Learning Continuous Grasping Function with a Dexterous Hand from Human Demonstrations

Jianglong Ye[1*], Jiashun Wang[2*], Binghao Huang[1], Yuzhe Qin[1], Xiaolong Wang[1]

Fig. 1: Examples of our generated trajectories learned from human demonstrations. Given hand-object trajectories from human video (a), we first translate them into robot manipulation demonstrations (b). We then train Continuous Grasping Function (CGF) to generate human-like trajectories and deploy them in simulation (c) and real robot (d).

Abstract—We propose to learn to generate grasping motion for manipulation with a dexterous hand using implicit functions. With continuous time inputs, the model can generate a continuous and smooth grasping plan. We name the proposed model Continuous Grasping Function (CGF). CGF is learned via generative modeling with a Conditional Variational Autoencoder using 3D human demonstrations. We will first convert the large-scale human-object interaction trajectories to robot demonstrations via motion retargeting, and then use these demonstrations to train CGF. During inference, we perform sampling with CGF to generate different grasping plans in the simulator and select the successful ones to transfer to the real robot. By training on diverse human data, our CGF allows generalization to manipulate multiple objects. Compared to previous planning algorithms, CGF is more efficient and achieves significant improvement on success rate when transferred to grasping with the real Allegro Hand. Our project page is available at https://jianglongye.com/cgf/.

Index Terms—Learning from Demonstration; Dexterous Manipulation; Deep Learning in Grasping and Manipulation

I. INTRODUCTION

LEARNING to perform grasping with a multi-finger hand has been a long-standing problem in robotics [1]–[4]. Using a dexterous hand instead of a parallel gripper offers the robot the flexibility on operating with daily life objects like humans do, but also largely increases the difficulty given the large Degree-of-Freedom of the dexterous hand. A typical method for this task is a 2-step paradigm including grasp pose estimations following by motion planning [5]–[7]. Recent works have also studied on using Reinforcement Learning with human demonstration guidance for grasping [8], [9].

While these approaches have shown encouraging results, they plan the grasping with finite discrete time steps. On the other hand, human grasping motion is continuous, can

we learn a continuous grasping process for robot hands? Making robot grasping continuous can lead to a more natural and human-like trajectory, and each step we sample will be differentiable which we can use a more robust PD control with feedforward. Recent progress on neural implicit functions have shown successful applications in learning continuous image representation [10], [11] and continuous 3D shape representation [12]–[14]. Can this success be migrated from representing 2D/3D space to time?

In this paper, we propose to learn Continuous Grasping Function (CGF) with a dexterous robotic hand. To mimic the continuous human motion, we utilize human grasp trajectories from videos to provide demonstrations and supervision in training. By training CGF with generative modeling on a large-scale of human demonstrations, it allows generalization to grasp multiple objects with a real Allegro robot hand as shown in Figure 1 (d).

Specifically, given the 3D hand-object trajectories from human videos (Figure 1 (a)), we first perform motion retargeting to convert the human hand motion to the robotic hand motion to obtain the robotic manipulation demonstrations (Figure 1 (b)). We then learn a CGF in the framework of a Conditional Variational AutoEncoder (CVAE) [15] by reconstructing the robotic hand motion with these demonstrations. Specifically, the conditional encoder of our CGF model will take the object point clouds as inputs and provides the object embedding. Taking the concatenation of the object embedding, a latent code z and a time parameter t, the decoder of CGF is an implicit function which outputs the dexterous hand parameters in the corresponding time t. By sampling a continuous-time sequence of t, we can recover a continuous grasping trajectory in any temporal resolution. By sampling the latent code z, we can achieve diverse trajectories for the same object. Figure 1 (c) shows an example of the inferred grasping trajectory in the simulator.

In our experiments on testing our model, we will perform sampling on the latent code z multiple times given a test object and generate diverse grasping trajectories. We then execute these trajectories in the simulator and select the one which can successfully grasp the object up. Different from the previous paradigm on grasping followed by planning, we empirically find our method is much more efficient since we avoid performing planning for each trajectory but directly generate the trajectory from CGF. Given the selected trajectories from the simulator, we can deploy them in the real world with an Allegro hand attached on an X-Arm 6 robot. Compared with planning, our method achieves better Sim2Real generalization with more natural and human-like motion, which leads to a better success rate.

We highlight our main contributions here: (i) A novel Continuous Grasping Function (CGF) model which allows smooth and dense sampling in time for generating grasping trajectory; (ii) CGF allows efficient generation of grasping plan and more robust control in simulation; (iii) We achieve much significant improvement on Sim2Real generation on Allegro hand by learning CGF from human demonstrations.

