Novel Demonstration Generation with Gaussian Splatting Enables Robust One-Shot Manipulation

Sizhe Yang *,1,2 Wenye Yu*,1,3 Jia Zeng[1] Jun Lv[3] Kerui Ren[1,3]
Cewu Lu[3] Dahua Lin[1,2] Jiangmiao Pang[1,†]
[1] Shanghai AI Laboratory [2] The Chinese University of Hong Kong
[3] Shanghai Jiao Tong University
* Equal contribution † Corresponding authors
Project page: https://yangsizhe.github.io/splatforge/

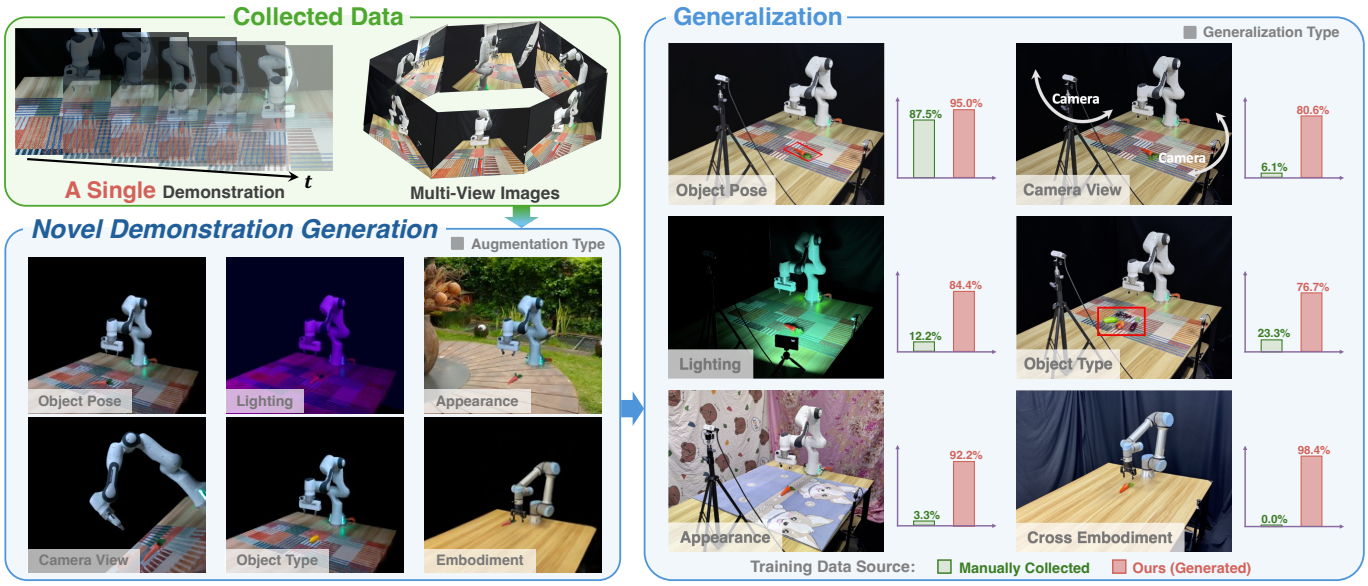


Fig. 1: **Starting from a single expert demonstration and multi-view images, our method generates diverse and visually realistic training data for policy learning, enabling robust performance across six different types of generalization in the real world.**

***Abstract*—Visuomotor policies learned through imitation learning often struggle to generalize to novel visual domains due to the limited diversity of expert demonstrations, since collecting extensive real-world data is exhaustive. To address this challenge, we propose SplatForge, a novel demonstration generation approach leveraging 3D Gaussian Splatting (3DGS), a technique offering an explicit and interpretable representation of 3D scenes. Our method reconstructs manipulation scenes with high fidelity and enables autonomous scene editing, giving rise to novel scene configurations. Stemming from a single expert demonstration, diversified data is generated across various visual domains, including different object poses, object types, camera views, scene appearance, lighting conditions, and robot embodiments. Comprehensive real-world experiments indicate that our demonstration generation pipeline significantly enhances the generalization of visuomotor policies when confronting multiple disturbances. Specifically, while policies trained on real-world demonstrations achieve an average success rate of less than 10%, our method lifts this number to 86.5% across a variety of challenging task settings and scenarios.**

I. Introduction

Imitation learning with visuomotor policy has become an emerging paradigm in the field of robot manipulation. However, the learned policy is vulnerable to deployment settings that deviate far from expert demonstrations due to insufficient coverage of possible visual domains. While scaling up and diversifying data in real-world setting is an effective way to enhance robustness [12], human-collected demonstrations entail prohibitive time and labor consumption. Therefore, a large quantity of work has been dedicated to generating visually diverse expert data, without interacting with real-world environment [8, 10, 65, 9, 35, 49].

Simulated environment offers a low-cost platform for data synthesis. However, the Sim-to-Real visual gap poses significant challenges, hindering the policy's performance in real-world scenarios. Additionally, replicating real-world manipulation scenes within simulation remains complex and arduous. Another line of work sheds light on augmenting image observations for better visual generalization. By editing different

semantic parts of the image inputs, these approaches generate novel scene configurations for the given tasks, in terms of background appearances [9, 65, 10], embodiment types [8], and camera views [49]. While these image augmentation methods are convenient, their limited consideration of spatial information results in less realistic and spatially inaccurate data generation.

