2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

GraspGen: A Diffusion-based Framework for 6-DOF Grasping with On-Generator Training

Adithyavairavan Murali Balakumar Sundaralingam Yu-Wei Chao Wentao Yuan* Jun Yamada*
Mark Carlson Fabio Ramos Stan Birchfield Dieter Fox* Clemens Eppner*
NVIDIA

https://graspgen.github.io

***Abstract*— Grasping is a fundamental robot skill, yet despite significant research advancements, learning-based 6-DOF grasping approaches are still not turnkey and struggle to generalize across different embodiments and in-the-wild settings. We build upon the recent success on modeling the object-centric grasp generation process as an iterative diffusion process. Our proposed framework, *GraspGen*, consists of a Diffusion-Transformer architecture that enhances grasp generation, paired with an efficient discriminator to score and filter sampled grasps. We introduce a novel and performant on-generator training recipe for the discriminator. To scale GraspGen to both objects and grippers, we release a new simulated dataset consisting of over 53 million grasps. We demonstrate that GraspGen outperforms prior methods in simulations with singulated objects across different grippers, achieves state-of-the-art performance on the FetchBench benchmark for grasping in clutter, and performs well on a real robot with noisy visual observations.**

I. Introduction

Robot grasping has seen significant advances in recent years: scaling data generation [1], integration with touch sensing [2], operating in complex cluttered environments [3], language prompting [4], and real-world RL algorithms [5].

However, recent results show that critical gaps still exist in the development of a general-purpose grasping system. In the FetchBench [6] benchmark, state-of-the-art (SOTA) grasping systems achieve sub-20% accuracies. Similarly, the OK-Robot effort [7], which introduced a knowledge-based system for mobile manipulation in-the-wild, reported a notable error rate of 8% (30 errors out of 375 trials) due to grasp model failures alone. These grasping models perform at approximately 60% accuracy in their evaluations. The evaluation of Robo-ABC [8] (which uses the grasp model from [9]) as a baseline in RAM [10] shows sub-50% success rates. These highlight the need for further advancements in grasping frameworks to ensure their reliability as subroutines in higher-level reasoning systems [7], [11], [12].

Furthermore, grasping systems are not yet turnkey, and making them more flexible remains an open systems research challenge. Classical model-based approaches to grasp generation required precise object pose information [13] which does not generalize to unknown objects. Other methods necessitate multi-view scans for a single object [14], making them impractical in clutter. Contact point-based architectures [15], [16] often struggle to generalize to different gripper morphologies, limiting their applicability to hardware beyond symmetric parallel-jaw grippers. We also show that they perform comparatively worse at grasp evaluation.

World-centric models have also been proposed to generate grasps in cluttered scenes with multiple objects [15], [16], [9]. These models were initially introduced to 1) account for instance segmentation errors [3], [15] and 2) to consider environment collisions during the grasp generation process. Instance segmentation has matured significantly with the recent advent of foundation models such as SAM2 [17], mitigating the need for world-centric models. Further, we embrace modularity and allow GraspGen to specialize on only the task of stable grasping (agnostic of the scene or robot), as there are other more mature downstream modules that precisely solve for kinematic reachability (e.g. cuRobo [18]) and collision checking (e.g. nvBlox [19]). World-centric models typically require simulating entire scenes [15], [16] or manual data collection in the real world [9]. These approaches are challenging to scale to larger scenes beyond tabletops and it is non-trivial to synthetically generate cluttered scenes that accurately represent real-world distributions at inference time. As such, we revisit object-centric models, simplifying grasp generation during both training and inference.

In this work, we propose a new framework *GraspGen*, a combination of a diffusion-based generator and an efficient discriminator. Our technical novelty is three-fold. First, we propose a novel training recipe (Alg 1). A highlight of this recipe is that the grasp discriminator is supervised with our On-Generator Dataset. Prior work using 6-DOF Grasp Discriminators [20], [21], [14] typically train with offline datasets of successful and unsuccessful grasps. However, we show that there is a distribution shift between a dataset of grasps sampled from the diffusion model (hence

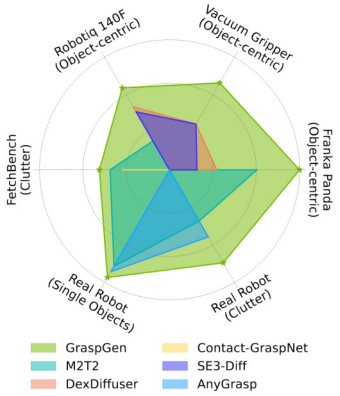


Fig. 1: GraspGen outperforms prior work in several settings.

* Work done at NVIDIA

called "On-Generator" dataset) and that of the offline dataset. We demonstrate that the performance of the discriminator improves substantially when training on this dataset. It becomes aware of the mistakes made by the diffusion model and assigns a corresponding lower score for potentially false positive grasps. Second, we show how architectural changes, such a transformer backbone and translation normalization, improves upon the entire model. Lastly, we demonstrate that GraspGen is a flexible system for scaling grasp generation across diverse settings, including: *embodiments* (compatibility with 3 distinct gripper types), *observability* (robustness to partial vs. complete point clouds), *complexity* (single-object vs. cluttered scenes), and *sim vs. real*. Apart from grasp accuracy, GraspGen enhances inference time and memory usage. We open source all aspects of this system — including data generation, documentation, data formats and a dataset consisting of 53M grasps as well as training — to support further research within the community.

II. Related Work

6-DOF Grasping. Planning robot grasps is usually formulated as a two-stage process: *grasp sampling* (GS) and *grasp analysis* (GA) [22]. Recently, generative models such as autoregressive models [23], Variational Autoencoder (VAE) [20], flow-matching [24], diffusion models [25], [26], [14], [27], [28], [29] have been proposed for GS. GA is typically done with a discriminator model to score and rank the sampled grasps [21], [20], [30], [3]. Some methods have a single model for both GA and GS for efficient inference — [15] proposed a contact point grasp representation and [16] extended this with a transformer for grasping as well as placing. In our framework, we improve upon both GA and GS.