II. RELATED WORK

Generalization in Dexterous Manipulation. Dexterous manipulation is one of the most challenging problems in robotics [1]–[4], [16], [17]. Recent studies on model-free [18]–[21] and model-based [22], [23] Reinforcement Learning (RL) have achieved encouraging results on multiple complex dexterous manipulation tasks. However, there is still a large challenge on generalization for RL. For example, when the RL policy in [18] can be transferred to the real robot, it is learned specifically for one object. On the other hand, an RL policy trained with multiple objects in simulator [21] has not yet been transferred to real. Instead of using RL, one line of works on dexterous grasping is first performing a grasp estimation and then planning for execution [5], [7], [24], which have shown great success on generalization to multiple objects and in real robots at the same time. Our work aligns more closely with this line of research. Through training on large-scale human demonstrations, our method is able to generate grasping trajectories for unseen objects based on their geometry. Compared to previous planning methods, our approach provides more diverse, smooth and natural grasping trajectories in a more efficient way, as evidenced by our experimental results.

Grasping Motion Synthesis. Synthesizing and predicting human grasp has been an active research field for both computer vision and robotics [6], [25]–[27]. For example, Grasping Field [26] is proposed as an implicit function that generates plausible human grasps given a 3D object. However, to apply on the robot hand, we will need to synthesize the full motion instead of a static pose. This motivates the research on synthesizing the hand-object interaction motions [28]–[31]. For example, a full body and hand motion are synthesized together to grasp an object in [30]. While related to our work, most approaches are still focusing on modeling the human hand. In this paper, we provide a framework where we first retarget the human hand trajectories to the robot hand trajectories and learn the robot grasping function with them.

Learning from Human Demonstrations. Our work is related to imitation learning or RL with human demonstrations for not only parallel grippers [32]–[34] but also dexterous hands [9], [35]–[38]. For example, DexMV is a platform proposed in [9] for extracting 3D human demonstrations from videos, generating robot demonstrations by retargeting, and augmenting Reinforcement Learning with these demonstrations for multiple manipulation tasks. Although both methods utilize robot demonstrations for training, DexMV's use of GT states as inputs limits its generalizability to multiple objects and real robot transfer. In contrast, our implicit function generates continuous grasping given a point cloud input and can be deployed to the real robot. As most RL approaches are still with full access to GT states, they are not directly comparable to our method.

Implicit Functions for Robotic Tasks. Beyond its successful applications in computer vision, implicit functions have recently been explored in robotic manipulation tasks [39]–[43]. For example, NeRF [14] is used as a manner for learning 3D representations for control in [39]. Instead of using implicit

Fig. 2: Pipeline overview. During training, human demonstrations are first translated to robot joint positions which serve as the supervision for grasping function learning. During inference, our trained CGF takes a sampled latent code z, object feature and query time sequence as inputs to generate the trajectory. We then execute these trajectories in the simulator and deploy successful ones to the real robot.

functions to learn static 3D representations, our work focuses on the continuity in time. We build a grasping function to directly generate the trajectory instead of a static scene.

III. METHOD

A. Overview

We aim to learn human-like robot grasping given object point clouds as input. We emphasize that learning from human demonstration could lead to more natural trajectories and the continuity helps the following control. To this end, we train Continuous Grasping Function with a CVAE framework to generate continuous trajectories and deploy them in the simulator and real robot consecutively. The pipeline is shown in Fig. 2. During training, we first perform hand motion retargeting on a large-scale hand-object interaction dataset [44] to collect demonstrations. Then the retargeting results served as the supervision for the grasping function learning. During inference, numerous continuous trajectories are sampled for a specific object and tested in the simulator. The successful trajectories will be deployed to the real robot. Besides, we can take more advantage of continuous implicit function by utilizing PD control with feedforward with the derivative of the joint positions.

B. Human Demonstration Translation

Data collection on human hand-object interactions is relatively well established and accessible. Using large-scale human hand-object interaction data, we can learn patterns of how dexterous hands manipulate objects, and our goal is to generalize it to the robot hand. In this paper, we use ground truth trajectories from DexYCB [44]. Translating human hand motion to robotics motion is the first step. Following [38], we formulate our hand motion retargeting problem as a position-based optimization problem. We encourage the robot's joint

position to be as close as possible to its corresponding human hand joint position,

$$\min_{q_t} \sum_{i=0}^{N} ||\mathcal{J}_i(q_t) - j_i||^2 + \lambda ||\mathcal{J}_i(q_t) - \mathcal{J}_i(q_{t-1})||^2 \tag{1}$$
$$\text{s.t.} \quad q_{lower} \le q_t \le q_{upper},$$

where q_t is joint position at time t, $\mathcal{J}_i$ is the forward kinematics function of the i-th joint and j_i is the Cartesian coordinates of the hand joint which matches the i-th joint of the robot. q_{lower} and q_{upper} are the joint limits. We also encourage smoothness by incorporating a normalization term to penalize a large distance between q_t and q_{t-1}. We set the initial $q_0 = \frac{1}{2}(q_{lower} + q_{upper})$.