Recently, 3D Gaussian Splatting (3DGS) [25] has become a burgeoning approach to superior reconstruction and rendering. Thanks to its explicit representation of the scene, 3DGS enables interpretable editing of the entire scene, which paves the way for generating novel manipulation configurations. Furthermore, as a 3D representation of the scene, 3DGS retains spatial information from real world and allows for consistent rendering from multiple perspectives, which makes it the real-world counterpart of a simulator's graphics engine for generating novel demonstrations.

Based on that, we propose a novel and efficacious approach to demonstration generation with Gaussian Splatting. Empowered by 3DGS, we achieve a high-fidelity reconstruction of the manipulation scene with multi-view images, which capture the scene. In order to align the reconstructed scene with real-world counterparts, we devise a novel frame alignment pipeline leveraging the differentiable rendering of Gaussian Splatting. 3D Gaussians of different scene components are segmented through off-the-shelf segmentation models and the robot United Robotics Description Format (URDF), which are then explicitly edited to form novel scene configurations. Remarkably, as illustrated in Fig. 1, with only one manually collected expert trajectory, we are able to churn out novel demonstrations in a wide variety of visual domains, including different object poses, object types, lighting conditions, scene appearance, camera views, and embodiment types, in a fully autonomous fashion. Crucially, compared to previous Sim-to-Real and image augmentation approaches, our method achieves a minimal visual gap between the generated data and deployment observations, along with precise handling of spatial information, enabling the trained policy to be directly deployed in the real world. Moreover, thorough real-world experiments indicate that our demonstration generation method significantly facilitates the robustness of visuomotor policies whilst encountering multiple disturbances. Policies derived through our method demonstrate exceptional performance compared to those trained exclusively on real-world demonstrations across Pick-and-Place tasks, articulated object manipulation, and long-horizon skills. Specifically, when subjected to strong perturbations, our policy significantly enhances the average success rate from below 10.0% to 86.5%.

In a nutshell, our contributions are three-fold:

1) We devise **SplatForge**, an autonomous demonstration generation system that merely requires a single expert demonstration and generates visually realistic and diverse data for robust policy learning.
2) We employ 3D Gaussian Splatting to achieve high-fidelity manipulation scene reconstruction and fine-grained 3D registration between real-world scenarios and their reconstructed digital counterparts. The reconstructed scenes are then autonomously edited to generate novel demonstrations.
3) Extensive real-world experiments justify that our approach significantly improves the robustness of visuomotor policies in unseen scenarios, with diversified object poses, object types, lighting conditions, scene appearance, camera views, and embodiment types.

II. Related work

A. Generalizable Policy in Robot Manipulation

Recent advancements in manipulation have significantly enhanced generalization. Some studies design the policy architecture to endow it with equivariant properties, which is helpful to generalizing to different object poses [58, 59, 43, 13]. One-shot imitation learning approaches like [53, 48, 6, 52, 67] enable the policy to handle various object poses given only one demonstration. Furthermore, some other work focuses on generalizing the policy to different camera views [66, 46, 61], scene appearance [30, 50], and embodiments [12]. Some studies exploit the power of Large Language Models (LLMs) and Vision Language Models (VLMs) to endow robots with generalization abilities [23, 7, 39, 14]. Instead of adopting generalizable policy architecture, auxiliary learning objectives and powerful foundation models, our work is concentrated on generating high-quality, diverse, and realistic data to instill generalization abilities to the learned policy.

B. Data Augmentation for Policy Learning

Given limited training data, data augmentation emerges as a way to improve the robustness of the policy. Previous work adopts image augmentation techniques to improve the resistance of visuomotor policies to observation noises [29, 28, 36, 37, 15, 19, 20]. However, these methods are mainly evaluated in simulated environments. To deploy learned policies in real-world setting, some previous work focuses on augmenting the appearance of the scene by incorporating image-inpainting models [65, 10, 9, 35]. Moreover, Tian et al. [49] generate augmented task demonstrations from different camera views and aim to learn a view-invariant policy. Ameperosa et al. [3]. Chen et al. [8] further devise a cross-embodiment pipeline by inpainting different robots to image observations. Nonetheless, these studies mainly augment task demonstrations on 2D images, which lack spatial information. Hence, only limited augmentation can be achieved, and the augmented demonstrations might be unrealistic compared to those generated directly from 3D representations. Our work reconstructs the scene with 3D Gaussian Splatting and edits the 3D representation for data augmentation, enabling our policy to achieve comprehensive generalization across object poses, object types, camera views, lighting conditions, scene appearance, and various embodiments..

C. Gaussian Splatting in Robotics

3D Gaussian Splatting (3DGS) [25] serves as an explicit radiance field representation for real-time rendering of 3D