Applications of 6-DOF Grasping. Understanding the applications of 6-DOF Grasping is crucial to designing the right modular framework. Popular applications that use 6-DOF grasp networks as a submodule include target-driven grasping in clutter [31], [15], [3], dynamic grasping [9], language-guided semantic manipulation [4] and task planners [11], [12]. Since downstream applications may require these networks to work with either single-view (in constrained environments) or multi-view camera observations, we train our framework to handle either situations.

Diffusion Models in Manipulation & Grasping. Diffusion models [32] are a powerful class of generative models. Recently, the robotics community has applied them to problems involving high-dimensional, multi-modal and continuous distributions: policy learning [33], [34], grasping [26], [21], [29], [28], [35], [14], motion planning [36] and rearrangement [37]. The closest paper to our work is SE3-Diff [26] which proposed the problem formulation of 6-DOF antipodal grasping as a diffusion process and DexDiffuser [21] which added a discriminator for grasp analysis. [29] suggested using DDIM instead of a DDPM [32] scheduler for faster diffusion inference (at the expense of grasp spatial coverage). We use DDPM since we do not want to compromise on coverge (which would affect performance in clutter) and we empirically found that DDPM achieved sufficiently fast

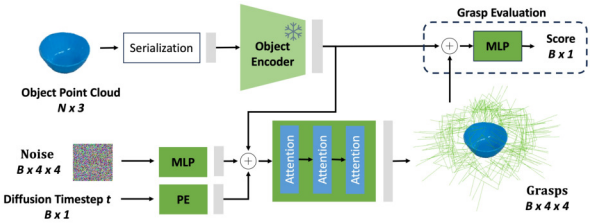


Fig. 2: Architecture for the diffusion noise prediction network and grasp evaluation. The model is iteratively used during diffusion grasp generation. However, the object encoder and MLP grasp evaluation is only called once. PE refers to Position Encoding.

inference in our implementation (∼20 Hz even before GPU acceleration). We leave diffusion acceleration techniques (e.g. DDIM, shortcut models[38]) to future work. Other methods have suggested inductive biases in the generative model — GraspLDM [27] proposed a VAE latent space to run grasp diffusion for modular task conditioning while SE(3)-equivariant inputs were introduced in [24]. Translation invariance emerges naturally since we do mean centering of point clouds and rely on data augmentation during training for rotation invariance. However, these tangential approaches can further complement GraspGen for task conditioning and reducing training time. Compared to prior work, we instead focus on: 1) improving GA performance with On-Generator training vis-a-vis standard grasp discriminators and 2) demonstrate that GraspGen generalizes to multiple settings (clutter, real robot and different embodiments).

III. GraspGen

The objective of grasp generation is to synthesize a large spatially-diverse set of grasp poses, as described in Sec III-A. We need the grasps to be diverse for performant execution in clutter, where many otherwise successful grasps are unreachable or in collision and hence are filtered out by the motion planner. In practice, the required output is a set of top-K grasps for the object. As such, generated grasps need to be scored and ranked to return the best performing grasps. This is done with grasp evaluation in Sec III-B.

A. Grasp Generation with Diffusion

We formulate the problem of 6-DOF grasp generation as a conditional diffusion model in SE(3) [26]. Conditioned on a specific object, the grasp distribution is continuous and highly multimodal, making it a suitable problem for generative modeling. At a high level, diffusion models entail adding noise sequentially to the training data. This process is reversed during inference time, where the data is generated from noise. Similar to prior work [26], [21], we formulate the problem as a Denoising Diffusion Probabilistic Model (DDPM) [32], which models a distribution using an iterative denoising process. The space on which we perform the diffusion is in the SE(3) Lie group. Unfortunately, the rotation space is not Euclidean, but DDPMs are proposed to model data coming from a Euclidean space in $\mathbb{R}^n$. Analogous to [26] we factorize SE(3) into SO(3) $\times$ $\mathbb{R}^3$, where $\mathbb{R}^3$ and SO(3) Lie algebra spaces are Euclidean.

Translation Normalization. Neural networks perform best when the dataset and labels are properly normalized [39]. For SO(3), the space is bounded between $[-\pi, \pi]$. However, translation is unbounded and heavily dependent on the scale of the object point cloud. While the object point clouds can be rescaled to be within a bound, the bounds of grasps in SE(3) vary based on each object's shape and pose. We normalize the grasp translations by the multiplier κ. Instead of setting this value arbitrarily or with a grid search, we compute this from the dataset statistics as $\kappa = \frac{1}{\frac{1}{N}\sum_{i=0}^{N}(max(t_i) - min(t_i))}$ where t_i is the translation component in $\mathbb{R}^3$ of all the positive grasps poses $\mathcal{G}_i^+$ for object i, aggregated across all objects.

Object Tokenization. We use the recently proposed PointTransformerV3 (PTv3) [40] as a backbone. PTv3 uses serialization to convert unstructured point clouds to a structured format, before applying a transformer on this serialized data. This sidesteps the computationally expensive process of nearest neighbor ball query search, a bottleneck in popular backbones like PointNet++ [41]. While point cloud transformer architectures [40] are making steady progress, to the best of our knowledge, no generative grasping paper has used them to encode objects. Several 6-DOF grasping papers have used PointNet++ [20], [26].

Diffusion Network. The diffusion noise prediction network, as used during inference time, is shown in Fig. 2. Both the point clouds as well as the grasps are transformed to the point cloud mean center before passing through the noise prediction network. During inference, we first sample a noise vector (with batch size B) and iteratively execute the diffusion reverse process to generate the grasps. Through hyper-parameter search we found that $T = 10$ denoising steps are sufficient for our setting. While diffusion models that generate images typically run for over hundreds of steps, we hypothesize that diffusion on grasps should be less complex, since the grasp dimensionality (3 for translation + 3 for rotation) is significantly lower than the dimensionality of pixels and videos ($> 50K$ for a 224×224 image). The training loss is a denoising loss on the position and orientation difference between the predicted vs. actual noise values: $L = \|\epsilon - \phi(t, \tilde{g}, \mathcal{X})\|_2^2$, where ϕ is the noise prediction network and $\mathcal{X}$ is the object point cloud. During training, we sample a random diffusion time step $t \in [0, T]$ and add random noise $\tilde{g} = g + \epsilon$ to the ground truth grasp $g \in \mathcal{G}^+$. The diffusion timesteps and grasp poses are processed with position encoding and a multilayer perceptron respectively. We empirically found that running two separate denoising processes with their own dedicated scheduler yielded better performance than running a single DDPM for the translation and rotation components. Note that the grasp scoring with the discriminator, as explained in the next section, is trained separately and is not used during diffusion model training.

