Multi-Task Real Robot Learning with Generalizable Neural Feature Fields

Anonymous Author(s)
Affiliation
Address
email

Abstract: It is a long-standing problem in robotics to develop agents capable of executing diverse manipulation tasks from visual observations in unstructured real-world environments. To achieve this goal, the robot needs to have a comprehensive understanding of the 3D structure and semantics of the scene. In this work, we present **GNFactor**, a visual behavior cloning agent for multi-task robotic manipulation with **G**eneralizable **N**eural feature **F**ields. GNFactor jointly optimizes a neural radiance field (NeRF) as a reconstruction module and a Perceiver Transformer as a decision-making module, leveraging a shared deep 3D voxel representation. To incorporate semantics in 3D, the reconstruction module incorporates a vision-language foundation model (e.g., Stable Diffusion) to distill rich semantic information into the deep 3D voxel. We evaluate GNFactor on 3 real robot tasks and perform detailed ablations on 10 RLBench tasks with a limited number of demonstrations. We observe a substantial improvement of GNFactor over current state-of-the-art methods in seen and unseen tasks, demonstrating the strong generalization ability of GNFactor. Videos are available at gnfactor-robot.github.io.

Keywords: Robotic Manipulation, Neural Radiance Field, Behavior Cloning

1 Introduction

One major goal of introducing learning into robotic manipulation is to enable the robot to effectively handle unseen objects and successfully tackle various tasks in new environments. In this paper, we focus on using imitation learning with a few demonstrations for multi-task manipulation. Using imitation learning helps avoid complex reward design and training can be directly conducted on the real robot without creating its digital twin in simulation [1, 2, 3, 4]. This enables policy learning on diverse tasks in complex environments, based on users' instructions (see Figure 1). However, working with a limited number of demonstrations presents great challenges in terms of generalization. Most of these challenges arise from the need to comprehend the 3D structure of the scene, understand the semantics and functionality of objects, and effectively follow task instructions based on visual cues. Therefore, a comprehensive and informative visual representation of the robot's observations serves as a crucial foundation for generalization.

The development of visual representation for robot learning has mainly focused on learning within a 2D plane. Self-supervised objectives are leveraged to pre-train the representation from the 2D image observation [6, 7, 8] or jointly optimized with the policy gradients [9, 10, 11]. While these approaches improve sample efficiency and lead to more robust policies, they are mostly applied to relatively simple manipulation tasks. To tackle more complex tasks requiring geometric understanding (*e.g.*, object shape and pose) and with occlusions, 3D visual representation learning has been recently adopted with robot learning [11, 12]. For example, Driess et al. [12] train the 3D scene representation by using NeRF and view synthesis to provide supervision. While it shows effectiveness over tasks requiring geometric reasoning such as hanging a cup, it only handles the simple scene structure with heavy masking in a single-task setting. More importantly, without a semantic

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

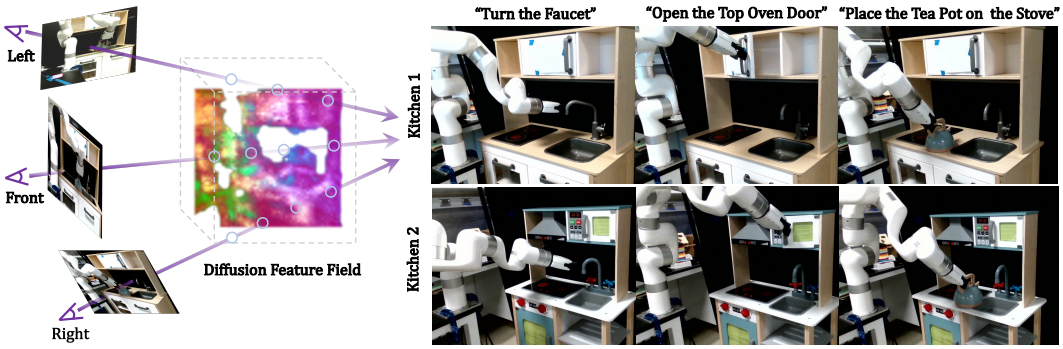


Figure 1: **Left:** Three camera views used in the real robot setup to reconstruct the feature field generated by Stable Diffusion [5]. We segment the foreground feature for better illustration. **Right:** Three language-conditioned real robot tasks across two different kitchens.

understanding of the scene, it will be very challenging for the robot to follow the user's language instructions.