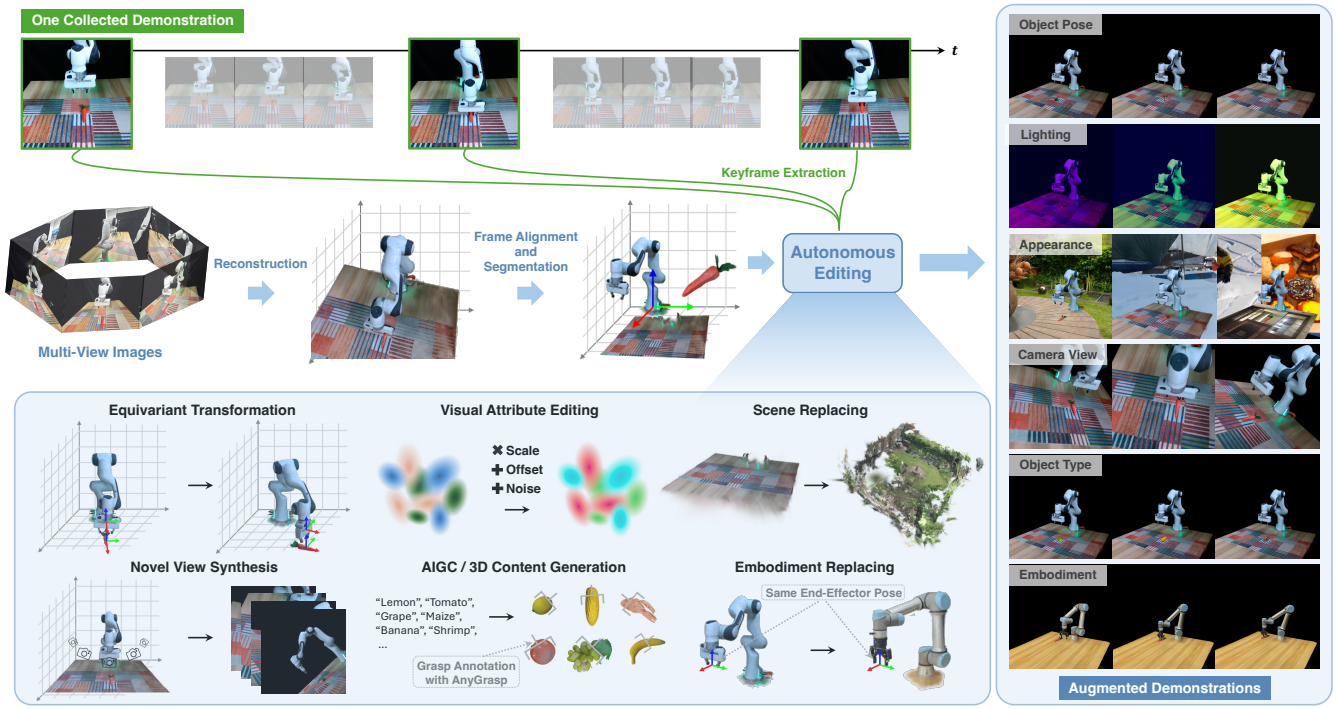


Fig. 2: **Method overview.** We start from a single manually collected demonstration and multi-view images that capture the whole scene. The former provides task-related keyframes, while the latter helps scene reconstruction. After aligning the reconstructed frame with the real-world frame and segmenting different scene components, we carry out autonomous editing of the scene in pursuit of six different types of generalization.

scenes. Previous work leverages 3DGS to select proper grasp poses [24, 68]. Furthermore, Lu et al. [34] exploit 3DGS to construct dynamics of the scene for multi-task robot manipulation. In order to predict the consequence of robots' interactions with the environment, Shorinwa et al. [47] leverage 3D semantic masking and infilling to visualize the motions of the objects that result from the interactions. Another line of work adopts the Real-to-Sim-to-Real pipeline, and utilizes 3DGS to reconstruct the real-world scene [31, 40, 55, 51]. However, importing reconstructed real-world objects to simulation is a strenuous process, and physical interactions tend to suffer from large sim-to-real gaps due to the flawed geometric reconstruction and lack of physical information in 3D reconstruction. Some recent work on 3DGS is centered around editing and relighting of the scene [63, 32, 17]. Our method enables autonomous editing of the reconstructed scene to generate diverse demonstrations with various configurations.

III. Preliminaries

A. 3D Gaussian Splatting

3D Gaussian Splatting (3DGS) [25] utilizes multi-view images for high-fidelity scene reconstruction. The scene is represented by a set of Gaussians $\{g_i\}_{i=1}^{N}$, where each Gaussian g_i consists of a position vector $\mu_i \in \mathbb{R}^3$, a rotation matrix $R_i \in \mathbb{R}^{3\times3}$, a scaling matrix $S_i = diag(s)(s \in \mathbb{R}^3)$, an opacity factor $\alpha_i \in \mathbb{R}$, and spherical harmonic coefficients c_i that encapsulate the view-dependent color appearance of the Gaussian. Given the scaling matrix and rotation matrix, the covariance matrix Σ_i is calculated as follows:

$$\Sigma_i = R_i S_i S_i^\top R_i^\top.$$

To derive the color C of a particular pixel during rendering procedure, 3DGS exploits a typical neural point-based approach, similar to Kopanas et al. [27], where the final color value is calculated as follows:

$$C = \sum_{i=1}^{N} c_i o_i \prod_{j=1}^{j=i-1} (1 - o_j),$$

$$o_i = \alpha_i \cdot \exp\left(\frac{1}{2}\delta_i^\intercal \Sigma_{i,2D}^{-1} \delta_i\right),$$

where N is the number of Gaussians that overlap with the pixel. Besides, α_i denotes the opacity of the i-th Gaussian. $\delta_i \in \mathbb{R}^2$ denotes the offset between the current pixel and the center of the i-th Gaussian projected to 2D image. $\Sigma_{i,2D} \in \mathbb{R}^{2\times2}$ stands for the covariance matrix of the i-th Gaussian projected to 2D image.

IV. Method

To generate high-fidelity and diverse data from only one expert trajectory, we present a novel demonstration generation approach based on 3DGS. An overview of our method is shown in Fig. 2. In the first place, we capture multiple images of the scene and objects. Subsequently, the Gaussian

models of them are reconstructed and preprocessed. With all the Gaussian models ready, we thereupon generate novel demonstrations and perform data augmentation in terms of object poses, object types, camera views, scene appearance, lighting conditions, and embodiments. A visuomotor policy is then trained on the augmented demonstrations and directly deployed on real robots.