B. Grasp Evaluation with On-Generator Training

A generative model trained solely on successful grasp data is prone to generating false positives due to model fitting errors. In practice, a mechanism is needed to score, rank and filter each grasp before executing it on the robot. To address this problem, earlier works [20], [14], [21] used a separately learned discriminator. We propose two key improvements.

On-Generator Training. Sim-to-real grasp models are typically trained with offline datasets of successful $\mathcal{G}^+$ and unsuccessful grasps $\mathcal{G}^-$. However, we show that there is a distribution shift between the grasps from the generative model $\hat{\mathcal{G}}$ and that of the offline dataset. We believe this is due to the nature of the grasp sampling algorithm during training. For instance, the unsuccessful grasps may never collide with the object (which is the case for ACRONYM [1]). However, some diffusion-generated grasps are slightly in collision with the object, potentially due to model fitting errors, and hence are unsuccessful. Furthermore, some generated grasps are occasionally outliers and are far away from the object. These correspond to noise samples with low likelihoods. We hypothesize that such failure modes can all be removed by training a discriminator with On-Generator training, as described in Algorithm 1. We first run inference on the training set with the diffusion model. This dataset corresponds to about 7K objects, each with 2K grasp samples per object. We then annotate this dataset by simulating the grasps using the same workflow used to generate the initial offline dataset. This On-Generator dataset approximately corresponds to the original size of the offline dataset.

Algorithm 1 GraspGen Training Recipe

Given: Object dataset $\mathcal{O}$, Grasp dataset $\{\mathcal{G}^+, \mathcal{G}^-\}$
Compute Translation Normalization: $\kappa \leftarrow trans\_norm(\mathcal{G}^+)$
Train Generator: $\pi^{gen} \leftarrow train\_DDPM(\mathcal{O}, \mathcal{G}^+, \kappa)$
Sample On-Generator dataset: $\hat{\mathcal{G}} \sim \pi^{gen}(\mathcal{O})$
Annotate On-Generator dataset: $\{\hat{\mathcal{G}}^+, \hat{\mathcal{G}}^-\} \leftarrow$ simulate$(\mathcal{O}, \hat{\mathcal{G}})$
Train Discriminator: $\pi^{dis} \leftarrow train\_classifier(\{\hat{\mathcal{G}}^+, \hat{\mathcal{G}}^-\}, \pi^{gen}, \kappa)$
Return: $(\pi^{gen}, \pi^{dis}, \kappa)$

Efficient Evaluation. Prior discriminator architectures [20], [14] had their own object encoder separately trained from scratch. We propose a simpler architecture, which reuses the object encoder from the generation stage for the subsequent grasp discrimination step. As shown in Fig. 2, an MLP takes in this object embedding and a corresponding grasp pose and predicts a sigmoid score of grasp success. Another important design decision is about efficiently combining the embedding for the object shape and grasp pose. In prior work [20], the grasp pose in SE(3) was converted to a point cloud (a handful of canonical points on the gripper were predefined and transformed with the grasp pose), concatenated with the object and passed into a PointNet with an additional input of a segmented point cloud. Instead, we simply concatenate the object embedding with the grasp pose. The discriminator is trained separately from the diffusion-based generator. Only the final MLP layer is trained from scratch with a binary cross entropy loss. The object encoder from the generator is frozen and re-used for the discriminator.

C. Dataset

Our dataset includes 6D gripper transformations and corresponding binary success labels for a repertoire of object meshes. The label generation process follows the pipeline used in ACRONYM [1]. While ACRONYM is based on ShapeNetSem, we use the more permissive, larger, and more

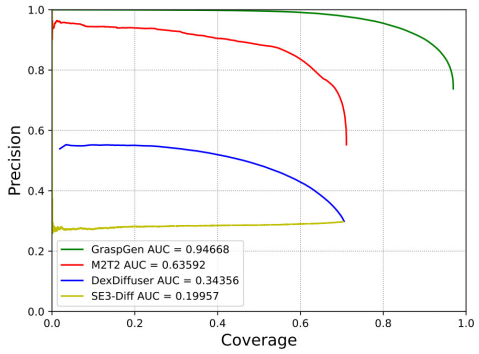


Fig. 3: Object-centric evaluation on Franka-ACRONYM [1].

diverse Objaverse dataset of 3D objects [42]. Specifically, we select a subset of meshes from Objaverse that overlap with the 1,156 categories in the LVIS dataset [43] and are licensed under CC-BY, totaling 36,366 meshes. To compare with models trained on ACRONYM, we further select a random subset of 8,515 object meshes to match the size to the ACRONYM dataset. For each object, 2K grasp transformations are uniformly sampled around the mesh. The label of a grasp is determined by simulating a shaking motion with the object in hand in the Isaac simulator [44]. A grasp is considered successful if a stable contact configuration is present after the shaking motion finishes. We construct datasets with around 17M grasps/gripper for the following commonly used grippers: Franka Panda, Robotiq 2F-140 and vacuum gripper (30mm suction cup, where success is labeled using an analytical model [45]).