C. Continuous Grasping Function Learning

Our generative model is based on Conditional Variational Auto-Encoder (CVAE) [45], where we propose Continuous Grasping Function (CGF) to replace the traditional decoder. During training, both encoder and CGF are used to learn the grasping generation in a robot hand reconstruction task with object point clouds and robot joint positions as inputs; during inference, only CGF is used to generate continuous grasping with only object point clouds as input. The architecture is shown in Figure 3.

During training, given the object point cloud $\mathcal{P}^o \in \mathbb{R}^{N \times 3}$ (N is the number of points) and a sequence of joints positions $\{q_t\}, t \in \{0, 1, \cdots, T\}$ (T is the number of frames) as inputs, we employ PointNet [46] and MLP to extract object feature $\mathcal{F}^o \in \mathbb{R}^{1024}$ and hand features $\{\mathcal{F}_t^h\} \in \mathbb{R}^{256}$ respectively. All these features are then concatenated as $\mathcal{F}^{oh}$ for the encoder input. The outputs of the encoder are the mean $\mu \in \mathbb{R}^{256}$ and variance $\sigma^2 \in \mathbb{R}^{256}$ of the Gaussian distribution $\mathcal{Q}(z \mid \mu, \sigma^2)$. The latent code z is sampled from the distribution for the hand reconstruction.

Fig. 3: Network architecture. Our generative model takes object point cloud and a sequence of joint positions as input and recovers corresponding robot hands. The proposed CGF takes the latent code z, object feature, and the query time t as inputs to predict the corresponding joint position $\hat{q}_t$. $\oplus$ denotes concatenation.

After the encoding and sampling, we use our CGF to decode continuous grasping. Inspired by implicit functions [12], [13], [47] for shape representation, our proposed CGF maps the query time t to the joint position q_t. We concatenate latent code z and the object feature $\mathcal{F}^o$ with time t as the input for CGF. Specifically, CGF is a MLP f parameterized by θ which predicts the corresponding joint position $\hat{q}_t$:

$$\hat{q}_t = f(t, z, \mathcal{F}^o; \theta). \tag{2}$$

We reverse the time and define $t = 0$ as the end of the grasping for the convenience of implementation, i.e., when the robot's hand touches the object. And as t grows, the hand moves further away from the object. Given the predicted joint position $\hat{q}_t$, a differentiable forward kinematics layer $\mathcal{J}_i$ is utilized to get the Cartesian coordinate of the i-th joint.

The first objective function is the reconstruction error, which is defined as the $\mathcal{L}2$ distance between the joint positions as well as their Cartesian coordinates. We denote them as $\mathcal{L}_q = \sum_{t=0}^{T-1} \|\hat{q}_t - q_t\|_2^2$ and $\mathcal{L}_j = \sum_{t=0}^{T-1} \sum_{i=0} \|\mathcal{J}_i(\hat{q}_t) - \mathcal{J}_i(q_t)\|_2^2$ respectively. Following the training of VAE [24], we define a KL loss to encourage the latent code distribution $\mathcal{Q}\left(z \mid \mu, \sigma^2\right)$ to be close to the standard Gaussian distribution, which is achieved by maximizing the KL-Divergence as $\mathcal{L}_{KL} = -D_{KL}\left(Q\left(z \mid \mu, \sigma^2\right) \| \mathcal{N}(0, I)\right)$. We also introduce a contact loss at the end of the grasping to push the tips of robot hand to the object surface, which is achieved by minimizing distances to their closest object points $\mathcal{L}_{contact} = \sum_i \min_{p \in \mathcal{P}^o} \|\mathcal{J}_i(q_0) - p\|_2^2$ where i belongs to the indices of all tips. The full training loss is:

$$\mathcal{L} = \lambda_q \mathcal{L}_q + \lambda_j \mathcal{L}_j + \lambda_{KL} \mathcal{L}_{KL} + \lambda_c \mathcal{L}_{contact}, \tag{3}$$

where $\lambda_q, \lambda_j, \lambda_{KL}$ and λ_c are weights for various losses.