In this section, we elaborate on our method. First, we present problem formulation in Sec. IV-A. Then, we detail the process of reconstruction and preprocessing in Sec. IV-B, including object and scene reconstruction, frame alignment with differentiable rendering, and novel pose generation for the robot and objects. Lying at the core of our method, the demonstration augmentation pipeline is expanded in Sec. IV-C. Finally, we describe our policy training and deployment protocol in Sec. IV-D.

A. Problem Formulation

As a prerequisite for demonstration generation, multiple images of the scene are obtained for 3D reconstruction. And a single expert demonstration $\mathcal{D}_{\text{expert}} = \{(I_t, q_t, a_t)_{t=1}^{T}\}$ for each task is collected, where T indicates the total steps of the trajectory, I_t represents the observation in the form of RGB images, q_t denotes joint position, and a_t refers to robot action that includes arm action (relative end-effector pose) and binary gripper action (open or close). Given the reconstructed scene and the single demonstration, the objective is to generate novel demonstrations $\mathcal{D}_{\text{aug}} = \{(I_t, q_t, a_t)_{t=1}^{T_k}\}_{k=1}^{N}$, which can be used to train a policy that robustly performs tasks when facing various challenges, such as changes in objects, background, lighting, camera views, and embodiments.

B. Reconstruction and Preprocessing

In pursuit of a high-fidelity reconstruction of the scene, we first capture a set of RGB images whose corresponding viewpoints should be as various as possible. During this process, the scene remains static and the robot is fixed at its default joint configuration, which we refer to as q_{default}. With the images ready, we utilize COLMAP [45, 44] to obtain a sparse scene reconstruction and an estimation of the camera pose corresponding to each image. To further enhance the reconstruction precision, we gain an depth estimation for each image with Depth Anything [60]. The images, camera poses, and depth prior serve as inputs to 3DGS [25], which returns 3D Gaussians representing the entire scene $\mathcal{G}_{\text{scene}}$, which contains 3D Gaussians corresponding to the robot, dubbed $\mathcal{G}_{\text{robot}}$.

However, the reconstructed 3D Gaussians of the robot are represented in an arbitrary frame $\mathcal{F}_{\text{scene}}$, and hence we need to align it with the real-world coordinate frame $\mathcal{F}_{\text{real}}$ to facilitate automated realistic editing.

The robot URDF file gives us access to the robot base frame $\mathcal{F}_{\text{URDF}}$. The real-world robot frame $\mathcal{F}_{\text{robot}}$, $\mathcal{F}_{\text{URDF}}$, and $\mathcal{F}_{\text{real}}$ are all aligned with each other, so that the real-world coordinate frame is equivalent to the robot base frame. Therefore, the actual problem turns into the frame alignment from $\mathcal{F}_{\text{scene}}$ to $\mathcal{F}_{\text{URDF}}$, and we denote the transformation matrix as $\mathcal{T}_{\text{URDF, scene}}$.

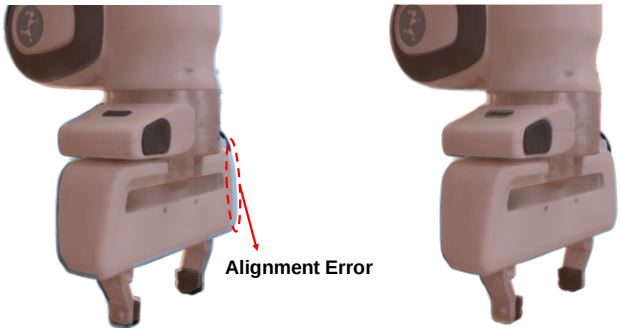


Fig. 3: **Comparison of frame alignment results between ICP and fine-grained optimization with differentiable rendering.** The semi-transparent orange overlay represents the ground truth rendered with URDF from the same camera view. The **left** shows the results of ICP, which have larger errors, while the **right** shows the results after further fine-grained optimization using differentiable rendering.

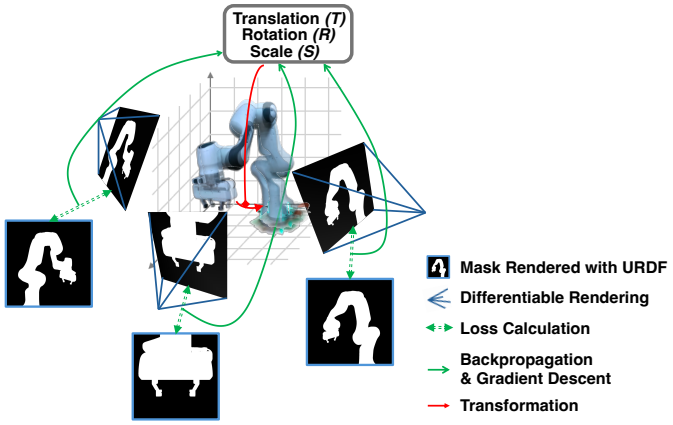


Fig. 4: **Illustration of frame alignment with differentiable rendering.** The loss is calculated between the mask rendered using Gaussian Splatting and the mask rendered with URDF. Subsequently, backpropagation and gradient descent are used to optimize the translation, rotation, and scale, which are then applied to the 3D Gaussians.