IV. Experimental Evaluation

A. Simulation Results

Baseline Methods. We compare GraspGen with multiple recent methods: Contact-point architectures M2T2 [16] and Contact-GraspNet [15], diffusion architectures DexDiffuser [21] and SE3-Diff [26] as well as AnyGrasp [9]. We reimplemented SE3-Diff with the main difference of using a PointNet++ backbone to process the point cloud input of unknown objects. Since the model does not score each generated grasp, we use the approximate log-likelihood instead. DexDiffuser [21] was originally proposed for dexterous grasping, where grasps are parameterized by a pose and gripper joint configuration. Since we focus on pinch and suction grasping, we reuse the architecture with only the pose input. We skip comparing to a Variational Autoencoder (VAE) baseline since it was already demonstrated to have worse performance than the baselines we compared to [26], [15] in their corresponding papers. We directly compare to these recent methods without repeating the VAE baseline. We skip comparing to GraspLDM since it is reported to show comparable real world performance with Contact-GraspNet [15], which we substantially outperform in FetchBench evaluations (Fig 4) and we compare to the more recent M2T2 [16] which itself outperforms [15] as well.

Full Point Cloud of Single Objects. We begin by evaluating grasp generation using a complete point cloud sampled from the object's mesh, without any self-occlusion. For each method, we evaluate on a test set of 815 objects with 2K grasps/object, resulting in a total of 1.6 million grasp executions. We trained all models using the same training set and splits. For M2T2, we just use a single transformer token, since there is only one object in the scene. The evaluation pipeline follows the setting of data collection in Sec. III-C, where the grasp poses are attempted with a isolated free-floating gripper. We present results on the widely used ACRONYM dataset [1] for Franka. Extended results on GraspGen datasets are in the Appendix. Due to license restrictions limiting model usage to registered machines, we were unable to compare to AnyGrasp [9] in these simulation experiments on the compute cluster but evaluated it in the real world on a registered desktop (Sec IV-E).

The results are summarized in the Precision-Coverage curve in Fig. 3. *Precision* represents the grasp success rate in simulation, while *Coverage* is a measure of spatial diversity of the grasps and is the percentage of the ground truth positive grasp set, $\mathcal{G}^+$, matched by the predicted grasps. The matching is done using nearest neighbour assignment (distance of 1cm) used in prior work [16], [15], [20]. GraspGen outperforms baselines by over $48\,\%$ in terms of Area Under Curve (AUC). All methods with discriminative reasoning (GraspGen, DexDiffuser, M2T2) outperformed SE3-Diff, a purely generative method and scored based on the approximate loglihood. This hightlights the importance of a discriminator in grasp generation. This result also demonstrates that the discriminator quality is crucial. GraspGen with its On-Generator training is able to better score the grasps compared to the discriminator of DexDiffuser [21], as it specifically trains for the distribution of grasps from the diffusion model. While the discriminator in GraspGen is trained on both positive and negative grasps, M2T2 is trained exclusively on positive grasps and only distinguishes between good and bad contact points, resulting in a worse performance.

Task-level Evaluation in Clutter. We evaluate GraspGen's ability to handle complex grasping in clutter using FetchBench [6], a simulation-based grasping benchmark with diverse procedural scenes. FetchBench simulates all stages of grasping, from perception, grasp pose detection, collision world modeling, to motion planning. The experiments are conducted with a Franka Panda robot in 100 scenes (see Fig 8) with 60 tasks per scene for a total of 6K grasp executions. To focus solely on grasp pose detection and eliminate confounding factors, we use the ground truth collision mesh of the scene for motion planning with cuRobo [18]. As summarized in Fig 4, we first report an *oracle* planner with ground truth grasps from the dataset [1], to demonstrate the best grasp performance possible without any sensing or model performance issues. We report two success metrics: 1) task success rate, which measures successful completion from grasping to placing the object, and 2) grasp success rate, which tracks successful grasps only. The latter is always higher, as some grasps succeed but may slip or collide during retraction. Interestingly, the *oracle* planner achieves only $65\,\%$ grasp success and $49.2\,\%$ task success. This low performance stems from several factors: 1) many

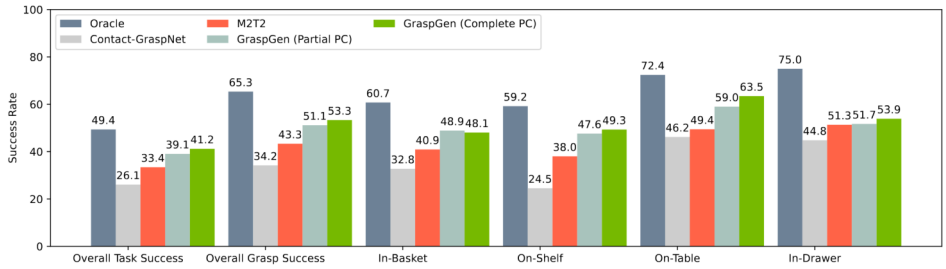
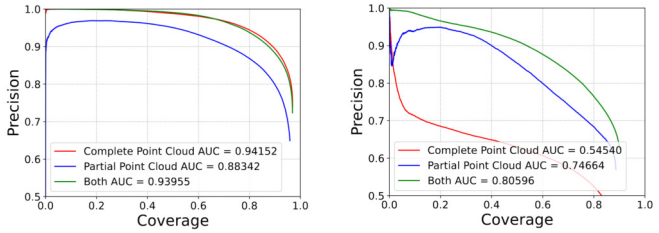
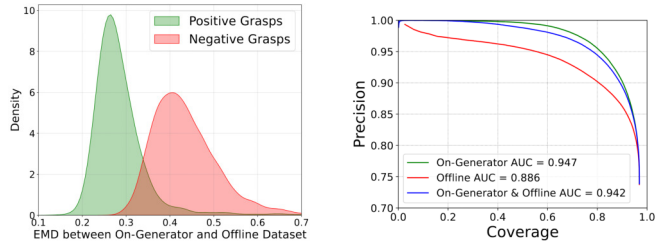


Fig. 4: Large-scale evaluation on FetchBench [6]. GraspGen surpasses all previous methods.



(a) Complete point clouds (b) Single-view point clouds

Fig. 5: Ablation on Observability.



(a) Distribution shift in the on-generator vs. offline datasets (b) Ablation on Offline vs. On-Generator Training

Fig. 6: Ablation on On-Generator Training and Datasets.