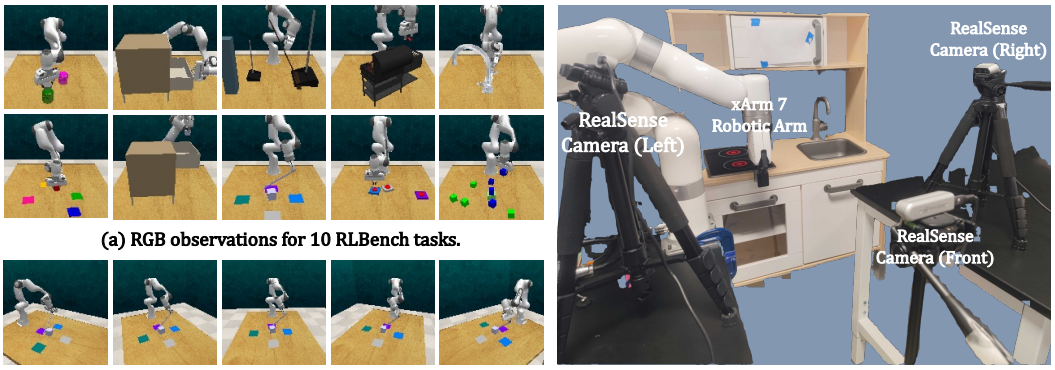
In this paper, we introduce learning a language-conditioned policy using a novel representation leveraging both 3D and semantic information for multi-task manipulation. We train **G**eneralizable **N**eural Feature **F**ields (**GNF**) which distills pre-trained semantic features from 2D foundation models into the Neural Radiance Fields (NeRFs). We conduct policy learning upon this representation, leading to our model **GNFactor**. It is important to note that GNFactor learns an encoder to extract scene features in a feed-forward manner, instead of performing per-scene optimization in NeRF. Given a single RGB-D image observation, our model will encode it into a 3D semantic volumetric feature, which is then processed by a Perceiver Transformer [13] architecture for action prediction. To conduct multi-task learning, the Perceiver Transformer will take in language instructions to get task embedding, and reason the relations between the language and visual semantics for manipulation.

There are two branches of training in our framework (see Figure 3): (i) *GNF training*. Given the collected demonstrations, we train the Generalizable Neural Feature Fields using view synthesis with volumetric rendering. Besides rendering the RGB pixels, we also render the features of the foundation models in 2D space. The GNF will learn from both pixel and feature reconstruction at the same time. To provide supervision for feature reconstruction, we apply a vision foundation model (e.g., pre-trained Stable Diffusion model [5]) to extract the 2D feature from the input view as the ground-truth. In this way, we can distill the semantic features into the 3D space in GNF. (ii) *GNFactor joint training*. Building on the 3D volumetric feature jointly optimized by the learning objectives of GNF, we conduct behavior cloning to train the whole model end-to-end.

For evaluation, we conduct real-robot experiments on three distinct tasks across two different kitchens (see Figure 1). We successfully train a single policy that effectively addresses these tasks in different scenes, yielding significant improvements over the baseline method PerAct [3]. We also conduct comprehensive evaluations using 10 RLBench simulated tasks [14] and 6 designed generalization tasks. We observe that GNFactor outperforms PerAct with an average improvement of 1.55x and 1.57x, consistent with the significant margin observed in the real-robot experiments.

2 Related Work

Multi-Task Robotic Manipulation. Recent works in multi-task robotic manipulation have led to significant progress in the execution of complex tasks and the ability to generalize to new scenarios [15, 2, 1, 16, 17, 3, 18, 19]. Notable methods often involve the use of extensive interaction data to train multi-task models [2, 1, 16, 17]. For example, RT-1 [1] underscores the benefits of task-agnostic training, demonstrating superior performance in real-world robotic tasks across a variety of datasets. To reduce the need for extensive demonstrations, methods that utilize keyframes - which encode the initiation of movement - have proven to be effective [20, 21, 22, 23, 24]. PerAct [3] employs the Perceiver Transformer [13] to encode language goals and voxel observations and shows its effectiveness in real robot experiments. In this work, we utilize the same action prediction frame-



(a) RGB observations for 10 RLBench tasks.

(b) Sampled views for GNF training in simulation.

(c) Real robot setup.

Figure 2: **Simulation environments and the real robot setup.** We show the RGB observations for our 10 RLBench tasks in Figure (a), the sampled views for GNF in Figure (b), and the real robot setup in Figure (c).

work as PerAct while we focus on improving the generalization ability of this framework by learning a generalizable volumetric representation under limited data.

3D Representations for Reinforcement/Imitation Learning (RL/IL). To improve manipulation policies by leveraging visual information, numerous studies have concentrated on enhancing 2D visual representations [8, 7, 6, 25], while for addressing more complex tasks, the utilization of 3D representations becomes crucial. Ze et al. [11] incorporates a deep voxel-based 3D autoencoder in motor control, demonstrating improved sample efficiency compared to 2D representation learning methods. Driess et al. [12] proposes to first learn a state representation by NeRF and then use the frozen state for downstream RL tasks. While this work shows the initial success of utilizing NeRF in RL, its applicability in real-world scenarios is constrained due to various limitations: *e.g.*, the requirement of object masks, the absence of a robot arm, and the lack of scene structure. The work closest to ours is SNeRL [26], which also utilizes a vision foundation model in NeRF. However, similar to NeRF-RL, SNeRL masks the scene structure to ensure functionality and the requirement for object masks persists, posing challenges for its application in real robot scenarios. Our proposed GNFactor, instead, handles challenging muti-task real-world scenarios, demonstrating the potential for real robot applications.