While point cloud registration approachs, such as Iterative Closest Point (ICP) [5], serve as a common solution to it, we find that there is still major misalignment between the two frames aligned with point cloud registration alone, as illustrated in Fig. 3. The reason lies in the fact that point cloud registration is based on point coordinates, whereas 3D Gaussians have a scale attribute, which causes a mismatch between their point coordinates and appearance. Therefore, we exploit the differentiable rendering of 3DGS to do further fine-grained alignment, as depicted in Fig. 4.

Suppose $\hat{\mathcal{T}}^{0}_{\text{URDF, scene}}$ is the initial transformation matrix obtained through ICP. We first apply $\hat{\mathcal{T}}^{0}_{\text{URDF, scene}}$ to $\mathcal{G}_{\text{robot}}$, leading to a partially aligned robot Gaussian $\hat{\mathcal{G}}_{\text{robot}}$. The aim of further alignment is to derive another transformation matrix $\hat{\mathcal{T}}_{\text{rel}}$, such that applying $\hat{\mathcal{T}}_{\text{rel}}$ to $\hat{\mathcal{G}}_{\text{robot}}$ gives a better alignment

to the pose of the robot defined in URDF. For this sake, we select N canonical camera views to capture the segmentation masks $\{\mathcal{I}_i^{\text{URDF}}\}_{i=1}^N$ and $\{\mathcal{I}_i^{\text{Gaussian}}\}_{i=1}^N$ (the pixel value is 1 if it belongs to the robot; otherwise, it is 0) with the robot URDF and $\hat{\mathcal{G}}_{\text{robot}}$ respectively. The pixel-wise differences between the images from the same canonical views are averaged to form the objective function of alignment:

$$\mathcal{L}_{\text{align}} = \frac{1}{N}\sum_{i=1}^{N}(\mathcal{I}_i^{\text{URDF}} - \mathcal{I}_i^{\text{Gaussian}})^2.$$

Due to the differentiability of Gaussian Splatting, we can rewrite the objective function as $\mathcal{L}_{\text{align}}(\hat{\mathcal{T}}_{\text{rel}})$ and optimize $\hat{\mathcal{T}}_{\text{rel}}$ through gradient descent. The optimized $\hat{\mathcal{T}}_{\text{rel}}$ is composed with $\hat{\mathcal{T}}^0_{\text{URDF, scene}}$, the result of which is applied to $\mathcal{G}_{\text{scene}}$ to form the scene reconstruction in $\mathcal{F}_{\text{real}}$. We refer to the aligned 3D Gaussians as $\mathcal{G}^*_{\text{scene}}$.

In order to decompose the scene into different parts, we first leverage Grounded-SAM [41] to perform task-related object segmentation. Then, the masked images are used to reconstruct 3D Gaussians for the objects. The 3D Gaussians corresponding to each link of the robot are segmented using the point cloud of each link in $\mathcal{F}_{\text{URDF}}$, which can be obtained with the robot's URDF and the renderer. Specifically, if the position of a 3D Gaussian is within a threshold distance from the point cloud of a link, the 3D Gaussian is assigned to that link. If a 3D Gaussian does not belong to any object or any link of the robot, it is classified as background. We suppose that the robot has l links and there are totally k objects in the scene. The reconstructed robot links, objects, and background are denoted as $\mathcal{G}^*_{\text{robot}} = \{\mathcal{G}^*_{\text{robot},i}\}_{i=1}^l$, $\mathcal{G}^*_{\text{obj}} = \{\mathcal{G}^*_{\text{obj},j}\}_{j=1}^k$, and $\mathcal{G}^*_{\text{bg}}$ respectively.

Similar to our frame alignment strategy, we utilize differentiable rendering to estimate the deployed camera poses in order to narrow the gap between the generated data and the deployment environment. The camera extrinsics are optimized through gradient descent, with the optimization objective:

$$\mathcal{L}_{\text{camera}} = SSIM(\mathcal{I}_{\text{Expert}}, \mathcal{I}_{\text{Gaussian}})^2,$$

where $\mathcal{I}_{\text{Expert}}$ denotes the image obtained from the collected expert demonstration, $\mathcal{I}_{\text{Gaussian}}$ represents the rendered image with reconstructed 3D Gaussians, and SSIM refers to Structural Similarity, which measures the perceptual similarity between two images.

Nonetheless, before moving on to novel demonstration generation, we need to figure out how to generate 3D Gaussians for the robot under novel joint configurations. To achieve that, we leverage the link-wise Gaussians $\{\mathcal{G}^*_{\text{robot},i}\}_{i=1}^l$ and the default joint configuration q_{default}. For each link $1 \leqslant i \leqslant l$, we access its relative pose to robot base frame under arbitrary joint configuration q through forward kinematics, denoted as $\mathcal{T}^i_{\text{fk}}(q)$. Hence, by transforming each link i with $\mathcal{T}^i_{\text{fk}}(q)\mathcal{T}^i_{\text{fk}}(q_{\text{default}})^{-1}$, we derive the corresponding 3D Gaussians under configuration q. The entire 3D Gaussians are thereby derived by composing Gaussians of all l links. As for the manipulated objects, we apply transformations in a similar manner. The way 3D Gaussians are transformed is detailed in Appendix A.

C. Demonstration Augmentation

Utilizing 3D Gaussians in $\mathcal{F}_{\text{real}}$, we implement our demonstration augmentation process, which systematically enhances the expert demonstration $\mathcal{D}_{\text{expert}}$ across six aspects: object poses, object types, camera views, embodiment types, scene appearance, and lighting conditions.

