slow sampling process. Therefore, other deep-learning methods have focused on learning the actual grasp sampler either from raw sensor inputs [5], [7], [27]–[29] or shape completed models [8]. The limitation of all of these methods, with the exception of [7], is that they are proposed for and evaluated on single object grasping. In comparison, the method presented here can generate diverse high-quality grasps fast on both single objects and objects in clutter.

B. Grasping in Clutter

Grasping in clutter poses additional constraints on grasp sampling as it typically requires grasps to not collide with obstacles. Therefore, to simplify the grasp planning problem in clutter much work resorts to planning grasps with simpler parallel-jaw grippers opposed to more complex dexterous hands [1]–[4], [9]–[13]. Murali *et al.* [4] split the concept of grasping in clutter into the following subcategories: bin-picking vs. structured clutter, planar vs. spatial grasping in clutter, model-based vs. model-free, and target-agnostic vs. target-driven.

In bin-picking, objects are located in a pile and are often small and light. Methods that target bin-picking [1]–[3], [9], [10] benefit from the pile structure as objects have more stable equilibriuma and as such collisions with obstacles do not jeopardize the pile structure. In comparison, methods that target grasping in structured clutter [4], [13]—where objects are mostly larger, heavier, and packed together—need to avoid all collisions as unintended contact between the robot and objects can easily tip objects over and thus change the scene structure. Our approach calculates if a grasp is in collisions by checking if the hand mesh is colliding, while [4] requires another trained network to predict potential collisions.

In cluttered scenes, the most successful methods have resorted to the simpler 4-DOF planar top-down grasps [1]–[3], [9] instead of full 6-DOF spatial grasps [4], [7], [13], [24]. While 4-DOF grasps are simpler, the restriction on the arm motion they imply hinders the robot's ability to grasp specific objects [4], [7], [13], especially with structured clutter. Because of these reasons, we developed our method to predict full 6-DOF grasps.

Model-based grasping in clutter assumes known object models and as such simplifies the planning problem [15]. Approaches that target model-free grasping in clutter where objects are unknown either plan grasps directly on raw sensor inputs [1]–[4], [7], [9], [13] or first estimate the shape of the unknown objects and then plan grasps [10], [14]. In this work, we estimate the shape of all the objects, as it allows us to plan grasps that are closer to the surface of the objects.

With target-driven grasp sampling methods [4] it is possible to specify the object to grasp opposed to target-free methods [1]–[3], [7], [9], [10], [13] which oftentimes simply chooses the object it can attain the best grasp on. Our method is target-driven as it segments objects and plan object-specific grasps using these segments.

In a nutshell, the work presented here falls into the category of target-driven model-free multi-finger spatial grasping in scenes with structured clutter. A similar multi-finger grasping in clutter approach was proposed in [7] but for a different use-case: target-agnostic bin-picking. That method used RL to train a deep grasping policy in simulation which transferred seamlessly to the real world, attaining a high grasp success rate on a diverse set of objects. Our grasp sampler is also trained on purely synthetic data but, in contrast to [7], does not require an extensive simulation setup to train.

DDGC goes beyond prior work on grasping in clutter with parallel-jaw grippers [1]–[4], [9]–[13] by using a more complex dexterous gripper and reasoning over more DOF, generalizing to a more complicated problem. In comparison to previous work on grasping in clutter with dexterous hands [7], [14], [15], our work goes beyond those by either being considerably faster (8 sec. vs 16 sec [14]) or allowing to select the object to grasp [7], [15].

III. Problem Formulation

In this work, we address the problem of grasping unknown objects in structured clutter with a multi-finger robotic gripper. We realize this by predicting a 6D gripper pose $\mathbf{p}$ and a hand joint configuration $\mathbf{q}$ for which the grasp does not collide with any obstacle but still has several contact points with the target object to ensure a robust grasp. More formally, we train a parametric model $\mathcal{M}_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$ that takes as input an RGB-D image $\mathbf{I}$ and produces a 6D gripper pose $\mathbf{p}$, and a hand joint configuration $\mathbf{q}$:

$$\mathcal{M}_{\boldsymbol{\theta}} : \mathbf{I} \rightarrow (\mathbf{p}, \mathbf{q}) \,.$$

The poses $\mathbf{p}$ of all grasps are in the target object's center of reference.

We further constrain the target hand joint configuration $\mathbf{q}$ to leave a small clearance between the fingers and the target object, and the gripper is closed until contact before attempting to lift an object. This is done to limit the impact of sensing uncertainties in the real-world as we avoid the need to generate precise configurations that touch the surface of the object.

IV. Method

To generate high-quality collision free grasps in cluttered scenes, we introduce the model shown in Fig. 1 which is inspired from prior work on human-hand grasp generation [30] and our own work on predicting multi-finger grasps on single objects [8]. This network consists of sub-modules that were used for single object grasping along with novel ones that were essential to adapt the architecture to grasping in clutter. Therefore, for completeness, we will next briefly overview the entire model and then describe the novel technical contributions in detail.