Neural Radiance Fields (NeRFs). Neural fields have achieved great success in novel view synthesis and scene representation learning these years [27, 28, 29, 30, 31, 32, 33]. NeRF [30] stands out for achieving photorealistic view synthesis by learning an implicit function of the scene, while NeRF requires per-scene optimization and is thus hard to generalize. Many following methods [34, 35, 36, 37, 38, 39, 40] propose more generalizable NeRFs. PixelNeRF [41] and CodeNeRF [35] encode 2D images as the input of NeRFs, while TransINR [34] leverages a vision transformer to directly infer NeRF parameters. N3F [42], DFF [43], FeatureNeRF [44], and LERF [45] utilize pre-trained vision foundation models such as DINO [46] as supervision besides the RGB image, which thus enabling the NeRF to learn the generalizable feature. In this work, we incorporate generalizable NeRF to reconstruct different views in RGB and embeddings from a pretrained Stable Diffusion model [5].

3 Method

In this section, we detail the proposed GNFactor, a multi-task agent with a 3D volumetric representation for real-world robotic manipulation. GNFactor is composed of a volumetric rendering module and a 3D policy module, sharing the same deep volumetric representation. The volumetric rendering module learns a Generalizable Neural Feature Field (GNF), to reconstruct the RGB image from cameras and the embedding from a vision-language foundation model, *e.g.*, Stable Diffusion [5]. The task-agnostic nature of the vision-language embedding enables the volumetric representation to learn generalizable features via neural rendering and thus helps the 3D policy module better handle multi-task robotic manipulation. The task description is encoded with CLIP [47] to obtain the task embedding T. An overview of GNFactor is shown in Figure 3.

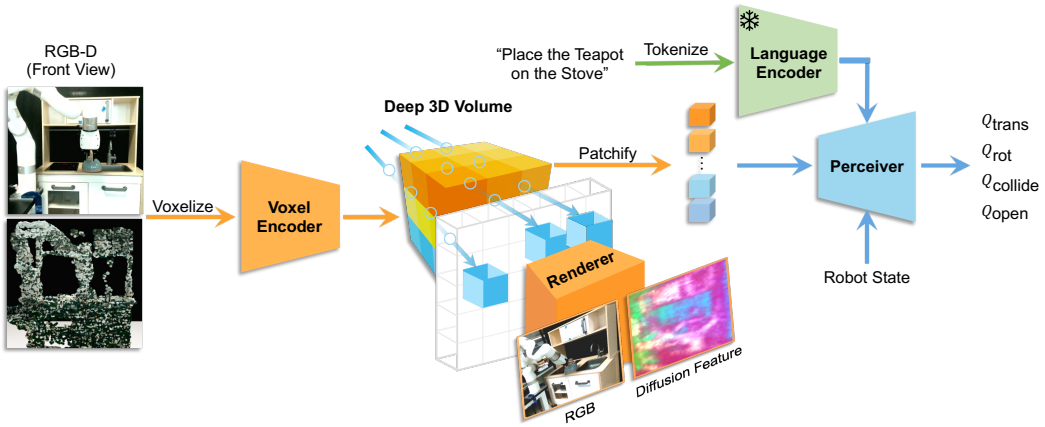


Figure 3: **Overview of GNFactor.** GNFactor takes an RGB-D image as input and encodes it using a voxel encoder to transform it into a feature in deep 3D volume. This volume is then shared by two modules: volumetric rendering (Renderer) and robot action prediction (Perceiver). These two modules are jointly trained, which optimizes the shared features to not only reconstruct vision-language embeddings (Diffusion Feature) and other views (RGB), but also to estimate accurate Q-values (Q_{trans}, Q_{rot}, Q_{collide}, Q_{open}).

3.1 Problem Definition

Due to the inefficiency of continuous action prediction and the extensive data requirements that come with it, we reformulate the behavior cloning problem as a *keyframe-prediction* problem [3, 48]. We first extract keyframes from expert demonstrations using the following metric: a frame in the trajectory is a keyframe when joint velocities approach zero and the gripper's open state remains constant. The model is then trained to predict the subsequent keyframe based on current observations. This formulation effectively transforms the continuous control problem into a discretized keyframe-prediction problem, delegating the internal procedures to the motion planner.

To effectively address complex real-world scenarios, we structure the observation space as a 3D voxel space $\mathcal{O} \in \mathbb{R}^{100^3 \times 3}$, as opposed to the commonly used 2D images [1, 2, 7, 8]. The 3D voxel observation originates from an RGB-D image captured by a **single front camera** with known extrinsic and intrinsic parameters, ensuring our method's practical applicability in the real world. In addition to the front camera view used for policy training, we also gather additional k views for training the GNF. We collect only RGB images for these additional views instead of RGB-D images. In real-world scenarios, we use $k = 2$, while in simulated environments, we set $k = 19$.

The action of the robot arm is represented by translation $a_{\text{trans}} \in \mathbb{R}^3$, rotation $a_{\text{rot}} \in \mathbb{R}^{(360/5)\times 3}$, gripper openness $a_{\text{open}} \in [0, 1]$, and collision avoidance $a_{\text{collision}} \in [0, 1]$. For the rotation a_{rot}, each rotation axis is discretized into $R = 5$ bins. The collision avoidance parameter $a_{\text{collision}}$ instructs the motion planner regarding the necessity to avoid collisions, which is crucial as our tasks encompasses both contact-based and non-contact-based motions.