1) Object pose

To perform object pose augmentation, we first extract keyframes from the expert demonstration using a heuristic approach. Whenever the gripper action toggles or joint velocities approach zero, we consider the current time step as a keyframe and record the end-effector pose with respect to robot base frame. After that, we apply rigid transformations to the target objects that are involved in the expert demonstration. The end-effector poses at keyframes are transformed equivariantly according to the target object. Eventually, we generate trajectories between consecutive keyframe poses with motion planning, the combination of which makes a complete augmented demonstration with novel object poses.

2) Object type

The object types can also be augmented with 3D Content Generation. We first prompt GPT-4 [2] to generate approximately 50 names of objects that can be grasped. Then, we use these object names as prompts to generate corresponding 3D Gaussians with a 3D content generation model [56]. We utilize an off-the-shelf grasping algorithm [16] to generate grasp poses with respect to the object frame. As we generate different object poses for augmentation, we obtain the corresponding end-effector poses by composing object pose and the relative grasp pose to the object, which turn into the keyframe poses in new demonstrations. The entire augmented trajectory is generated in the same manner as IV-C1.

3) Camera view

One merit of 3DGS lies in its ability to perform novel view synthesis. Thereby, we are able to choose different camera poses from $\mathcal{D}_{\text{expert}}$ and obtain novel-view demonstrations. Whereas we can render novel-view observations from arbitrary camera pose, we need to ensure that the augmented camera view does not deviate so much from the expert that it loses sight of the manipulation scene. Hence, we first designate a target point $O_c = (x_c, y_c, z_c)$ in $\mathcal{F}_{\text{real}}$, towards which the camera should face during the entire episode. We then define a coordinate frame $\mathcal{F}_c$, whose origin is O_c and orientation is the same as $\mathcal{F}_{\text{real}}$. The position of camera is represented by spherical coordinates (r, θ, φ) in $\mathcal{F}_c$. Thus, by limiting the target point within the manipulation scene and randomizing the spherical coordinates, we are able to generate camera poses that produce meaningful observations yet possess diversity. The hyperparameters of randomization for the target point and the spherical coordinates are detailed in Appendix B.

4) Embodiment type

To generalize the expert demonstration to different types of robots, we replace $\mathcal{G}^*_{\text{robot}}$ with the 3D Gaussians of another embodiment, dubbed $\mathcal{G}^{\text{new}}_{\text{robot}}$, which is attained from

the corresponding URDF file or real-world reconstruction. The keyframe end-effector poses are reused because they are embodiment-agnostic action representations. Hence, through motion planning, we can easily derive the end-effector poses and joint positions of the new embodiment for all time steps in augmented demonstrations. The 3D Gaussians of the new embodiment under novel joint configurations is obtained from $\mathcal{G}_{\text{robot}}^{\text{new}}$ as mentioned in Sec. IV-B. The policy trained on these augmented demonstrations is directly deployed on novel embodiments.

5) Scene appearance

Inconsistency between scene appearance accounts for a large visual gap between training and deployment environments. To resolve this issue, we propose to exploit reconstructed diverse 3D scenes and also large-scale image datasets to augment the scene appearance. We adopt COCO [33] as the image dataset, and attach images to the table top and background 3D Gaussian planes that surround the entire manipulation scene. Moreover, we gather datasets for 3D reconstruction [22, 64, 26, 4], and derive corresponding 3D Gaussians by 3DGS training. The resulting 3D Gaussian scenes substitute for $\mathcal{G}_{\text{bg}}^{*}$, forming novel scene appearance for data augmentation. The edge of utilizing reconstructed 3D scenes is their their consistent and diverse geometry across multiple camera views, which helps produce more realistic demonstrations. Nevertheless, due to the expense of 3DGS training on large-scale reconstruction datasets, we complement them with 2D images for greater appearance diversity.

6) Lighting condition

Discrepancy in lighting conditions is another barrier to deploying trained policy in unseen scenarios. To compensate for that, we augment the diffuse color of each Gaussian in the reconstructed scene through random scaling, offset, and noise. Concretely, for a Gaussian with original diffuse color (r, g, b), the augmented diffuse color values can be expressed as $(s_r r + o_r + \Delta_r, s_g g + o_g + \Delta_g, s_b b + o_b + \Delta_b)$, where (s_r, s_g, s_b) stand for scaling factors, (o_r, o_g, o_b) stand for offsets, and $(\Delta_r, \Delta_g, \Delta_b)$ stand for random Gaussian noise. The scaling factors and offsets simulate changes in color contrast and scene brightness. Thus, they are shared among all the Gaussians in the scene. On the other hand, the random Gaussian noise is sampled independently for each Gaussian to simulate noise in images captured by cameras. The details of scaling factors, offsets, and Gaussian noise are elaborated in Appendix B.

An illustration of augmented demonstrations with six types of generalizations can be found in Appendix B.

D. Policy Training

We employ a modern, widely adopted transformer-based architecture [18, 50, 38, 54] to serve as the policy network, which is detailed in Appendix C. We process RGB images with ResNet-18 [21], and encode joint state using a multi-layer perceptron (MLP). The latent of images and robot state is fed into a transformer encoder. Finally, an action decoder utilizes an MLP to convert the action latent into the action vector a_t. The policy is trained with Behavioural Cloning (BC) in an end-to-end manner, aiming to maximize the likelihood of expert actions in demonstrations. We denote $o_k \triangleq (I_k, q_k)$ as the observation at the k-th frame of demonstrations $\mathcal{D}$, and π as our policy. The loss function can then be expressed as Eq. 1:

$$\mathcal{L}^{\text{BC}} = \mathbb{E}_{(o_k, a_k)\sim\mathcal{D}} \|a_k - \pi(o_k)\|^2, \tag{1}$$

Specifically, I_k consists of two images from different eye-on-base cameras. We adopt relative end-effector pose as the action representation, which depicts the relative transformation between two consecutive end-effector poses under robot base frame. Further details of the training process can be found in Appendix D.