During inference, our CGF can easily sample a large number of diverse trajectories. We use PointNet to get the object feature $\mathcal{F}^o$ and sample a random latent code z from the standard Gaussian distribution, then with a time query sequence, our CGF can produce a continuous and natural trajectory. By sampling a large amount of z, we can find trajectories with a smaller gap between the human hand and robot hand, which guarantees both natural and successful trajectories.

D. PD Control with Feedforward

Different from previous works, we utilize an implicit function to output the target joint position q_d, and given the continuity property of the implicit function, we can easily get the derivative (target joint velocity) $\dot{q}_d$ and the second-order derivative (target joint acceleration) $\ddot{q}_d$. Thus we can use a more robust controller, PD control with feedforward in the form of

$$\tau = \text{ID}(\ddot{q}_d, q, \dot{q}) - K_p e - K_v \dot{e}, \tag{4}$$

where $e = q - q_d$, $\dot{e} = \dot{q} - \dot{q}_d$, τ is the joint torque and $\text{ID}(\ddot{q}_d, q, \dot{q})$ is the inverse dynamics. K_p and K_v are hyperparameters. q, $\dot{q}$, and $\ddot{q}$ are the joint position, velocity, and acceleration of the robot. As far as we know, our method is the first end-to-end manner that can directly get the derivative to use the PD control with feedforward.

IV. EXPERIMENTS

We conduct quantitative and qualitative evaluations in both simulator and real world. We show that by learning from human demonstrations, CGF is more efficient in finding successful grasping and our generation results can be transferred to the real robot with a higher success rate.

A. Experimental Setting

Datasets. We utilize the DexYCB dataset [44] to serve as human demonstrations. DexYCB contains 1,000 sequences of human grasping motions with 20 YCB objects [48]. We use 15 of them as training objects. We remove the handover process from the demonstrations, filter out all left-handed sequences and perform hand motion retargeting (Sec. III-B) to translate the hand motion into Allegro robot motion.

Baselines. We mainly compare CGF to two-step methods, i.e., grasping synthesis followed by motion planning. We provide two grasping synthesis baselines: GraspTTA [27] and GraspIt [49], where the former is also a generative model and the latter is based on searching. For GraspTTA, we apply the same hand motion retargeting to obtain the Allegro hand joint positions. Then, we utilize rapidly exploring random tree (RRT) [50] and cross-entropy method (CEM) [51] with model predictive control (MPC) [52] for planning the trajectory to reach the grasp pose, which are named GraspTTA+RRT and GraspTTA+CEM-MPC respectively. For GraspIt, due to the high Degree-of-Freedom of the dexterous hand, we firstly

Fig. 4: Qualitative evaluation in the simulation. Because of human demonstrations, our CGF generates a more natural and reasonable trajectory, which is helpful for the sim-to-real transfer. G is short for GraspTTA.

leverage the large set of grasping poses from ContactGrasp [6] to construct a low-dimensional subspace via EigenGrasp [53]. We then use GraspIt [49] and RRT for the grasping searching (including post-optimization) and motion planning respectively. We name it GraspIt+RRT. For the smoothness evaluation, we additionally perform linear interpolation between the beginning and ending joint positions generated by CGF and take it as a trivial baseline.

Implementation Details. For training CGF, we sample $2,000$ points on the object mesh as the input object point clouds. During training, we employ the Adam optimizer with the learning rate $5e-4$, where the learning rate is reduced by half per 500 epochs. The batch size is 32. The training takes 1000 epochs in total. The dimension of the latent code z is 256. CGF is a 4-layer MLP with channels (1281, 512, 256, 25) and internal ReLU activation. The loss weights are $\lambda_q = 1$, $\lambda_j = 10$, $\lambda_{KL} = 1e-3$ and $\lambda_c = 50$. We sample $10,000$ latent codes for CGF and output the trajectories. We then pick the successful trajectories in the simulator to evaluate in the real world and count the success rate. For simulation experiments, environments are built upon the SAPIEN [54] simulator.

For GraspTTA, we generate 200 grasp poses for each object. For CEM-MPC, we set popsize $M = 100$, time horizon $T = 5$, number of elites $e = 10$, and iterations $I = 2$. For GraspIt, we search for valid grasp poses with 100 different random seeds and use the contact energy as the objective function. For motion planning with RRT, we set 10,000 nodes in the tree and set step size $\epsilon = 0.01$ and probability of sampling $\beta = 0.5$.