3.2 Learning Volumetric Representations with Generalizable Neural Feature Fields

In our initial step, we transform the RGB-D image into a 100^3 voxel. Then the 3D voxel encoder encodes this 3D voxel and outputs our volumetric representation $v \in \mathbb{R}^{100^3 \times 128}$. To enhance the volumetric representation v with structural knowledge and language semantics, we learn a Generalizable Neural Feature Field (GNF) that takes the deep volume v as the scene representation and the model is learned by reconstructing the additional views and the features predicted by a 2D vision-language foundation model [5]. The entire neural rendering process is described as follows.

We denote $v_{\mathbf{x}} \in \mathbb{R}^{128}$ as the sampled 3D feature for the 3D point $\mathbf{x}$ using the volumetric representation v. $v_{\mathbf{x}}$ is formed with trilinear interpolation due to the discretized nature of the volume v. Our GNF primarily consists of three functions: (i) one density function $\sigma(\mathbf{x}, v_{\mathbf{x}}) : \mathbb{R}^3 \mapsto \mathbb{R}_+$ that maps the 3D point $\mathbf{x}$ and the 3D feature $v_{\mathbf{x}}$ to the density σ, (ii) one RGB function $\mathbf{c}(\mathbf{x}, \mathbf{d}, v_{\mathbf{x}}) : \mathbb{R}^{3\times 3} \mapsto$

$\mathbb{R}^3$ that maps the 3D point $\mathbf{x}$, the view direction $\mathbf{d}$, and the 3D feature $v_\mathbf{x}$ to color, and (iii) one vision-language embedding function $\mathbf{f}(\mathbf{x}, \mathbf{d}, v_\mathbf{x}) : \mathbb{R}^{3\times 3} \mapsto \mathbb{R}^{512}$ that maps the 3D point $\mathbf{x}$, the view direction $\mathbf{d}$, and the 3D feature $v_\mathbf{x}$ to the vision-language embedding. In Figure 3, the corresponding components of these three functions are illustrated. Specifically, the transformation from the Deep 3D Volume to the 2D input for the Renderer corresponds to the density function; the output head of the Renderer, which estimates other views (RGB) corresponds to the RGB function; and the second output head of the Renderer, which estimates the vision-language embedding (Diffusion Feature), corresponds to the vision-language embedding function. Given a pixel's camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, which is defined by the camera origin $o \in \mathbb{R}^3$, view direction $\mathbf{d}$ and depth t with bounds $[t_n, t_f]$, the estimated color and embedding of the ray can be calculated by:

$$\begin{aligned} \hat{\mathbf{C}}(\mathbf{r}, v) &= \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t), v_{\mathbf{x}(t)})\mathbf{c}(\mathbf{r}(t), \mathbf{d}, v_{\mathbf{x}(t)})\mathrm{d}t\,, \\ \hat{\mathbf{F}}(\mathbf{r}, v) &= \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t), v_{\mathbf{x}(t)})\mathbf{f}(\mathbf{r}(t), \mathbf{d}, v_{\mathbf{x}(t)})\mathrm{d}t\,, \end{aligned} \tag{1}$$

where $T(t) = \exp\left(-\int_{t_n}^{t} \sigma(s)\mathrm{d}s\right)$. The integral is approximated with numerical quadrature in the implementation. Our GNF is then optimized to reconstruct the RGB image and the vision-language embedding from multiple views and diverse scenes by minimizing the following loss:

$$\mathcal{L}_{\text{recon}} = \sum_{\mathbf{r}\in\mathcal{R}} \|\mathbf{C}(\mathbf{r}) - \hat{\mathbf{C}}(\mathbf{r})\|_2^2 + \lambda_{\text{feat}}\|\mathbf{F}(\mathbf{r}) - \hat{\mathbf{F}}(\mathbf{r})\|_2^2\,, \tag{2}$$

where $\mathbf{C}(\mathbf{r})$ is the ground truth color, $\mathbf{F}(\mathbf{r})$ is the ground truth vision-language embedding generated by Stable Diffusion, $\mathcal{R}$ is the set of rays generated from camera poses, and λ_{feat} is the weight for the embedding reconstruction loss. For efficiency, we sample b_{ray} rays given one target view, instead of reconstructing the entire image. To help the GNF training, we use a coarse-to-fine hierarchical structure as the original NeRF [30] and apply depth-guided sampling [49] in the "fine" network.

3.3 Action Prediction with Volumetric Representations

The volumetric representation v is optimized not only to achieve reconstruction of the GNF module, but also to predict the desired action for accomplishing manipulation tasks within the 3D policy. As such, we jointly train the representation v to satisfy the objectives of both the GNF and the 3D policy module. In this section, we elaborate the training objective and the architecture of the 3D policy.

We employ a Perceiver Transformer [3] to handle the high-dimensional multi-modal input, *i.e.*, the 3D volume, the robot's proprioception, and the language feature. We first condense the shared volumetric representation v into a volume of size $20^3 \times 128$, and flatten the 3D volume into a sequence of small cubes of size 8000×128. The robot's proprioception is projected into a 128-dimensional space and concatenated with the volume sequence for each cube, resulting in a sequence of size 8000×256. We then project the language token features from CLIP into the same dimensions (77×256) and concatenate these features with a combination of the 3D volume, the robot's proprioception state, and the CLIP token embedding. The result is a sequence with dimensions of 8077×256.