V. Experiments

We conduct comprehensive experiments in the real world to verify the effectiveness of our demonstration generation pipeline. Specifically, we aim to answer: given a single expert demonstration and multi-view images of the scene,

1) How efficient is data generation compared to manually collecting data?

2) How does the policy trained on generated demonstrations perform across various tasks compared to that trained on manually collected data?

3) How does the policy perform as the generated data scale up?

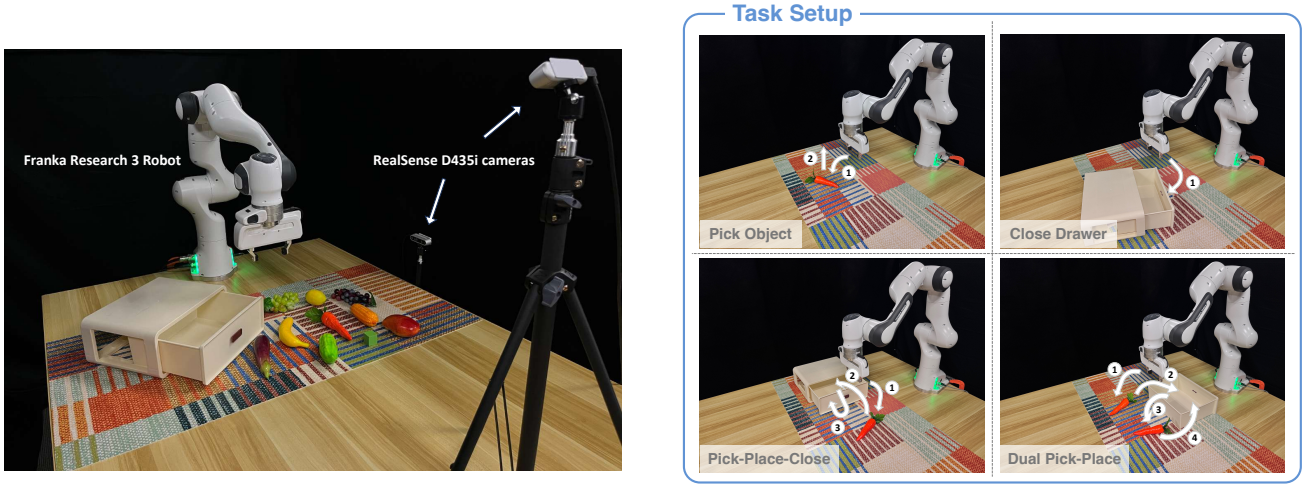
4) Can generated demonstrations enhance the robustness of the policy when facing various deployment settings, such as changes in object types, camera views, scene appearance, lighting conditions, and embodiment types?

A. Experiment Setup

The real-world experiment setup is presented in Fig. 5(a). Concretely, we collect the expert demonstration on Franka Research 3 (FR3) Robot. Two Intel Realsense D435i eye-on-base cameras are mounted on the table top, capturing RGB image observations for the policy. We employ a 3D Spacemouse to collect teleoperated demonstrations at a frequency of 10 Hz. Policy inference is carried out on an NVIDIA RTX4090 GPU, with a latency of 0.1s imposed.

In order to manifest the generalization ability of our pipeline to different task settings, we select four tasks for evaluation: *Pick Object*, *Close Drawer*, *Pick-Place-Close*, and *Dual Pick-Place*.

In *Pick Object* task, the policy picks up a target object which is placed at different poses within a 30cm×40cm workspace. In *Close Drawer* task, the policy closes a drawer whose position is constrained to a 15cm×40cm workspace, while its rotation about the z-axis is restricted to $[-\frac{\pi}{8}, \frac{\pi}{8}]$. In *Pick-Place-Close* task, the policy is expected to grasp an object, place it in the drawer, and then close the drawer. The drawer is placed in a 5cm×5cm workspace, with a fixed orientation. The target object is located in a 10cm×10cm workspace, whose rotation falls into range $[-\frac{\pi}{8}, \frac{\pi}{8}]$. In *Dual Pick-Place* task, the policy attempts to pick two target objects in a row



(a) Real-world experiment setup

(b) Task illustration

Fig. 5: **Real-world experiment setup and task illustration. Left:** We employ a Franka Research 3 Robot and two eye-on-base RealSense D435i cameras. **Right:** We design four real-world manipulation tasks: *Pick Object*, *Close Drawer*, *Pick-Place-Close*, and *Dual Pick-Place*, whose details are elaborated in Sec. V-A.

and place them in a fixed drawer. Both of the objects are located in 10cm$\times$10cm workspaces, with yaw angles between $-\frac{\pi}{8}$ and $\frac{\pi}{8}$. Task setups are illustrated in Fig. 5(b). These four tasks call for the ability to execute basic Pick-and-Place skills, manipulate articulated objects, and perform long-horizon tasks, and thus make up an all-round evaluation in terms of different task settings.