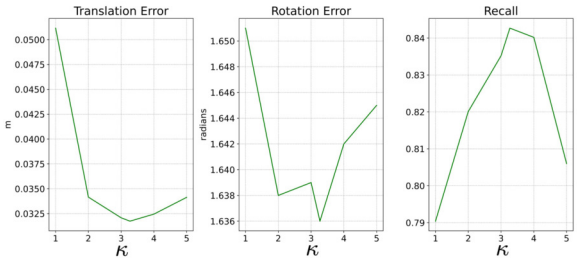


Fig. 7: Ablation on Translation Normalization on Franka-ACRONYM [1]. κ is plotted on the x-axis.

scenarios allow grasping but lack a collision-free retraction path, 2) some problems exceed the capabilities of existing motion planners [18], and 3) objects are in complex poses where stable grasps exist but are inaccessible. Addressing these challenges requires more advanced non-prehensile manipulation and/or reasoning policies beyond the scope of this paper. Nonetheless, GraspGen achieves SOTA results surpassing Contact-GraspNet [15] and M2T2 [16] by 16.9 % and 7.8 % respectively.

Sensitivity to Occlusions. Prior work has proposed 6-DOF grasp generation for either single-view partial point clouds with strong self-occlusion [20], [15], [16] or complete point clouds, by fusing multiple camera views [26], [14]. As shown in Fig. 5, GraspGen trained on partial point clouds performs poorly on complete point clouds and vice versa. By training on a mix (50-50 split) and generalizing to both settings, GraspGen provides flexibility for downstream applications.

B. Analysis of On-Generator Training

On-Generator training of the discriminator is essential for enabling the model to recognize its own failure modes and filter out erroneous predictions. Before we show results with On-Generator training, we first demonstrate that there is a measurable distribution shift between the offline data distribution vs. diffusion-model generated sample distribution. We use the Earth Mover's Distance (EMD) (detailed in the Appendix) and plot this separately for positive and negative grasps for the entire training set of (∼7K) objects in Fig 6(left). There is a substantial non-zero EMD between the offline and on-generator datasets. It is especially more pronounced for negative grasps, since the spatial manifold of unsuccessful grasps (e.g. grasps far away or colliding an object are still considered negative examples) is larger than that of successful grasps (i.e. grasps need to be close to the object surface). This distribution shift justifies our need to train our discriminator to specifically filter out unsuccessful grasps.

As shown in Figure 6, the model trained exclusively on On-Generator data achieved the highest performance. The model trained with the offline dataset performed the worst (6.5 % lower AUC). We hypothesize that the On-Generator training captures the false positives of the diffusion model better than the offline dataset. For example, the generator may produce grasps that are slightly in collision with the object's collision mesh – cases absent from the offline dataset, which contains only collision-free grasps. Additionally, model fitting errors can introduce outliers with small translation or rotation errors, leading to unstable grasps.

C. Ablation Studies

Analysis of the Discriminator. Compared to prior grasp discriminator architectures [20], GraspGen is more accurate (6.7% higher AUC, 5.87% higher mean Average Precision (mAP) of the binary classification sigmoid scores) [20], [21] and uses 21× less memory [20] as explained in the Appendix.

Translation Normalization. We found a convex relationship between the performance of the diffusion model and the normalization multiplier. Fig. 7 summarizes the key results, where the goal is to minimize the translation/rotation error

Fig. 8: We evaluate in diverse cluttered environments in NVIDIA IsaacSim simulation [6] *(top)* and real *(bottom)*.

while maximizing the coverage (recall). While an optimal scale factor can be found via hyperparameter grid search, we empirically observe that computing the normalization multiplier using the equation detailed in Sec III-A) results in a local minima and provides a reliable alternative. We computed κ=3.27, κ=2.02 for the Franka and Robotiq respectively.

Rotation Representation. All tested representations — 6D rotation representation, Euler angles, Lie Algebra — performed comparably. See supplementary for full experimental results. We use Lie Algebra for all other evaluations.

Pointcloud Encoder. We demonstrate substantial gains using the SOTA transformer backbone PointTransformerV3 (PTv3) [40] over PointNet++ [41]. PTv3 reduces translation error by $5.3\,\mathrm{mm}$, reduces rotation error by 14.4° and increases recall by $5\,\%$.

D. Performance on Multiple Grippers

While the main paper presents comparisons for the Franka gripper, results for the Robotiq 2F-140 and suction grippers are included in the Appendix. GraspGen is the most proficient method across grippers. In the Franka-sim experiments, GraspGen outperforms M2T2 by $37\,\%$ (Fig. 3), with even larger margins in Robotiq-sim ($44\,\%$) and real-robot experiments ($57\,\%$). This is likely because M2T2 relies on a contact point representation [15], which is designed for symmetric, non-adaptive grippers and struggles with adaptive grippers like the Robotiq 2F-140. GraspGen also outperforms SE3-Diff [26] across all three grippers. We note that GraspGen performance varies based on the gripper and we attribute this to gripper complexity. While the Franka is a standard antipodal gripper, suction is symmetric along the approach direction and the Robotiq is a more complicated underactuated grasping motion.

E. Real Robot Evaluation

GraspGen generalizes to the real world despite being only trained in simulation. Our hardware setup consists of a UR10 arm with a single extrinsically calibrated RealSense D435 RGB-D camera overlooking a tabletop scene. Motion planning is done with cuRobo [18] on a Jetson while NVBlox [19] is used for collision avoidance. We use SAM2 [17] running on a 6000 Ada GPU for instance segmentation, as well as FoundationStereo [46] for depth estimation.