This sequence is combined with a learnable positional encoding and passed through the Perceiver Transformer, which outputs a sequence of the same size. We remove the last 77 features for the ease of voxelization [3] and reshape the sequence back to a voxel of size $20^3 \times 256$. This voxel is then upscaled to $100^3 \times 128$ and referred to as v_{PT}. v_{PT} is shared across three action prediction heads (Q_{trans}, Q_{rot}, Q_{collide} in Figure 3) to determine the final robot actions at the same scale as the observation space. To retain the learned features from GNF training, we create a skip connection between our volumetric representation v and v_{PT}. The combined volume feature (v, v_{PT}) is used to predict a 3D Q-function $\mathcal{Q}_{\text{trans}}$ for translation, as well as Q-functions for other robot operations like gripper openness ($\mathcal{Q}_{\text{open}}$), rotation ($\mathcal{Q}_{\text{rot}}$), and collision avoidance ($\mathcal{Q}_{\text{collide}}$). The model optimizes the cross-entropy loss like a classifier,

$$\mathcal{L}_{\text{action}} = -\mathbb{E}_{Y_{\text{trans}}}\left[\log \mathcal{V}_{\text{trans}}\right] - \mathbb{E}_{Y_{\text{rot}}}\left[\log \mathcal{V}_{\text{rot}}\right] - \mathbb{E}_{Y_{\text{open}}}\left[\log \mathcal{V}_{\text{open}}\right] - \mathbb{E}_{Y_{\text{collide}}}\left[\log \mathcal{V}_{\text{collide}}\right]\,, \tag{3}$$

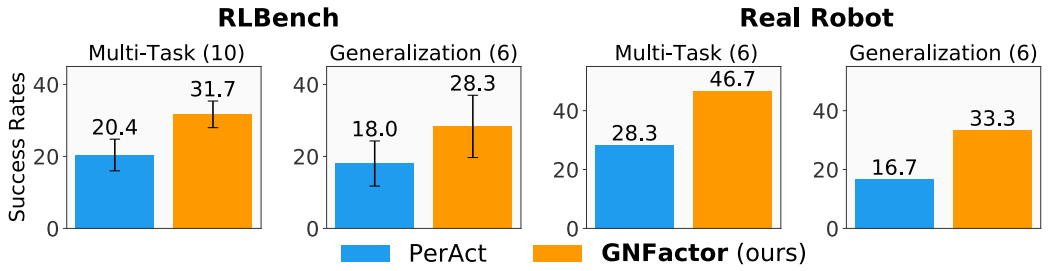


Figure 4: **Main experiment results.** We present the average success rates in both the multi-task and generalization settings across RLBench tasks and real robot tasks. The error bar represents one standard deviation. The number in the bracket denotes the number of tasks.

where $\mathcal{V}_i = \text{softmax}(\mathcal{Q}_i)$ for $\mathcal{Q}_i \in [\mathcal{Q}_{\text{trans}}, \mathcal{Q}_{\text{open}}, \mathcal{Q}_{\text{rot}}, \mathcal{Q}_{\text{collide}}]$ and $Y_i \in [Y_{\text{trans}}, Y_{\text{rot}}, Y_{\text{open}}, Y_{\text{collide}}]$ is the ground truth one-hot encoding. The overall learning objective for GNFactor is as follows:

$$\mathcal{L}_{\text{GNFactor}} = \mathcal{L}_{\text{action}} + \lambda_{\text{recon}} \mathcal{L}_{\text{recon}} \,, \tag{4}$$

where λ_{recon} is the weight for the reconstruction loss to balance the scale of different objectives. To train the GNFactor, we employ a joint training approach in which the GNF and 3D policy module are optimized jointly, without any pre-training. From our empirical observation, this approach allows for better fusion of information from the two modules when learning the shared features.

4 Experiments

In this section, we conduct experiments to answer the following questions: (i) Can GNFactor surpass the baseline model in simulated environments? (ii) Can GNFactor generalize to novel scenes in simulation? (iii) Does GNFactor learn a superior policy that handles real robot tasks in two different kitchens with noisy and limited real-world data? (iv) What are the crucial factors in GNFactor to ensure the functionality of the entire system? Our concluded results are given in Figure 4.

4.1 Experiment Setup

For the sake of reproducibility and benchmarking, we conduct our primary experiments in RLBench simulated tasks. Furthermore, to show the potential of GNFactor in the real world, we design a set of real robot experiments across two kitchens. We compare our GNFactor with the strong language-conditioned multi-task agent PerAct [3] in both simulation and the real world, emphasizing the universal functionality of GNFactor. Figure 2 shows our simulation tasks and the real robot setup. We briefly describe the tasks and details are left in Appendix A.

Simulation. We select 10 challenging language-conditioned manipulation tasks from the RLBench tasksuites [14]. Each task has at least two variations, totaling 166 variations. These variations encompass several types, such as variations in shape and color. Therefore, to achieve high success rates with very limited demonstrations, the agent needs to learn generalizable knowledge about manipulation instead of merely overfitting to the given demonstrations. We use the RGB-D image of size $128 \times 128 \times 3$ from the single front camera as the observation. To train the GNF, we also add additional 19 camera views to provide RGB images as supervision.