A. Network Overview

The network in Fig. 1 is split into 5 sub-modules: Scene Completion, Image Encoding, Grasp Generation, Discriminator and Finger Refinement. Out of these Image Encoding and Grasp Generation are novel while the others have been used in prior work [8], [10], [30].

Scene Completion refers to the process of producing a full 3-D model of the scene by segmenting the scene and shape completing each object [10]. The method is agnostic to the segmentation algorithm. For shape completion, we use a pre-trained fully convolutional hour-glass shaped Deep Neural Network (DNN) that operates on voxelized point-clouds as input and outputs a completed voxel grid of the object. The object is post-processed into a mesh by first merging it with the input point-cloud and then running the marching cube algorithm [31]. The shape completed objects are later used for grasp generation, finger refinement, and collision detection.

The *Image Encoding* module, further detailed together with the Grasp Generation module in the next section, produces an encoding $\mathrm{E}(\mathbf{I})$ of the geometry of the scene from an RGB image $\mathbf{I}$ using a Convolutional Neural Network (CNN). This encoding, together with an input grasp orientation $\mathbf{r}_0$, is then passed through the fully-connected *Grasp Generation* network to generate a coarse gripper joint configuration $\mathbf{q}_c$ and a 6D coarse object-centric grasp pose $\mathbf{p}_c = (\mathbf{t}_c, \mathbf{r}_c)$ where $\mathbf{t}_c$ represents a translation vector and $\mathbf{r}_c$ rotation as normalized axis angles. The coarse grasp pose is then refined into global coordinates $\mathbf{p}^* = (\mathbf{t}_c + \mathbf{t}_0, \mathbf{r}_c + \mathbf{r}_0) = (\mathbf{t}, \mathbf{r})$ by adding the center of the mesh $\mathbf{t}_0$ to the coarse translation vector $\mathbf{t}_c$ and the input rotation $\mathbf{r}_0$ to the coarse rotation $\mathbf{r}_c$.

When training DDGC, the input rotation $\mathbf{r}_0$ is set to a grasp rotation from the dataset with added zero-mean Gaussian noise while, at test-time, the input rotations are randomly sampled from around the object. Similarly, the center of the target object $\mathbf{t}_0$ is known during training because we are operating on known meshes, while during the physical experiments we use the center of the shape-completed mesh.

In this work, the grasp configuration $\mathbf{q}_c$ represents the 7-DOF of a Barrett hand shown in Fig. 6. However, the Grasp Generation network is only tasked to output the finger spread of the coarse gripper configuration $\mathbf{q}_c$ and leaves the rest of the DOF in a default zero position. The reason for doing this is that we use the Finger Refinement layer to refine the coarse gripper joint configuration $\mathbf{q}_c$ to be close to the target object.

The *Finger Refinement* layer is tasked to rotate each finger of the gripper until it touches the object. It is represented as the forward kinematics of the gripper and as such is fully differentiable and does not include any additional trainable parameters. Given an input 6D pose $\mathbf{p}^*$ and coarse hand configuration $\mathbf{q}_c$ it produces a refined configuration $\mathbf{q}^* = \mathbf{q}_c + \Delta\mathbf{q}$ by rotating each articulated finger $\Delta\mathbf{q}$ radians until the distance between the finger and the object mesh is below a predefined threshold t_d which we set to 1 cm.

To ensure that the distribution of generated grasps is close to the training distribution we add a Wasserstein discriminator D [32] trained with the gradient penalty [33]. Given the grasp generator G, the objective function to minimize is

$$\begin{aligned}\mathcal{L}_{disc} &= \mathbb{E}\left[\mathrm{D}(\mathrm{G}(\mathrm{E}(\mathbf{I}), \mathbf{r}_0))\right] - \mathbb{E}\left[\mathrm{D}(\widehat{\mathbf{q}}, \widehat{\mathbf{t}}, \widehat{\mathbf{r}})\right],\\ \mathcal{L}_{gp} &= \mathbb{E}\left[\left(\|\nabla_{\widetilde{\mathbf{q}}, \widetilde{\mathbf{T}}, \widetilde{\mathbf{r}}}\mathrm{D}(\widetilde{\mathbf{q}}, \widetilde{\mathbf{t}}, \widetilde{\mathbf{r}})\|_2 - 1\right)^2\right],\end{aligned} \tag{1}$$

where $\widehat{\mathbf{q}}$, $\widehat{\mathbf{t}}$, and $\widehat{\mathbf{r}}$ are samples from the training dataset and $\widetilde{\mathbf{q}}$, $\widetilde{\mathbf{t}}$, and $\widetilde{\mathbf{r}}$ are linear interpolations between predictions and those samples.