We specifically evaluate the model trained on the GraspGen Robotiq 2F-140 dataset. We compare GraspGen to M2T2 [16] since it had the best performance in simulation among all competitors. AnyGrasp [9] is another recent grasping in clutter framework trained on real colored point clouds of tabletop objects. Due to license restrictions, we were unable to compare to AnyGrasp in our simulation experiments but evaluated it in the real world. For both methods, we use the weights and configuration released in the respective papers. We had two modifications for M2T2: a 90°rotation of the point cloud around z and a 3D box crop encompassing the robot's workspace to match its training distribution. For AnyGrasp, we had to apply a translation offset along the camera's z axis to match the original training dataset, which was collected at a fixed camera depth (despite randomized elevation/azimuth). We empirically found that inference without non-maximal suppression was better, most likely since our motion planner [18] is proficient with goal set targets. We were unable to get consistent grasp predictions without these modifications for both models. All models return a set of predicted grasps and confidence scores. We use the top-100 grasps as pose targets for the motion planner. The planner filters out grasps that are in collision or do not have an inverse kinematics solution.

Method	Isolated Objects	Clutter			Overall
		Table	Basket	Shelf	
GraspGen	**90.5%**	**83.3%**	**80.0%**	**71.4%**	**81.3%**
M2T2 [16]	81.0%	75.0%	40.0%	14.3%	52.6%
AnyGrasp [9]	85.7%	83.3%	42.9%	42.9%	63.7

TABLE I: Real Robot - Grasp Success Rates

We evaluate four different settings: isolated objects without any clutter, multiple objects on a table, inside a basket, and on a shelf (clockwise, Fig. 8 *(right)*). As shown in Table I, GraspGen achieved an overall success rate of $81.3\,\%$, outperforming M2T2 and AnyGrasp by $28\,\%$ and $17.6\,\%$ respectively. GraspGen performed well across different environments, though it struggled in the more challenging shelf and basket setting. Motion planning is more diffucult in these settings as cuRobo filters out most grasps due to kinematic/collision restrictions. As such, the models need to both 1) generalize to these settings and 2) generate grasps with high coverage, to increase the chances of having feasible grasps after all the filtration steps. Since both M2T2 and AnyGrasp are scene-centric models trained only with data for tabletop clutter, they were unable to generalize to more complicated environments. M2T2 also did not generate grasps on some smaller objects, most likely due to the low point cloud resolution on these objects when reasoning at the scene level. Examples of grasp predictions are in the Appendix.

V. Conclusion & Limitations

We presented GraspGen, a 6-DOF grasp generation framework with an improved diffusion model, validated across various settings (embodiments, clutter, sim, real). GraspGen outperforms baseline methods and achieves state-of-the-art results on the FetchBench [6] benchmark. In terms of limitations, the performance of GraspGen depends on the quality of depth sensing and instance segmentation.

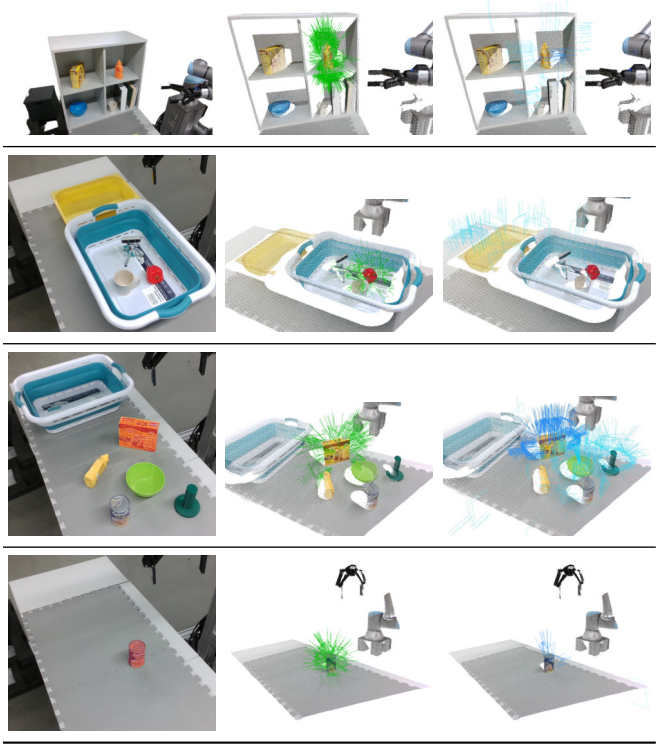
RGB+Segmentation	GraspGen (Ours)	M2T2 [16]
		

TABLE II: Qualitative grasp predictions from the real robot experiments overlaid on the colored point cloud in the robot frame, for GraspGen (middle column) and the M2T2 [16] baseline (right column). From top to bottom, we show a representative example from each of the environments (row 1-3 is in clutter): shelf, basket, tabletop, single isolated object. The target object to grasp is highlighted in red on the left column. GraspGen only generates grasps for the target object (green grasps in middle column). Since M2T2 is a scene-centric model, we plot the predicted grasps for all objects in light blue and the predictions specific for the target object in dark blue. Overall, GraspGen's predictions are more focused on the target object and have greater coverage than M2T2.

VI. Appendix

We provide the following details in the Appendix: 1) qualitative grasp visualizations, 2) dataset statistics 3) explanation of the Radar chart in Sec. I 4) Robotiq 2F-140 and suction gripper baselines and 6) explanation of EMD in Fig 6.

Qualitative Visualizations of Grasp Predictions: Qualitative grasp predictions are shown in Table II. Overall, GraspGen's predictions are more focused on the target object and have greater coverage, when compared to the M2T2 [16] baseline. M2T2 sometimes does not generate any grasps for some target objects (the bell pepper in the 2nd row), especially when they are small. We believe this is because M2T2 is a scene-centric model, and the resolution is insufficient to capture the geometry of smaller objects (i.e. there are too few points on them) - this is unavoidable for models reasoning at the scene-level. Furthermore, M2T2 generates several false positive grasps in the environment, which may seep into the contact mask of any neighboring target objects.

Further Dataset Statistics: A detailed comparison between our dataset and prior work is shown in Table III.