B. Evaluation Metrics

Smoothness. To measure the continuity of the generated grasping, we propose to compare the smoothness. We first normalize joint positions by making all joints start at 0 and

Method	Smoothness - Joints ↓			Smoothness - Cartesian ↓		
	Pos	Vel (log)	Acc (log)	Pos	Vel (log)	Acc (log)
G + RRT	**3.40**	3.43	6.47	9.71	3.95	6.99
G + CEM-MPC	7.42	3.12	5.54	16.36	3.48	5.90
Ours	10.42	**2.06**	**3.51**	**6.69**	**1.99**	**3.47**
Linear Interpolation	1.00	-3.58	-1.99	1.96	0.70	1.38

TABLE I: For smoothness of joint positions and Cartesian coordinates, our CGF outperforms baselines by a large margin, which helps produce more natural trajectories and better control. G is short for GraspTTA.

end at 1 in all trajectories. Then we calculate discrete first-order and second-order gradients, i.e. velocity and acceleration. Smoothness is defined as the sum of the $\mathcal{L}1$ distances of position, velocity, and acceleration between neighboring frames. Note that the linear interpolation is the upper bound with a joint position smoothness of 1.0.

Cost per Successful Trajectory in Simulator. While sampling more grasps with a generative model or more random configurations with a planning algorithm could increase the probability of finding a successful trajectory, the cost should not be neglected. Thus, the average cost per successful trajectory is evaluated in the simulator for our method and baselines. The cost is defined as the number of environment steps for GraspTTA+CEM-MPC and our method or collision checks for GraspTTA+RRT, per successful trajectory. Additionally, wall-clock time is provided for a comprehensive evaluation.

Success Rate in the Real World. We also evaluate the success rate in the real world. Note the successful trajectories in the simulator do not guarantee success in the real world because of the sim-to-real gap. For our method and all the baselines, we collect 20 successful trajectories, deploy them in the real world and count the success rate. This metric reflects the sim-to-real transfer ability.

C. Simulated Experimental Results

Smoothness Evaluation. We compare the smoothness with three baselines and summarize the results in Tab. I. For better

Fig. 5: Grasping interpolation. We show the first frame and the last frame of the grasping trajectory. Yellow lines indicate the trajectory of the palm joint. Our method produces diverse grasping and the interpolation between them is also plausible. To the best of our knowledge, this result on interpolating both robot hand grasping pose and trajectory has not been shown before.

Method	Cost (log) / Succ. Traj. ↓		Time (s) / Succ. Traj. ↓	
	Seen	Unseen	Seen	Unseen
G + RRT	5.30*	4.97*	56.61	41.47
G + CEM-MPC	5.85	5.58	146.78	123.48
Ours	**4.30**	**4.19**	**11.70**	**8.31**
Linear Interpolation	-	-	-	-

TABLE II: Success cost evaluation. Since our method only requires fewer simulation steps to test a trajectory, the average cost and time for a successful trajectory is much lower than baselines. Note that we calculate the amount of collision detection for RRT. G is short for GraspTTA.

Method	Success Rate (%) ↑					
	Banana	Cleanser	Meat Can	Soup Can	Bottle	Ball
G + RRT	10.0	15.0	10.0	5.0	5.0	10.0
G + CEM-MPC	10.0	10.0	15.0	5.0	0.0	15.0
GI + RRT	60.0	65.0	50.0	50.0	65.0	40.0
Ours	**70.0**	**85.0**	**70.0**	**80.0**	**85.0**	**65.0**

TABLE III: Real-world experiments. For the successful trajectories in the simulation, our CGF has a higher success rate in the real world. G is short for GraspTTA and GI is short for GraspIt.

comparison, the smoothness for velocity and acceleration is in the log form. For both joint positions and Cartesian coordinates, our CGF outperforms baselines by a large margin. Note that for the joint position smoothness, our method does not show an advantage over the baseline. This is due to that humans tend to follow a natural curve rather than the shortest path. Nevertheless, the improvement in the smoothness of velocity and acceleration plays an important role in producing natural trajectories, suppressing vibration, and achieving high accuracy control [55], [56].

Success Cost Evaluation. The average cost and time per successful trajectory are presented in Table II. The number of simulation steps and collision detections are shown in the log form for better comparison. Our method, which directly executes target joint positions generated from CGF, obtains a significantly lower average cost and time per successful trajectory compared to GraspTTA+RRT and GraspTTA+CEM-MPC. The main reasons are: (i) Our method does not require excessive exploration like MPC-CEM or sampling numerous configurations like RRT; (ii) Our method produces more natural trajectories than two-step methods, yielding a higher probability for finding a successful trajectory. We also evaluate on unseen objects, our method still surpasses the baselines with a slight decrease in metrics. This decrease in cost for unseen objects can be attributed to that the difficulty of finding a successful trajectory is largely determined by the object geometry. The linear interpolation baseline, which generates the smoothest trajectory, never grasps objects successfully in

the simulator.