B. Multi-Finger Deep Grasping in Clutter

As mentioned in the previous section the novel sub-modules in this work are the *Image Encoding* and *Grasp Generation* networks. The use of these two networks were, in part, explored in our previous work on single object robotic grasping [8]. However, the architectures we proposed there cannot handle the complexity of cluttered scenes because of the following reasons:

1) the image encoding network operates on input RGB images where everything except the target object is masked out,
2) the grasp generation network requires a coarse estimate of the input grasp configuration.

In a cluttered scene, a consequence of encoding only image information about the target object is that information about obstacles is ignored and as such cannot prime the grasp sampler to generate collision-free grasps. To circumvent this issue, we kept the original input RGB image unmasked and instead added an additional channel with the target object mask. When training the network we have ground-truth masks of all objects, while at run-time we produce the masks from a segmented point-cloud of the scene. An additional benefit of adding the mask of the object is that we effectively make the complete grasp sampling network target-driven, that is we can choose which object to grasp by selecting the corresponding mask.

In our previous work on dexterous multi-finger grasping [8], the grasp generation network also produced residual grasp joints that were added to initial ones given from a classification network. The classification network classified grasps according to one of the 33 grasp taxonomies listed in [34]. However, this required both to train an additional network and to label all grasps according to their taxonomy which is time-consuming as it requires human domain knowledge to correctly label the grasps.

C. Loss Functions

An empirical finding in our previous work on multi-finger grasp sampling of single objects [8] is that the discriminator helps in producing realistic-looking grasps but it alone cannot guide the training enough. Therefore, we add the following complementary loss functions: a contact loss $\mathcal{L}_{cont}$, a collision loss $\mathcal{L}_{coll}$, an orientation loss $\mathcal{L}_{orient}$, and a plane loss $\mathcal{L}_{plane}$. Out of these $\mathcal{L}_{cont}$, $\mathcal{L}_{coll}$, and $\mathcal{L}_{orient}$ were used in our previous work [8], while $\mathcal{L}_{plane}$ was originally introduced in [30].

For grasps to be successful they need to have multiple contact points with the object. To encourage such behavior we added the contact loss defined as

$$\mathcal{L}_{cont} = \frac{1}{|V_{cont}|}\sum_{v\in V_{cont}} \min_k \|v, O_k\|_2, \tag{2}$$

where O_k are the object vertices. V_{cont} is the subset of vertices on the hand that are often in contact with the target object, *i.e.*, vertices that were closer than 5 mm to the object in at least 8% of the grasps in the dataset. Vertices in V_{cont} are mainly located on the gripper's fingertips and palm.

In our previous work [8], we identified having a loss that encourages grasps to point towards the object to grasp as helping the training. This is realized by encouraging the approach vector of the gripper $\hat{\mathbf{a}}$ to point in the same direction as the vector $\hat{\mathbf{o}}$ connecting the hand to the object's center:

$$\mathcal{L}_{orient} = 1 - \hat{\mathbf{a}}^\top\hat{\mathbf{o}}. \tag{3}$$

To penalize collisions between grasps and the environment which, in scenes with structured clutter is of uttermost importance, we added the following loss

$$\mathcal{L}_{coll} = \frac{1}{|V_i|}\sum_{j}^{|O|}\sum_{\mathbf{v}\in V_i} \mathrm{A}_{\mathbf{v}} \min_k \|\mathbf{v}, O_k^j\|_2. \tag{4}$$

This loss penalizes the distance between all vertices of a grasp that lie inside an object $\mathbf{v} \in V_i$ and their closest surface points to object O^j, where $|O|$ is the number of objects present in the scene. We generate surface points of every object by uniformly sampling k points from their mesh. The weight A_v is the average area of all incident faces to a vertex v, which is important when working with non-uniform gripper tessellations [8].

Finally, we also need to encourage grasps to stay above the plane the objects are resting on. By representing the plane as a point $\mathbf{p}_t$ on its surface and its normal $\hat{\mathbf{n}}_t$, we can penalize all vertices $\mathbf{v}$ on the gripper mesh that are located under the table with

$$\mathcal{L}_{plane} = \sum_{\mathbf{v}\in V} \min\left(0, (\mathbf{v} - \mathbf{p}_t)^{\mathrm{T}} \cdot \hat{\mathbf{n}}_t\right). \tag{5}$$

Equation (5) is only negative when the normal vector $\hat{\mathbf{n}}_t$ and the vector connecting the point on the plane $\mathbf{p}_t$ and a vertex $\mathbf{v}$ points in different directions indicating that the vertex $\mathbf{v}$ is located below the table.

The final loss is a weighted linear combination of the six individual losses $\mathcal{L} = w_{disc}\mathcal{L}_{disc} + w_{gp}\mathcal{L}_{gp} + w_{cont}\mathcal{L}_{cont} + w_{coll}\mathcal{L}_{coll} + w_{orient}\mathcal{L}_{orient} + w_{plane}\mathcal{L}_{plane}$. The network is trained end-to-end.