Quantitative Metrics in Radar Chart in Sec. I: The radar chart in Sec. I shows the aggregate performance of GraspGen and the baselines in various settings. For the object-centric simulation experiments, the key metric was Area Under Curve (AUC) of a Precision-Coverage curve, as displayed in Fig 3 for the Franka and explained in text below for Robotiq 2F-140 grippers. For the Suction gripper, we did not train a M2T2 model since we were not able to directly apply the contact-graspnet representation. Instead, we simply report the coverage metric comparing just the generators of GraspGen and SE3-Diff [26]. For both the FetchBench and real robot experiments, we report grasp success rates shown in Fig 4 and Table I respectively.

Baseline Comparisons for Robotiq 2F-140: GraspGen (AUC: 0.69) outperforms both M2T2 [16] (AUC: 0.24) and SE3-Diffusion Fields (AUC: 0.43) by a substantial margin. M2T2 uses the contact point formulation from [15], which is designed for symmetric, non-adaptive grippers pinch grippers and hence does not directly transfer to adaptive grippers like the Robotiq 2F-140. Due to the poor performance of this model, we use the M2T2 Franka Panda model with a fixed translation offset (-10cm along the z axis) for real-world robot experiments (see Table I in Sec. IV-E).

Baseline comparisons for Suction: Compared to SE3-Diffusion Fields, GraspGen achieves a slightly higher L2 translation error (1.5 cm), a lower rotation error (by 7.2°) and a substantially larger coverage (+3.6%).

Computing Earth-Movers-Distance (EMD) in Fig. 6: We want to measure the distribution shift between the data generated by the diffusion model $\mathcal{G}$ compared to the training dataset $\hat{\mathcal{G}}$. Given these two datasets for the same object, we subsample 500 grasps from each. For each pose $g_i \in \mathcal{G}$ and $g_j \in \hat{\mathcal{G}}$, we measure the pair-wise distance using the cost function introduced in [26]: $d(g_i, g_j) = \|t_i - t_j\| + \left\|\text{LogMap}(\mathbf{R}_i^{-1}\mathbf{R}_j)\right\|$. We then solve a Linear Sum Assignment optimization problem, which effectively searches for the one-to-one assignment between the samples in both distributions based on the lowest distance. We repeat this process for 5 random subsamples of 500 grasps from both distributions, and average to get a score for each object.

Explanation of Memory Usage: GraspGen uses 21× less memory than prior work in discriminators [20]. In [20] the grasp poses $g \in SE(3)$ were transformed to a grasp point cloud $X \in \mathbb{R}^{N\times 3}$ (where $N = 5$ are predefined set of points on the gripper) and input to a PointNet++ [41] backbone with an additional segmentation label to specify the points from the gripper vs. object. This caused the GPU memory to scale $\mathcal{O}(N \times B)$ where B is the batch size. However, in GraspGen (Fig. 2), we simply pass in the grasp poses in $SE(3)$ (i.e. memory scales with $\mathcal{O}(B)$ instead).

Dataset Ablation: Our proposed datasets are comparable with prior datasets. For Franka, the model trained on ACRONYM [1] had a AUC of 0.95 and 0.91 respectively on the ACRONYM and GraspGen test sets, while the model trained on GraspGen had a AUC of 0.84 and 0.91 respectively. We attribute the slight mismatch to the simulation physics engine (Flex [51] in ACRONYM vs. PhysX [56] in ours).

Dataset	Year	#Grippers	#Objects	#Grasps	Grasp Label	Synthesis Method	Code + Data
HO-3D [47]	2020	1 (Human hand)	10	78K	Only +ve	Human Demo	✓
EGAD [48]	2020	1 (2-finger)	2,331	233K	Only +ve	Evolutionary Algorithm	✓
DDG [49]	2020	1 (5-finger)	500	50K	Only +ve	GraspIt + modified Q1	✗
DexYCB [50]	2021	1 (Human hand)	20	582K	Only +ve	Human Demo	✓
Acronym [1]	2021	1 (2-finger)	8,872	17.7M	+ve & -ve	Flex [51]	✓
UniGrasp [52]	2020	12 (2 & 3finger)	1000	2M+	Only +ve	Contact Points Network + FastGrasp	✓
DexGraspNet [53]	2023	1 (5-finger)	5,355	1.3M	Only +ve	Differentiable grasping	✓
Fast-Grasp'D [54]	2023	3 (3-5 finger)	2,350	1M	Only +ve	Differentiable grasping	✗
MultiGripperGrasp [55]	2024	11 (2-5 finger & Human)	345	30.4M	Ranked	GraspIt + Isaac Sim [56]	✓
GraspGen (Ours)	**2025**	**3 (2-finger & Suction)**	**8,515**	**53.1M**	**+ve & -ve**	**Sampling + Isaac Sim [56]**	✓

TABLE III: Comparison of GraspGen with existing grasping datasets. Adapted from [55].

REFERENCES

[1] C. Eppner *et al.*, "Acronym: A large-scale grasp dataset based on simulation," in *ICRA*, 2021.

[2] R. Calandra *et al.*, "More than a feeling: Learning to grasp and regrasp using vision and touch," *Robotics and Automation Letters*, 2018.

[3] A. Murali *et al.*, "6-dof grasping for target-driven object manipulation in clutter," in *ICRA*, 2020.

[4] C. Tang *et al.*, "Graspgpt: Leveraging semantic knowledge from a large language model for task-oriented grasping," *RAL*, 2023.

[5] D. Kalashnikov *et al.*, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *CoRL*, 2018.

[6] B. Han *et al.*, "Fetchbench: A simulation benchmark for robot fetching," in *Conference on Robot Learning*, 2024.

[7] P. Liu *et al.*, "Ok-robot: What really matters in integrating open-knowledge models for robotics," *preprint arXiv:2401.12202*, 2024.

[8] Y. Ju *et al.*, "Robo-abc: Affordance generalization beyond categories via semantic correspondence for robot manipulation," in *European Conference on Computer Vision*. Springer, 2025, pp. 222–239.

[9] H.-S. Fang *et al.*, "Anygrasp: Robust and efficient grasp perception in spatial and temporal domains," *Transactions on Robotics*, 2023.

[10] Y. Kuang *et al.*, "Ram: Retrieval-based affordance transfer for generalizable zero-shot robotic manipulation," in *CoRL*, 2025.