Real robot. We use the xArm7 robot with a parallel gripper in real robot experiments. We set up two toy kitchen environments to make the agent generalize manipulation skills across the scenes and designed three manipulation tasks, including *open the oven*, *turn the faucet*, and *relocate the teapot*, as shown in Figure 1. We set up three RealSense cameras around the robot. Among the three cameras, the front one captures the RGB-D observations for the policy training and the left/right one provides the RGB supervision for the GNF training.

Expert Demonstrations. We collect 20 demonstrations for each RLBench task with the motion planner. The task variation is uniformly sampled. We collect 5 demonstrations for each real robot task using a VR controller. Details for collection remain in Appendix C.

Generalization tasks. To further show the generalization ability of GNFactor, we design additional 6 simulated tasks and 3 real robot tasks based on the original training tasks and add task distractors.

Table 1: **Multi-task test results on RLBench.** We evaluate 25 episodes for each checkpoint on 10 tasks across 3 seeds and report the success rates of the finally saved checkpoints. Our method outperforms the most competitive baseline PerAct [3] with an average improvement of **1.55**x.

Method / Task	close jar	open drawer	sweep to dustpan	meat off grill	turn tap	**Average**
PerAct	18.7±8.2	54.7±18.6	0.0±0.0	40.0±17.0	38.7±6.8	
GNFactor	**25.3±6.8**	**76.0±5.7**	**28.0±15.0**	**57.3±18.9**	**50.7±8.2**	
Method / Task	slide block	put in drawer	drag stick	push buttons	stack blocks	
PerAct	18.7±13.6	**2.7±3.3**	5.3±5.0	18.7±12.4	**6.7±1.9**	20.4
GNFactor	**20.0±15.0**	0.0±0.0	**37.3±13.2**	**18.7±10.0**	4.0±3.3	**31.7**

Table 2: **Generalization to unseen tasks on RLBench.** We evaluate 20 episodes for each task with the final checkpoint across 3 seeds. We denote "L" as a larger object, "S" as a smaller object, "N" as a new position, and "D" as adding a distractor. Our method outperforms PerAct with an average improvement of **1.57**x.

Method / Task	drag (D)	slide (L)	slide (S)	open (n)	turn (N)	push (D)	**Average**
PerAct	6.6±4.7	33.3±4.7	5.0±4.1	25.0±10.8	18.3±6.2	20.0±7.1	18.0
GNFactor	**46.7±30.6**	25.0±4.1	6.7±6.2	**31.7±6.2**	**28.3±2.4**	**31.7±2.4**	**28.3**

Training details. One agent is trained with two NVIDIA RTX3090 GPU for 2 days (100k iterations) with a batch size of 2. The shared voxel encoder of GNFactor is implemented as a lightweight 3D UNet with only 0.3M parameters. The Perceiver Transformer keeps the same number of parameters as PerAct [3] (25.2M parameters), making our comparison with PerAct fair.

4.2 Simulation Results

We report the success rates for multi-task tests on RLBench in Table 1 and for generalization to new environments in Table 2. We conclude our observations as follows:

Dominance of GNFactor over PerAct for multi-task learning. As shown by Table 1 and Figure 4, GNFactor achieves higher success rates across various tasks compared to PerAct, particularly excelling in challenging long-horizon tasks. For example, in `sweep to dustpan` task, the robot needs to first pick up the broom and use the broom to sweep the dust into the dustpan. We find that GNFactor achieves an impressive success rate of 28.0%, while PerAct could not succeed at all. In simpler tasks like `open drawer` where the robot only pulls the drawer out, both GNFactor and PerAct perform reasonably well, with success rates of 54.7% and 76.0% respectively.

Generalization ability of GNFactor to new tasks. In Table 2, we observe that the change made on the environments such as distractors impacts all the agents negatively, while GNFactor shows better capability of generalization on 5 out of 6 tasks compared to PerAct. We also find that for some challenging variations such as the smaller block in the task `slide (S)`, both GNFactor and PerAct struggle to handle. This further emphasizes the importance of robust generalization skills.

Ablations. We summarize the key components in GNFactor that contribute to the success of the volumetric representation in Table 4. From the ablation study, we gained several insights: (i) Striking a careful balance between the neural rendering loss and the action prediction loss is critical for optimal performance. (ii) The volumetric representation benefits from Diffusion features and depth-guided sampling, where the depth prior is utilized to enhance the sampling quality in neural rendering. This suggests that there is a correlation between reconstruction quality and performance in manipulation tasks, which contributes to the success of the GNFactor. (iii) Utilizing information from multiple views proves to be beneficial for the single-view decision module.

Table 4: **Ablations.** We report the averaged success rates on 10 RLBench tasks. "DGS" is short for depth-guided sampling.

Ablation	Success Rate (%)
GNFactor	**36.8**
w/o. Diffusion	30.0
Diffusion $\rightarrow$ DINO	30.4
w/o. DGS	29.2
$k = 19 \rightarrow 9$	33.2
$\lambda_{\text{feat}} = 0.01 \rightarrow 1.0$	35.2
$\lambda_{\text{recon}} = 0.01 \rightarrow 1.0$	35.2

Furthermore, we provide a novel view synthesis in the real world, generated by GNFactor in Figure 5. We selected three novel views from the same scene and evaluate the quality of the rendered images. The rendered views are somewhat blurred since the volumetric presentation learned by GNFactor is optimized to minimize both the neural rendering loss as well as the action prediction loss. This learned representation not only provides the GNFactor with task generalizability, as shown in Table 3, but also captures the components as illustrated in Figure 5. Notably, for the view synthesis in the real world, we do not have access to ground truth point clouds for either training or testing.