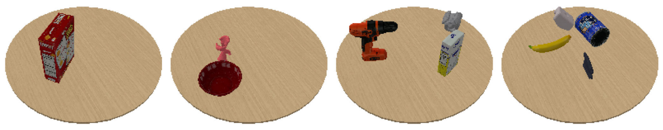

Fig. 2. Example scenes from our data-set with one up to four objects.

D. Implementation Details

The network was implemented in PyTorch 1.5.1. The model was trained on scenes containing 1, 2, 3, or 4 objects randomly placed on a table surface; some examples are shown in Fig. 2 and details on how we generated them are given in Section V. The images were resized to 256×256. We trained our networks with a learning rate of 10^{-4} and a batch size of 64. The image encoder was fine-tuned on weights from a pre-trained ResNet-50 while the weights for the shape completion network were not trained at all and instead set to the same as in [24]. The generator was trained once every 5 forward passes to improve the relative quality of the discriminator. We trained the networks for 12 000 epochs, reducing the learning rate linearly for the last 4000 epochs.

The weights of the loss functions were set to the same values as in [8] except the plane loss which was set to the same value as reported in [30]. During training, we did not witness any detrimental competition between the losses. Instead, they reduced in tandem.

V. DDGC Dataset

Training deep networks, such as DDGC, typically requires a lot of diverse training data to obtain good performance. Moreover, the usual assumption is that train and test data should originate from a similar underlying distribution. This, in the case of real-world robotics experiments, means we need to gather training data from real robot interactions, a procedure that is both time-intensive and wears the robotic hardware. Many prior works have, however, empirically shown that training grasp generation networks on simulated data alone opposed to on real-world data works well even for real-world grasping [2], [8], [35].

Based on these findings, we train DDGC on only simulated data of cluttered scenes. To simulate cluttered scenes we use the physics simulator PyBullet [36]. A scene is generated by first selecting the number of objects to place in it and then, for one object at a time, we randomly generate a position on the table that is not in collision with any other object and place it there in one of its stable resting poses[1]. If objects have predefined textures we apply them, otherwise, we randomly generate texture maps. Finally, we render the RGB image and the object masks from a $45°$ viewing angle and store them together with the object poses.

Training DDGC requires not only RGB and masked images but also multi-finger grasps on each object. Multi-finger grasps were first generated separately for each object in their canonical pose using GraspIt!. To avoid bias and to generate a dense set of grasps we uniformly sampled grasps around the object with two different distances to the object (2 and 8 cm) and three different finger spreads ($0°$, $45°$, and $90°$). These object-centric grasps were then transferred to the cluttered scenes by applying the same random transform that the objects went through when placing them in the scene. A grasp was marked successful if it was collision-free.

We generated one training set of cluttered scenes on 18 objects from the YCB object set [37] and 120 objects from the EGAD! training set [38]. We also generated three validation sets: one on the 49 validation objects in the EGAD! validation set, one on the 152 objects in the KIT object model database [39], and one with both EGAD! validation set and KIT object model database. To generate datasets with a varying amount of clutter, we generated 100 random scenes with 1, 2, 3 or 4 objects in it, totaling 400 scenes per dataset.

VI. Experiments and Results

The two main questions we wanted to answer in the experiments were:

1) Is DDGC able to generate high quality grasps in scenes with clutter?
2) Is our generative grasp sampler, which is purely trained on synthetic data, able to transfer to real objects?

In order to provide justified answers to these questions, we conducted two separate experiments. In the first experiment (Section VI-A) we evaluate grasp quality and hand-object interpenetration in simulation. In the second experiment (Section VI-B) we evaluate the clearance rate in cluttered scenes on real hardware.

In all our experiments, we benchmark our approach against Multi-FinGAN [8] and the simulated-annealing planner in GraspIt! [6] which is the only planner that can do multi-finger target-driven grasping in clutter. We let the simulated-annealing planner run for 75 000 steps. To run GraspIt! with clutter, the target object was set as a graspable object while the rest of the objects were set as obstacles. We also included the table the objects were resting on as an obstacle.

In both experiments, DDGC and Multi-FinGAN generated 100 grasps per object. Grasps that were in collision with the table or any other objects were removed. As we a-priori knew that all objects rest on a flat surface we only sampled, for both DDGC and Multi-FinGAN, input rotations $\mathbf{r}_0$ that rotated the gripper's approach direction to point towards the negative normal direction of the supporting surface. This heuristic effectively removed input rotations that correspond to grasps coming from below the table.