Qualitative Evaluation. We visualize the typical trajectories of the baselines and our method in Fig. 4. Since RRT is only planning the reachable target joint positions, the trajectory may not be natural. And a small perturbation in the execution process may cause it to collide with objects. CEM-MPC is not a long-time horizon method, which will make it fall into a local optimum quickly. As shown in Fig. 4, the middle finger is not in the ideal position. However, our CGF, learning from the human demonstration, could generate a much more natural and smooth trajectory. This is helpful for the sim-to-real transfer, which we will discuss in the next section.

Grasping Diversity. The ability to generate diverse outputs is one of the motivations for using CVAE. We perform interpolation in the latent space and show the results in Fig. 5. CGF produces diverse grasping and the interpolation between them is also plausible. We believe this is an interesting and potentially very useful property of our model. To the best of our knowledge, this result on interpolating both robot hand grasping pose and trajectory has not been shown before.

D. Real-World Robot Experiments

Setup. For the real-world robot experiments, we attached an Allegro hand on an X-Arm 6 robot. We select 6 real objects from YCB [48] which are *Banana*, *Bleach Cleanser*, *Ball Potted Meat Can*, *Tomato Soup Can* and *Mustard Bottle*, where the first 3 objects are unseen. The initial pose of the object is given and we use open-loop control for grasping.

Results with a Real Robot Hand. We evaluate the sim-to-real transfer ability of the trajectories generated by our method

Fig. 6: Real-world results on *Meat Can*, *Soup Can* and *Ball*. Our CGF successfully transfers the simulation trajectory to the real robot.

	Mass 1x	Mass 2x	Mass 3x
Velocity 1x	5.66%	4.15%	7.10%
Velocity 2x	14.95%	7.73%	2.10%
Velocity 3x	4.29%	1.65%	16.77%

TABLE IV: Ablation study on the PD control with feedforward. We report improvements on the success rate of the PD control with feedforward over the default PD control with different velocities and masses.

	Cost/S.T. - Noisy Points ↓		Cost/S.T. - Noisy Pose ↓	
s	Seen	Unseen	Seen	Unseen
0	4.30	4.19	4.30	4.19
0.001	4.28	4.26	4.31	4.38
0.01	4.32	4.38	4.38	4.56
0.1	4.39	4.67	4.48	5.64
1	4.61	4.89	-	-

TABLE V: Analysis Study for Noises. Our method is robust against noises on both point cloud and pose.

and baselines. For each object and method, we collect 20 successful trajectories in the simulator and deploy them in the real world. We report the success rate in Tab. III. Although all the trajectories succeeded in the simulator, our method has a much higher success rate in the real world than the baselines. We believe there are two main reasons leading to better sim-to-real transfer ability: (i) Learning from human demonstration can lead to more natural behavior trajectory; (ii) The use of implicit function also helps provide a continuous and more smooth trajectory. In contrast, the motion planning used in the 2-step procedure baselines often generates unnatural trajectories which reduces the success rate when deploying in the real world. Specifically, while GraspTTA [27] is able to generate grasp poses in the simulation, it does not ensure stable robotic grasping in the real world, and an unnatural trajectory approaching the grasp accumulates additional errors. On the other hand, GraspIt [49] is able to generate stable robot grasp pose in the real world, however, the 2-step procedure with it still performs worse than our method. We further provide a visualization of our successful trajectories in the real world in Fig. 6, which shows that our method could generate natural and human-like grasping.

E. Ablation Study

To demonstrate the advantages of continuous grasping, we ablate the PD control with feedforward (Sec. III-D) which relies on the derivatives of CGF. Note that baselines can not directly utilize PD control with feedforward since they are not continuous. We compare the performance of the PD control with feedforward to the default PD control in the simulator with different velocities and masses. Specifically, we set the speed to 1x, 2x and 3x and the mass of the robot to 1x, 2x and 3x. We combine them in pairs and report the improvements on the success rate in Tab. IV. We show that the PD control with feedforward has improvements in all various cases, which proves its robustness.