Table 3: **Multi-task test results on real robot.** We evaluate 10 episodes for each task and report the resulting success rate. We denote "oven" as "open oven", "faucet" as "turn faucet", and "teapot" as "relocate teapot". The number in the parenthesis suggests the kitchen ID and "d" suggests testing with distractors.

Method / Task	oven (1)	faucet (1)	teapot (1)	oven (1,d)	faucet (1,d)	teapot (1,d)	**Average**
PerAct	30	**80**	0	10	50	0	
GNFactor	**40**	**80**	**20**	**30**	**60**	**20**	
Method / Task	oven (2)	faucet (2)	teapot (2)	oven (2,d)	faucet (2,d)	teapot (2,d)	
PerAct	10	50	0	10	30	0	22.5
GNFactor	**40**	**60**	**40**	**20**	**50**	**20**	**40.0**

Instead, the point clouds are sourced from RealSense cameras and are therefore imperfect. Despite the limitations in achieving accurate pixel-level reconstruction results, we focus on learning semantic understanding of the whole scene from distilling Diffusion features, which is more important for policy learning.

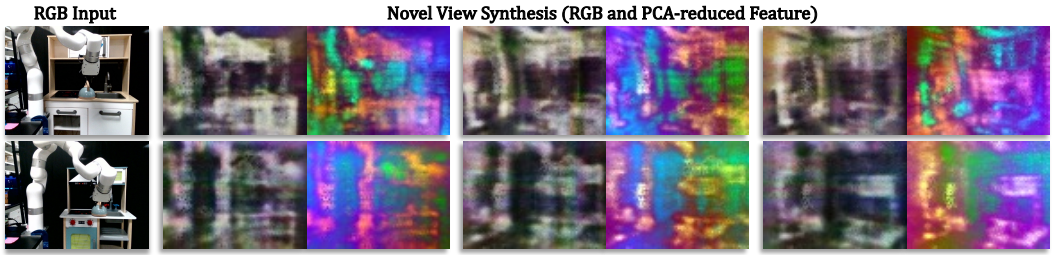


Figure 5: **View synthesis of GNFactor in the real world.** Novel views of the input scene in RGB and the corresponding PCA-reduced feature visualization. The visualization is relatively noisy since: (i) During training, the volumetric representation is optimized with both behavior cloning and rendering losses simultaneously; (ii) In inference, we do not optimize per-step for rendering but just perform one feedforward to obtain the feature.

4.3 Real Robot Experiments

We summarize the results of our real robot experiment in Table 3. From the experiments, GNFactor outperforms the PerAct baseline on almost all tasks. Notably, in the *teapot* task where the agent is required to accurately determine the grasp location and handle the teapot from a correct angle, PerAct fails to accomplish the task and obtains a zero success rate across two kitchens. We observed that it is indeed challenging to learn a delicate policy from only 5 demonstrations. However, by incorporating the representation from the embedding of a vision-language model, GNFactor gains an understanding of objects. As such, GNFactor does not simply overfit to the given demonstrations.

The second kitchen (Figure 1) presents more challenges due to its smaller size compared to the first kitchen. This requires higher accuracy to manipulate the objects effectively. The performance gap between GNFactor and the baseline PerAct becomes more significant in the second kitchen. Importantly, our method does not suffer the same performance drop transitioning from the first kitchen to the second, unlike the baseline.

5 Conclusion and Limitations

In this work, we propose GNFactor, a visual behavior cloning agent for real-world multi-task robotic manipulation. GNFactor utilizes a Generalizable Neural Feature Field (GNF) to learn a 3D volumetric representation, which is also used by the action prediction module. We employ the vision-language feature from the foundation model Stable Diffusion besides the RGB feature to supervise the GNF training and observe that the volumetric representation enhanced by the GNF is helpful for decision-making. GNFactor achieves strong results in both simulation and the real world, across 10 RLBench tasks and 3 real robot tasks, showcasing the potential of GNFactor in real-world scenarios. **We are committed to releasing the code.**

One major limitation of GNFactor is the requirement of multiple views for the GNF training, which can be challenging to scale up in the real world. Currently, we use three fixed cameras for GNFactor, but it would be interesting to explore using a cell phone to randomly collect camera views, where the estimation of the camera poses would be a challenge.

References

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv*, 2022.

[2] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *CoRL*, 2022.

[3] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *CoRL*, 2023.

[4] M. Dalal, A. Mandlekar, C. Garrett, A. Handa, R. Salakhutdinov, and D. Fox. Imitating task and motion planning with visuomotor transformers. *arXiv*, 2023.

[5] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[6] S. Parisi, A. Rajeswaran, S. Purushwalkam, and A. Gupta. The unsurprising effectiveness of pre-trained vision models for control. In *ICML*, 2022.

[7] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3m: A universal visual representation for robot manipulation. *arXiv*, 2022.