A. Grasping in Simulation

Here we present quantitative results of grasps generated by DDGC, Multi-FinGAN, and the simulated-annealing planner in GraspIt! on the three different held out validation sets: EGAD! val, KIT, and EGAD! val + KIT. To quantify the quality of a grasp we calculated both its intersection with all objects and the well known ϵ-quality metric [40] which represents the radius of

[1]The stable resting poses were calculated using trimesh (https://github.com/mikedh/trimesh github.com/mikedh/trimesh)

TABLE I
SIMULATION EXPERIMENT RESULTS. ↑: HIGHER THE BETTER; ↓: LOWER THE BETTER

	GraspIt!			Multi-FinGAN			DDGC		
	EGAD! val	KIT	EGAD! val + KIT	EGAD! val	KIT	EGAD! val + KIT	EGAD! val	KIT	EGAD! val + KIT
Avg. ϵ-quality over 10 grasps ↑	0.21	0.20	0.20	0.32	0.38	0.35	**0.74**	**0.73**	**0.72**
Avg. interpenetration over 10 grasps (cm^3) ↓	**2.92**	4.76	**3.63**	4.12	5.92	5.05	4.00	**3.30**	3.64
Grasp Sampling for 10 grasps (sec.) ↓	44.09	39.97	35.75	**9.10**	**8.80**	**9.00**	9.60	9.40	9.40

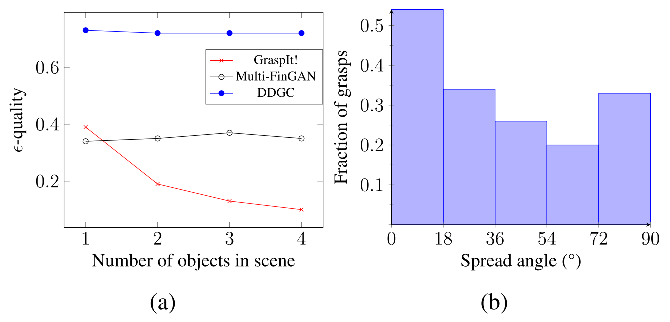

Fig. 3. (a) shows that DDGC constantly finds higher-quality grasps than both GraspIt! and Multi-FinGAN. (b) shows a histograms of the finger spread of grasps generated with DDGC.

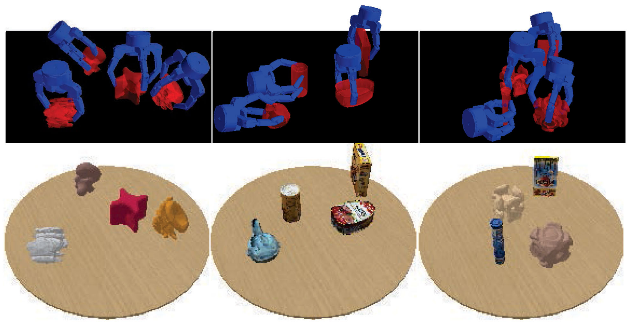

Fig. 4. Some example grasps shown in the top row proposed by DDGC in simulated scenes using the RGB input shown on the bottom row. The black background represents the table surface (best viewed in color).

the largest 6D ball centered at the origin that can be enclosed by the wrench space's convex hull.

For all methods, we rank the grasps according to their ϵ-quality and report the average performance on the 10 top-scoring grasps. We also report how the metrics vary across increasing amount of clutter.

The results are presented in Table I and Fig. 3. What is immediately evident is that DDGC generates significantly higher quality grasps than both Multi-FinGAN and GraspIt!. DDGC is also over 4 times faster than GraspIt!. As shown in Fig. 3(a), with increasing clutter the performance of GraspIt! deteriorates heavily much due to the optimization falls into local minimas between objects while the performance of DDGC and Multi-FinGAN is constant. The main reason Multi-FinGAN generates lower ϵ-quality grasps than DDGC is because 27.25% of its grasps are located below the table compared to only 1.25% by DDGC. Some example grasps using DDGC are shown in Fig. 4.

Fig. 3(b) shows a histogram over the finger spread of the *Barrett hand* produced by DDGC. Based on this figure, we can conclude that DDGC produces diverse grasps, but seems to slightly favor grasps with no spread. One reason zero-spread grasps, which are similar to parallel-jaw grasps, are favored over others is because those grasps have smaller occupancy footprint due to the kinematic structure of the *Barrett hand*.

In summary, the results from the simulation experiments show that DDGC is able to quickly sample high-quality collision-free grasps in cluttered scenes. Next, we will investigate its performance on real robotic hardware.

B. Physical Grasping

As a final experiment we studied the clearance rate of DDGC, Multi-FinGAN, and GraspIt! on real robotic hardware. The clearance rate objective measures how many objects a grasping method can remove in a cluttered scene, given a predefined grasping budget. The setup consists of a *Franka Emika Panda* equipped with a *Barrett hand* and a Kinect 360° camera to capture the input RGB-D images. The camera views the scene at a 55° viewing angle. For extrinsic calibration, we used an Aruco marker [41]. We added the shape-completed objects to the planning scene to ensure collision-free motion planning.