[11] M. Dalal *et al.*, "Local policies enable zero-shot long-horizon manipulation," in *ICRA*, 2025.

[12] A. Deshpande *et al.*, "Graspmolmo: Generalizable task-oriented grasping via large-scale synthetic data generation," in *CoRL*, 2025.

[13] X. Deng *et al.*, "Self-supervised 6d object pose estimation for robot manipulation," in *ICRA*, 2020.

[14] T. G. W. Lum *et al.*, "Get a grip: Multi-finger grasp evaluation at scale enables robust sim-to-real transfer," in *CoRL*, 2024.

[15] M. Sundermeyer *et al.*, "Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes," in *ICRA*, 2021.

[16] W. Yuan *et al.*, "M2t2: Multi-task masked transformer for object-centric pick and place," in *Conference on Robot Learning*, 2023.

[17] N. Ravi *et al.*, "Sam 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024.

[18] B. Sundaralingam *et al.*, "curobo: Parallelized collision-free minimum-jerk robot motion generation," in *ICRA*, 2023.

[19] A. Millane *et al.*, "nvblox: Gpu-accelerated incremental signed distance field mapping," 2024.

[20] A. Mousavian *et al.*, "6-dof graspnet: Variational grasp generation for object manipulation," in *ICCV*, 2019.

[21] Z. Weng *et al.*, "Dexdiffuser: Generating dexterous grasps with diffusion models," *Robotics and Automation Letters*, 2024.

[22] R. Newbury *et al.*, "Deep learning approaches to grasp synthesis: A review," 2022.

[23] J. Tobin *et al.*, "Domain randomization and generative models for robotic grasping," in *IROS*, 2018.

[24] B. Lim *et al.*, "Equigraspflow: Se(3)-equivariant 6-dof grasp pose generative flows," in *CoRL*, 2024.

[25] Y.-H. Wu *et al.*, "Learning generalizable dexterous manipulation from human grasp affordance," in *CoRL*, 2023.

[26] J. Urain *et al.*, "Se(3)-diffusionfields: Learning smooth cost functions for joint grasp and motion optimization through diffusion," *ICRA*, 2023.

[27] K. BARAD *et al.*, "Graspldm: Generative 6-dof grasp synthesis using latent diffusion models," in *IEEE Access*, 2024.

[28] R. Freiberg *et al.*, "Diffusion for multi-embodiment grasping," in *Robotics and Automation Letters*, 2024.

[29] J. Carvalho *et al.*, "Grasp diffusion network: Learning grasp generators from partial point clouds with diffusion models in so(3)xr3," 2024. [Online]. Available: https://arxiv.org/abs/2412.08398

[30] P. Song *et al.*, "Implicit grasp diffusion: Bridging the gap between dense prediction and sampling-based grasping," in *CoRL*, 2024.

[31] P. Xie *et al.*, "Rethinking 6-dof grasp detection: A flexible framework for high-quality grasping," *Pattern Recognition*, 2025.

[32] J. Ho *et al.*, "Denoising diffusion probabilistic models," in *Neural Information Processing Systems*, 2020.

[33] C. Chi *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," *IJRR*, 2024.

[34] T.-W. Ke *et al.*, "3d diffuser actor: Policy diffusion with 3d scene representations," in *CoRL*, 2024.

[35] Y. Zhang *et al.*, "Diffgrasp: Whole-body grasping synthesis guided by object motion using a diffusion model," in *AAAI*, 2025.

[36] H. Huang *et al.*, "Diffusionseeder: Seeding motion optimization with diffusion for rapid motion planning," in *CoRL*, 2024.

[37] W. Liu *et al.*, "Structdiffusion: Language-guided creation of physically-valid structures using unseen objects," in *RSS*, 2023.

[38] K. Frans *et al.*, "One step diffusion via shortcut models," in *ICLR*, 2025.

[39] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.

[40] X. Wu *et al.*, "Point transformer v3: Simpler, faster, stronger," in *CVPR*, 2024.

[41] C. R. Qi *et al.*, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space." *NeurIPS*, 2017.

[42] M. Deitke *et al.*, "Objaverse: A universe of annotated 3d objects," in *Computer Vision and Pattern Recognition*, 2023.

[43] A. Gupta *et al.*, "Lvis: A dataset for large vocabulary instance segmentation," in *CVPR*, 2019, pp. 5356–5364.

[44] V. Makoviychuk *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv:2108.10470*, 2021.

[45] J. Mahler *et al.*, "Dex-net 3.0: Computing robust robot vacuum suction grasp targets in point clouds using a new analytic model and deep learning," in *ICRA*, 2018.

[46] B. Wen *et al.*, "Foundationstereo: Zero-shot stereo matching," in *CVPR*, 2025.

[47] S. Hampali *et al.*, "Honnotate: A method for 3d annotation of hand and object poses," in *CVPR*, 2020.

[48] D. Morrison *et al.*, "Egad! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation," *RAL*, 2020.

[49] M. Liu *et al.*, "Deep differentiable grasp planner for high-dof grippers," *arXiv preprint arXiv:2002.01530*, 2020.

[50] Y.-W. Chao *et al.*, "Dexycb: A benchmark for capturing hand grasping of objects," in *CVPR*, 2021.

[51] M. Macklin *et al.*, "Unified particle physics for real-time applications," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, pp. 1–12, 2014.

[52] L. Shao *et al.*, "Unigrasp: Learning a unified model to grasp with multifingered robotic hands," *Robotics and Automation Letters*, 2020.

[53] R. Wang *et al.*, "Dexgraspnet: A large-scale robotic dexterous grasp dataset for general objects based on simulation," in *ICRA*, 2023.

[54] D. Turpin *et al.*, "Fast-grasp'd: Dexterous multi-finger grasp generation through differentiable simulation," *arXiv:2306.08132*, 2023.

[55] L. F. Casas *et al.*, "Multigrippergrasp: A dataset for robotic grasping from parallel jaw grippers to dexterous hands," in *IROS*, 2024.

[56] NVIDIA, "Nvidia isaac sim," 2023.