[8] I. Radosavovic, T. Xiao, S. James, P. Abbeel, J. Malik, and T. Darrell. Real-world robot learning with masked visual pre-training. In *CoRL*, 2023.

[9] M. Laskin, A. Srinivas, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *ICML*, 2020.

[10] N. Hansen and X. Wang. Generalization in reinforcement learning by soft data augmentation. In *ICRA*, 2021.

[11] Y. Ze, N. Hansen, Y. Chen, M. Jain, and X. Wang. Visual reinforcement learning with self-supervised 3d representations. *RA-L*, 2023.

[12] D. Driess, I. Schubert, P. Florence, Y. Li, and M. Toussaint. Reinforcement learning with neural radiance fields. *NeurIPS*, 2022.

[13] A. Jaegle, F. Gimeno, A. Brock, O. Vinyals, A. Zisserman, and J. Carreira. Perceiver: General perception with iterative attention. In *ICML*, 2021.

[14] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison. Rlbench: The robot learning benchmark & learning environment. *RA-L*, 2020.

[15] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3758–3765. IEEE, 2018.

[16] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *CoRL*, 2022.

[17] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv*, 2018.

[18] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[19] R. Yang, H. Xu, Y. Wu, and X. Wang. Multi-task reinforcement learning with soft modularization. *Advances in Neural Information Processing Systems*, 33:4767–4777, 2020.

[20] S. Song, A. Zeng, J. Lee, and T. Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *IEEE Robotics and Automation Letters*, 5(3): 4978–4985, 2020.

[21] A. Murali, A. Mousavian, C. Eppner, C. Paxton, and D. Fox. 6-dof grasping for target-driven object manipulation in clutter. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6232–6238. IEEE, 2020.

[22] A. Mousavian, C. Eppner, and D. Fox. 6-dof graspnet: Variational grasp generation for object manipulation. In *ICCV*, 2019.

[23] Z. Xu, Z. He, and S. Song. Universal manipulation policy network for articulated objects. *RA-L*, 2022.

[24] Y. Li, S. Agrawal, J.-S. Liu, S. K. Feiner, and S. Song. Scene editing as teleoperation: A case study in 6dof kit assembly. In *IROS*, 2022.

[25] N. Hansen, Z. Yuan, Y. Ze, T. Mu, A. Rajeswaran, H. Su, H. Xu, and X. Wang. On pre-training for visuo-motor control: Revisiting a learning-from-scratch baseline. In *ICML*, 2022.

[26] D. Shim, S. Lee, and H. J. Kim. Snerl: Semantic-aware neural radiance fields for reinforcement learning. *ICML*, 2023.

[27] Y. Chen, S. Liu, and X. Wang. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8628–8638, 2021.

[28] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34: 852–863, 2021.

[29] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019.

[30] 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. *Communications of the ACM*, 65(1):99–106, 2021.

[31] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5379–5389, 2019.

[32] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019.

[33] V. Sitzmann, M. Zollhöfer, and G. Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *Advances in Neural Information Processing Systems*, 32, 2019.

[34] Y. Chen and X. Wang. Transformers as meta-learners for implicit neural representations. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVII*, pages 170–187. Springer, 2022.

[35] W. Jang and L. Agapito. Codenerf: Disentangled neural radiance fields for object categories. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12949–12958, 2021.

[36] K.-E. Lin, Y.-C. Lin, W.-S. Lai, T.-Y. Lin, Y.-C. Shih, and R. Ramamoorthi. Vision transformer for nerf-based view synthesis from a single input image. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 806–815, 2023.

[37] J. Reizenstein, R. Shapovalov, P. Henzler, L. Sbordone, P. Labatut, and D. Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10901–10911, 2021.

[38] K. Rematas, R. Martin-Brualla, and V. Ferrari. Sharf: Shape-conditioned radiance fields from a single view. *arXiv preprint arXiv:2102.08860*, 2021.

[39] A. Trevithick and B. Yang. Grf: Learning a general radiance field for 3d representation and rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15182–15192, 2021.

[40] Q. Wang, Z. Wang, K. Genova, P. P. Srinivasan, H. Zhou, J. T. Barron, R. Martin-Brualla, N. Snavely, and T. Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2021.

[41] A. Yu, V. Ye, M. Tancik, and A. Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021.

[42] V. Tschernezki, I. Laina, D. Larlus, and A. Vedaldi. Neural feature fusion fields: 3d distillation of self-supervised 2d image representations. *arXiv*, 2022.

[43] S. Kobayashi, E. Matsumoto, and V. Sitzmann. Decomposing nerf for editing via feature field distillation. *NeurIPS*, 2022.

[44] J. Ye, N. Wang, and X. Wang. Featurenerf: Learning generalizable nerfs by distilling foundation models. *arXiv*, 2023.

[45] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik. Lerf: Language embedded radiance fields. *arXiv preprint arXiv:2303.09553*, 2023.

[46] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

[47] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[48] S. James, K. Wada, T. Laidlow, and A. J. Davison. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation. In *CVPR*, 2022.

[49] H. Lin, S. Peng, Z. Xu, Y. Yan, Q. Shuai, H. Bao, and X. Zhou. Efficient neural radiance fields for interactive free-viewpoint video. In *SIGGRAPH Asia*, 2022.

[50] Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, and C.-J. Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv*, 2019.