As shown in Fig. 1 our method requires an input RGB image along with the individual object masks, and segmented point-clouds of each object as seen by the depth image. To segment each object we first subtracted the background and the table and then extracted each segment with the Euclidean Cluster Extraction method in PCL[2]. To generate the object masks, we transformed each of the point-cloud-segments to a binary depth-image and multiplied these with the input RGB image. The object masks were also used to generate grasps for Multi-FinGAN. Fig. 1 shows examples of an input RGB image, object mask, and segmented point-cloud. Each segmented point-cloud was shape-completed into a mesh and the average of its vertices was used as the center of the object which was subsequently added to the translation produced by the Grasp Generation network.

DDGC, Multi-FinGAN, and GraspIt! were benchmarked once on each of the seven different scenes shown in Fig. 5. Each scene contained four objects of varying size, shape, and softness. The objects were manually placed to create scenes where objects occluded each other. We chose a grasping budget

[2]https://pcl.readthedocs.io/en/latest/cluster_extraction.html pcl.readthedocs.io/en/latest/cluster_extraction.html

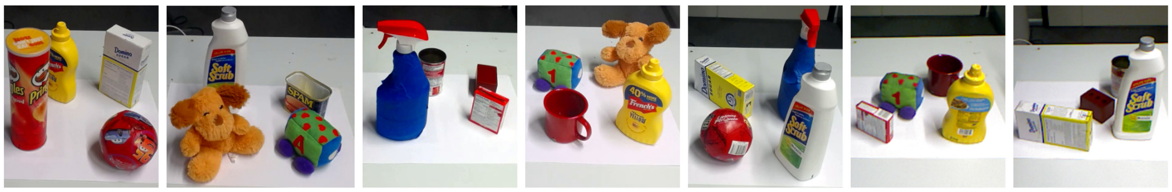

Fig. 5. Scenes used for testing.

TABLE II
REAL HARDWARE EXPERIMENT RESULTS

	GraspIt!	Multi-FinGAN	DDGC
Average clearance rate (%) ↑	60.71	57.15	**71.43**
Average grasp success rate (%) ↑	32.72	34.90	**40.00**
Grasp Sampling for 10 grasps (sec.) ↓	40.40	10.70	**8.10**

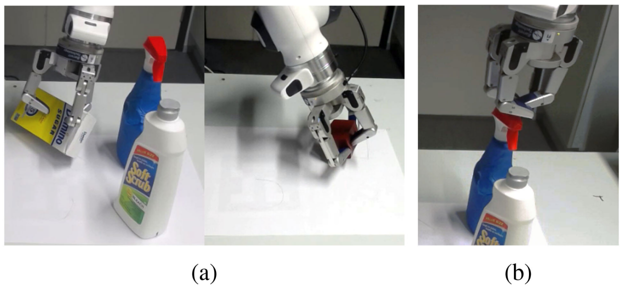

Fig. 6. Some successful (a) and failed (b) grasps.

of eight grasps which is twice the number of objects in the scene, a heuristic that has been used in similar works [4], [10]. A grasp was considered a success if the robot grasped the object and moved it to the start position without dropping it. If an object in the scene was accidentally moved or dropped, it was manually reset to its original pose.

The experimental results (Table II) indicate that DDGC reaches a higher average clearance rate and grasp success rate than both Multi-FinGAN and GraspIt!. Again, DDGC is significantly faster than GraspIt!, producing and evaluating grasps roughly 5 times as fast. The performance gain of DDGC compared to Multi-FinGAN indicates that encoding information of the complete scene is useful for generating successful grasps in clutter, a finding that was also reported in [13]. For all methods the scene completion took, on average, 4 seconds. Both DDGC and Multi-FinGAN were robust to noisy shape-completions and object masks.

Examples of successful grasps generated with DDGC are shown in Fig. 6(a). These examples show that DDGC is able to produce a diverse set of grasp poses and configurations compatible with the clutter in the scene.

C. Discussion

Together, all results show that DDGC is able to generate high-quality collision-free grasps in cluttered simulated and real-world scenes. When comparing it to GraspIt!, DDGC achieves better performance, in a fraction of the time. The difference in performance is particularly significant in simulation, where DDGC achieves significantly higher ϵ-quality than GraspIt! which explicitly optimizes grasps on such a metric. One potential reason for this difference is that the Finger Refinement layer gracefully refines the fingers to establish multiple contact points with the object resulting in higher quality grasps. This difference was not as evident in the real-world experiment, indicating that low ϵ-qualities do not necessarily translate to unsuccessful grasps.