In addition, we analyze the robustness of CGF by adding noises on the input point cloud and the object pose. For the object point cloud $\mathcal{P}^o \in \mathbb{R}^{N \times 3}$, we add N independently sampled Gaussian noises $n \sim \mathcal{N}(0, \Sigma)$, where $\Sigma = \text{diag}(\sigma)$ is the diagonal covariance matrix. The noise scale was set with $\sigma = s\left(\max_i \mathcal{P}_i^o - \min_i \mathcal{P}_i^o\right)$, with s as the parameter controlling the noise scale and $\mathcal{P}_i^o$ is the i-th point in $\mathcal{P}^o$. For the object pose $T \in \mathbb{SE}(3)$, we also add Gaussian noises $n \sim \mathcal{N}(0, \Sigma)$ to its translation part, with the same distribution as that for the point clouds. The results in Tab. V show that our method is robust against both point cloud and pose noise, despite the higher cost of finding successful trajectories resulting from increased noise.

V. CONCLUSION

We propose a novel implicit function named Continuous Grasping Function to generate smooth and dense grasping trajectories. CGF is learned in the framework of a Conditional Variational AutoEncoder using 3D human demonstrations. During inference, we sample various grasping plans in the simulator and deploy the successful ones to the real robot.

By training on diverse human-object data, our method allows generalization to manipulate multiple objects. Compared to previous planning algorithms, CGF is more efficient and has a better sim-to-real generalization ability.

REFERENCES

[1] J. K. Salisbury and J. J. Craig, "Articulated hands: Force control and kinematic issues," *The International Journal of Robotics Research*, 1982.

[2] D. Rus, "In-hand dexterous manipulation of piecewise-smooth 3-d objects," *The International Journal of Robotics Research*, 1999.

[3] A. M. Okamura, N. Smaby, and M. R. Cutkosky, "An overview of dexterous manipulation," in *ICRA*, 2000.

[4] M. R. Dogar and S. S. Srinivasa, "Push-grasping with dexterous hands: Mechanics and a method," in *IROS*, 2010.

[5] J. Varley, J. Weisz, J. Weiss, and P. Allen, "Generating multi-fingered robotic grasps via deep learning," in *IROS*, 2015.

[6] S. Brahmbhatt, A. Handa, J. Hays, and D. Fox, "Contactgrasp: Functional multi-finger grasp synthesis from contact," in *IROS*, 2019.

[7] Q. Lu, K. Chenna, B. Sundaralingam, and T. Hermans, "Planning multi-fingered grasps as probabilistic inference in a learned deep network," in *Robotics Research*, 2020.

[8] P. Mandikal and K. Grauman, "Learning dexterous grasping with object-centric visual affordances," in *ICRA*, 2021.

[9] Y. Qin, Y.-H. Wu, S. Liu, H. Jiang, R. Yang, Y. Fu, and X. Wang, "Dexmv: Imitation learning for dexterous manipulation from human videos," *ECCV*, 2022.

[10] Y. Chen, S. Liu, and X. Wang, "Learning continuous image representation with local implicit image function," in *CVPR*, 2021.

[11] E. Dupont, Y. W. Teh, and A. Doucet, "Generative models as distributions of functions," *arXiv*, 2021.

[12] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *CVPR*, 2019.

[13] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *CVPR*, 2019.

[14] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020.

[15] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *NeurIPS*, 2015.

[16] N. C. Dafle, A. Rodriguez, R. Paolini, B. Tang, S. S. Srinivasa, M. Erdmann, M. T. Mason, I. Lundberg, H. Staab, and T. Fuhlbrigge, "Extrinsic dexterity: In-hand manipulation with external forces," in *ICRA*, 2014.

[17] B. Calli, A. Kimmel, K. Hang, K. Bekris, and A. Dollar, "Path planning for within-hand manipulation over learned representations of safe states," in *International Symposium on Experimental Robotics*. Springer, 2018.

[18] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," *arXiv*, 2018.

[19] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, "Solving rubik's cube with a robot hand," *arXiv*, 2019.

[20] W. Huang, I. Mordatch, P. Abbeel, and D. Pathak, "Generalization in dexterous manipulation via geometry-aware multi-task learning," *arXiv*, 2021.

[21] T. Chen, J. Xu, and P. Agrawal, "A system for general in-hand object re-orientation," in *CORL*, 2022.

[22] V. Kumar, E. Todorov, and S. Levine, "Optimal control with learned local models: Application to dexterous manipulation," in *ICRA*. IEEE, 2016.

[23] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar, "Deep dynamics models for learning dexterous manipulation," in *CORL*, 2020.

[24] S. Andrews and P. G. Kry, "Goal directed multi-finger manipulation: Control policies and analysis," *Computers & Graphics*, 2013.

[25] D. Morrison, J. Leitner, and P. Corke, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," in *RSS*, 2018.

[26] K. Karunratanakul, J. Yang, Y. Zhang, M. J. Black, K. Muandet, and S. Tang, "Grasping field: Learning implicit representations for human grasps," in *3DV*, 2020.