It is especially noteworthy that although DDGC is trained purely on simulated data it generalized seamlessly to real-world scenes without any fine-tuning. We hypothesize that this level of generalization is due to the grasps being originally generated in the object's reference frame, which correspond to the simulated data, and only later transformed to world coordinates by adding the center of the mesh to the pose. This effectively makes the network translation invariant and it can therefore focus on generating high-quality collision-free grasps no matter the scale.

Nevertheless, DDGC also produced some catastrophically poor grasps. One example of such a grasp is shown in Fig. 6(b). Here the generated grasp was a precise pinch grasp where only the fingertips should touch the object, but due to inaccurate extrinsic calibration and low friction between the *Barrett hand* and the target object such grasps typically failed. In other failure cases, the gripper completely missed the object. One potential reason such grasps scored high is that they originally focused on, *e.g.*, a corner of the object and due to imperfect extrinsic calibration the grasp simply missed the object. Both of these issues could be addressed by priming DDGC to produce more power type grasps.

Although DDGC is 4-5 times faster than GraspIt!, sampling 100 grasps still takes several seconds. The main reason for such a long sampling time stems from the expensive collision calculation in Eq. (4) which accounts for more than 85% of the total computational time. Nevertheless, sub-second sampling is still possible for 10 or fewer grasps.

VII. CONCLUSION

We presented DDGC, a generative deep network that produces high-quality 6-DOF dexterous grasps in structured clutter in a matter of seconds. Grasping in clutter with dexterous multi-finger hands is especially difficult as the added occupancy footprint of such grippers makes it non-trivial to generate collision-free grasps. DDGC overcomes such difficulties by combining scene-completion, scene-encoding, and a fully differentiable forward kinematics layer to generate dexterous collision-free grasps. We compared DDGC to Multi-FinGAN and GraspIt! in both simulation and real-world grasping and the results show that DDGC produces higher quality grasps than both of these and is over 4-5 times faster than GraspIt!.

Despite the impressive results, there is still room for improvements. In terms of sampling speed, DDGC could be sped up significantly if instead of doing explicit collision checking, it would rather use a separate network for collision detection as in [4]. Another limiting factor is that the high-quality grasps produced by DDGC do not always translate to successful real grasps. Instead of explicitly calculating the quality metric, a critique network to score multi-finger grasps is probably a good solution but this is still a more open problem than on parallel-jaw grasps [1], [35].

In conclusion, the work presented here shows that we can quickly generate high-quality collision-free multi-finger grasps in clutter. This, in turn, enables the use of multi-finger grasps in practical applications which has so far been limited due to the excessive computation time needed by other methods.

Acknowledgment

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

[1] J. Mahler *et al.*, “Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics,” 2017, *arXiv:1703.09312*.

[2] D. Morrison, P. Corke, and J. Leitner, “Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach,” 2018, *arXiv:1804.05172*.

[3] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, “Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection,” 2016, *arXiv:1603.02199 [cs]*.

[4] A. Murali, A. Mousavian, C. Eppner, C. Paxton, and D. Fox, “6-dof grasping for target-driven object manipulation in clutter,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 6232–6238.

[5] M. Kokic, J. A. Stork, J. A. Haustein, and D. Kragic, “Affordance detection for task-specific grasping using deep learning,” in *Proc. IEEE-RAS 17th Int. Conf. Humanoid Robot.*, 2017, pp. 91–98.

[6] A. T. Miller and P. K. Allen, “Graspit! a versatile simulator for robotic grasping,” *IEEE Robot. Automat. Mag.*, vol. 11, no. 4, pp. 110–122, Dec. 2004.

[7] B. Wu *et al.*, “Generative attention learning: A “general” framework for high-performance multi-fingered grasping in clutter,” *Auton. Robots*, vol. 44, pp. 971–990, 2020.

[8] J. Lundell *et al.*, “Multi-fingan: Generative coarse-to-fine sampling of multi-finger grasps,” 2020, *arXiv:2012.09696*.

[9] V. Satish, J. Mahler, and K. Goldberg, “On-policy dataset synthesis for learning robot grasping policies using fully convolutional deep networks,” *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 1357–1364, Apr. 2019.

[10] J. Lundell, F. Verdoja, and V. Kyrki, “Beyond top-grasps through scene completion,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 545–551.

[11] 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.

[12] M. Gualtieri, A. Ten Pas, K. Saenko, and R. Platt, “High precision grasp pose detection in dense clutter,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 598–605.

[13] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox, “Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes,” 2021, *arXiv:2103.14127*.

[14] J. Varley, J. Weisz, J. Weiss, and P. Allen, “Generating multi-fingered robotic grasps via deep learning,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 4415–4420.

[15] D. Berenson and S. S. Srinivasa, “Grasp synthesis in cluttered environments for dexterous hands,” in *Proc. IEEE 8th IEEE-RAS Int. Conf. Humanoid Robots*, 2008, pp. 189–196.