[27] H. Jiang, S. Liu, J. Wang, and X. Wang, "Hand-object contact consistency reasoning for human grasps generation," in *ICCV*, 2021.

[28] K. Hsiao and T. Lozano-Perez, "Imitation learning of whole-body grasps," in *IROS*, 2006.

[29] Y. Ye and C. K. Liu, "Synthesis of detailed hand manipulations using contact sampling," *ACM Transactions on Graphics (TOG)*, 2012.

[30] Y. Wu, J. Wang, Y. Zhang, S. Zhang, O. Hilliges, F. Yu, and S. Tang, "Saga: Stochastic whole-body grasping with contact," *arXiv*, 2021.

[31] O. Taheri, V. Choutas, M. J. Black, and D. Tzionas, "Goal: Generating 4d whole-body motion for hand-object grasping," in *CVPR*, 2022.

[32] K. Schmeckpeper, O. Rybkin, K. Daniilidis, S. Levine, and C. Finn, "Reinforcement learning with videos: Combining offline observations with interaction," *arXiv*, 2020.

[33] L. Shao, T. Migimatsu, Q. Zhang, K. Yang, and J. Bohg, "Concept2robot: Learning manipulation concepts from instructions and human demonstrations," in *RSS*, 2020.

[34] S. Young, D. Gandhi, S. Tulsiani, A. Gupta, P. Abbeel, and L. Pinto, "Visual imitation made easy," *arXiv*, 2020.

[35] A. Gupta, C. Eppner, S. Levine, and P. Abbeel, "Learning dexterous manipulation for a soft robotic hand from human demonstrations," in *IROS*, 2016.

[36] S. Christen, S. Stevšić, and O. Hilliges, "Guided deep reinforcement learning of control policies for dexterous human-robot interaction," in *ICRA*, 2019.

[37] A. Sivakumar, K. Shaw, and D. Pathak, "Robotic telekinesis: learning a robotic hand imitator by watching humans on youtube," *arXiv*, 2022.

[38] Y. Qin, H. Su, and X. Wang, "From one hand to multiple hands: Imitation learning for dexterous manipulation from single-camera teleoperation," *RA-L*, 2022.

[39] X. Li, S. De Mello, X. Wang, M.-H. Yang, J. Kautz, and S. Liu, "Learning continuous environment fields via implicit functions," *ICLR*, 2022.

[40] A. Simeonov, Y. Du, A. Tagliasacchi, J. B. Tenenbaum, A. Rodriguez, P. Agrawal, and V. Sitzmann, "Neural descriptor fields: Se (3)-equivariant object representations for manipulation," *ICRA*, 2022.

[41] Y. Li, S. Li, V. Sitzmann, P. Agrawal, and A. Torralba, "3d neural scene representations for visuomotor control," in *CORL*, 2022.

[42] Z. Jiang, Y. Zhu, M. Svetlik, K. Fang, and Y. Zhu, "Synergies between affordance and geometry: 6-dof grasp detection via implicit representations," *arXiv*, 2021.

[43] M. Adamkiewicz, T. Chen, A. Caccavale, R. Gardner, P. Culbertson, J. Bohg, and M. Schwager, "Vision-only robot navigation in a neural radiance world," *RA-L*, 2022.

[44] Y.-W. Chao, W. Yang, Y. Xiang, P. Molchanov, A. Handa, J. Tremblay, Y. S. Narang, K. Van Wyk, U. Iqbal, S. Birchfield *et al.*, "Dexycb: A benchmark for capturing hand grasping of objects," in *CVPR*, 2021.

[45] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," *NeurIPS*, 2015.

[46] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017.

[47] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," in *CVPR*, 2019.

[48] 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 *ICRA*, 2015.

[49] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *Robotics & Automation Magazine*, 2004.

[50] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[51] R. Rubinstein, "The cross-entropy method for combinatorial and continuous optimization," *Methodology and Computing in Applied Probability*, 1999.

[52] J. M. Maciejowski, *Predictive control: with constraints*. Pearson education, 2002.

[53] M. Ciocarlie, C. Goldfeder, and P. Allen, "Dexterous grasping via eigengrasps: A low-dimensional approach to a high-complexity problem," in *RSS Workshop*, 2007.

[54] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang *et al.*, "Sapien: A simulated part-based interactive environment," in *CVPR*, 2020.

[55] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *Journal of neuroscience*, 1985.

[56] A. Piazzi and A. Visioli, "Global minimum-jerk trajectory planning of robot manipulators," *IEEE transactions on industrial electronics*, 2000.