[16] A. Sahbani, S. El-Khoury, and P. Bidaud, “An overview of 3D object grasp synthesis algorithms,” *Robot. Auton. Syst.*, vol. 60, no. 3, pp. 326–336, 2012.

[17] J. Bohg, A. Morales, T. Asfour, and D. Kragic, “Data-driven grasp synthesis - a survey,” *IEEE Trans. Robot.*, vol. 30, no. 2, pp. 289–309, Apr. 2014.

[18] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, “Automatic grasp planning using shape primitives,” in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 2, 2003, pp. 1824–1829.

[19] M. T. Ciocarlie and P. K. Allen, “Hand posture subspaces for dexterous robotic grasping,” *Int. J. Robot. Res.*, vol. 28, no. 7, pp. 851–867, 2009.

[20] C. Borst, M. Fischer, and G. Hirzinger, “Grasping the dice by dicing the grasp,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 4, 2003, pp. 3692–3697.

[21] C. Goldfeder, P. K. Allen, C. Lackner, and R. Pelossof, “Grasp planning via decomposition trees,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2007, pp. 4679–4684.

[22] R. Pelossof, A. Miller, P. Allen, and T. Jebara, “An svm learning approach to robotic grasping,” in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 4, 2004, 2004, pp. 3512–3518.

[23] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen, “Shape completion enabled robotic grasping,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 2442–2447.

[24] J. Lundell, F. Verdoja, and V. Kyrki, “Robust grasp planning over uncertain shape completions,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*Macau, China: IEEE, 2019, pp. 1526–1532.

[25] D. Watkins-Valls, J. Varley, and P. Allen, “Multi-modal geometric learning for grasping and manipulation,” 2018, *arXiv:1803.07671*.

[26] W. Agnew *et al.*, “Amodal 3D reconstruction for robotic manipulation via stability and connectivity,” 2020, *arXiv:2009.13146*.

[27] L. Shao *et al.*, “Unigrasp: Learning a unified model to grasp with multifingered robotic hands,” *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 2286–2293, Apr. 2020.

[28] U. R. Aktas, C. Zhao, M. Kopicki, A. Leonardis, and J. L. Wyatt, “Deep dexterous grasping of novel objects from a single view,” 2019, *arXiv:1908.04293*.

[29] Q. Lu, M. Van der Merwe, and T. Hermans, “Multi-fingered active grasp learning,” 2020, *arXiv:2006.05264*.

[30] E. Corona, A. Pumarola, G. Alenya, F. Moreno-Noguer, and G. Rogez, “Ganhand: Predicting human grasp affordances in multi-object scenes,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 5031–5041.

[31] W. E. Lorensen and H. E. Cline, “Marching cubes: A high resolution 3D surface construction algorithm,” in *Proc. 14th Annu. Conf. Comput. Graph. Interactive Techn.,* New York, NY, USA: ACM, 1987, pp. 163–169. [Online]. Available: http://doi.acm.org/10.1145/37401.37422

[32] S. M. Arjovsky and L. Bottou, “Wasserstein generative adversarial networks,” in *Proc. 34 th Int. Conf. Mach. Learn.*, Sydney, Australia, 2017, pp. 214–223.

[33] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of wasserstein gans,” in Proc. *Adv. Neural Inf. Process. Syst.*, 2017, pp. 5767–5777.

[34] T. Feix, J. Romero, H.-B. Schmiedmayer, A. M. Dollar, and D. Kragic, “The grasp taxonomy of human grasp types,” *IEEE Trans. Hum.-Mach. Syst.*, vol. 46, no. 1, pp. 66–77, Feb. 2016.

[35] A. Mousavian, C. Eppner, and D. Fox, “6-dof graspnet: Variational grasp generation for object manipulation,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 2901–2910.

[36] E. Coumans and Y. Bai, “Pybullet,” 2016, [Online]. Available: http: //pybullet.org

[37] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, “The ycb object and model set: Towards common benchmarks for manipulation research,” in *Proc. Int. Conf. Adv. Robot.* IEEE, 2015, pp. 510–517.

[38] D. Morrison, P. Corke, and J. Leitner, “Egad! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation,” *IEEE Robot. Automat. Lett.*, vol. 5, no. 3, pp. 4368–4375, 2020.

[39] A. Kasper, Z. Xue, and R. Dillmann, “The kit object models database: An object model database for object recognition, localization and manipulation in service robotics,” *Int. J. Robot. Res.*, vol. 31, no. 8, pp. 927–934, 2012.

[40] A. T. Miller and P. K. Allen, “Examples of 3D grasp quality computations,” in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 2, 1999, pp. 1240–1246.

[41] A. T. Miller, and P. K. Allen, “Examples of 3D grasp quality computations,” in *Proc. Int. Conf. Robot. Autom.*, vol. 2, 1999, pp. 1240–1246.