We conduct extensive real-world experiments to prove the effectiveness of our data generation pipeline in terms of different types of generalizations. Notably, the evaluation of object pose generalization ability is incorporated into all experiments, including those focused on the other five types of generalization (object types, camera views, embodiment types, lighting conditions, and scene appearance). This is because object pose generalization is a fundamental requirement for task completion ability. For the other five types of generalization, the details are provided in Sec. V-D.

Success rate (SR) is chosen as the evaluation metric in all of our experiments. Each policy is evaluated with 30 trials for a certain task. In every task, each policy is evaluated on the same set of target object poses for sake of fairness.

B. Efficiency of Augmenting Demonstrations

To answer Question 1, we need to justify that our pipeline is economical with both labor and time. The labor-saving property is obvious because we only require one expert demonstration in real-world setting. Thereby, we compare the average time consumption of manually collecting a real-world demonstration to that of generating a novel demonstration through our pipeline. Specifically, we adopt eight processes on an NVIDIA RTX 4090 GPU for paralleled data generation to efficiently utilize computational resources.

The comparison study is conducted on all four tasks, and the result is shown in Table I. Our data generation pipeline that executed on a single GPU is more than 28 times faster than collecting data in the real world, with an average time consumption of 0.66s across all four tasks. With no human interference, our demonstration generation approach is able to churn out visually diverse training data with little time expenditure.

C. Performance of the Policy Trained on Augmented Data

To answer Question 2 and 3, we compare the policies trained on generated demonstrations and manually collected demonstrations in terms of their success rates when facing various object poses. Moreover, we explore the performance of policies as generated data gradually scale up.

The main results of the experiment are illustrated in Fig. 6. While policies trained on real-world demonstrations still have an edge over those trained on the same number of generated ones, our method manifests salient improvement in success rate as the number of generated demonstrations scales up. Concretely, visuomotor policies trained on 800 generated demonstrations achieve comparable performance to those trained on 200 manually collected demonstrations. Moreover, training with 1800 generated demonstrations raises the success rate to an average of 95%, significantly surpassing the success rate achieved with 200 manually collected demonstrations. It is also worth mentioning that the policy achieves a 96.7% success rate on *Dual Pick-Place* task with our generated data, which is nearly 20% higher than the baseline (manually collected). These findings testify the effectiveness of our method in generating novel object poses for better generalization of visuomotor policies, and indicate promising scaling property as generated data scale up.

TABLE I: **Comparison of demonstration collection time (s).** We calculate the average time cost of data collection of a single demonstration over 100 demonstrations. Our method achieves more than 28 times the speed compared to the baseline.

Task Type	Pick Object	Close Drawer	Pick-Place-Close	Dual Pick-Place	Average
Real-world	13.2	10.1	24.7	27.0	18.8
Ours	**0.43**	**0.34**	**0.86**	**1.0**	**0.66**

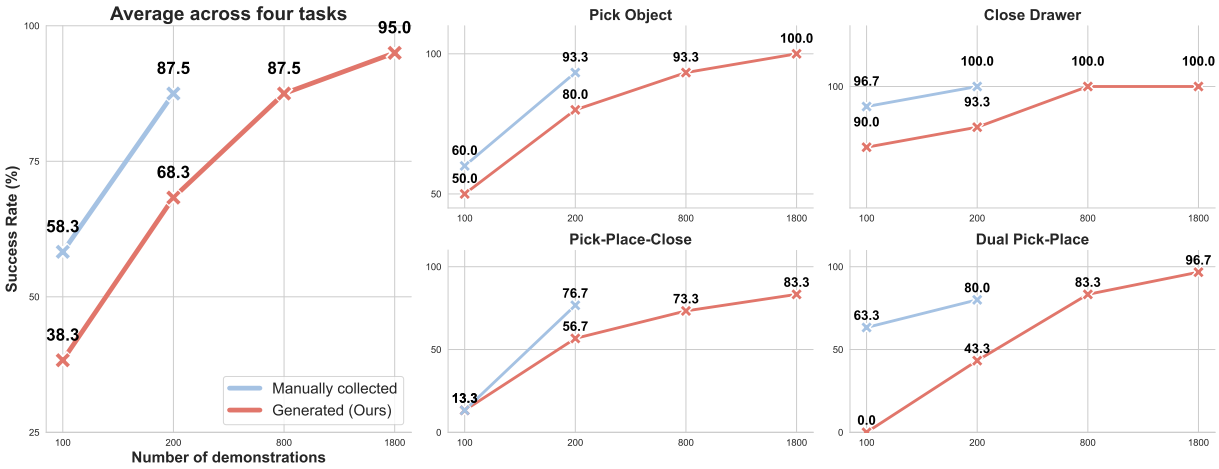


Fig. 6: **Main results. Right:** For each task, we evaluate the success rate of policies trained from manually collected data and those generated by our method over 30 trials, using different number of demonstrations. **Left:** We present the average success rate across four tasks. Our method shows promising scalability as the number of demonstration grows.

D. Robustness when Facing Various Deployment Settings

To answer Question 4, we augment the expert demonstration in five different dimensions: lighting conditions, scene appearance, camera views, object types, and embodiment types. For each of these generalization types except embodiment type, we conduct a comparison with policies that are trained on three types of data:

1) manually collected real-world demonstrations;
2) generated demonstrations with augmentation for object pose;
3) generated demonstrations with augmentation for both object pose and the specific type.

We adopt a different experiment setting for the generalization of embodiment type, which will be expanded in Sec. V-D5. A brief illustration of real-world experiments for different generalization types is presented in Fig. 7.

1) Lighting condition

To demonstrate the effectiveness of lighting augmentation in our approach, we adopt five different scenarios for policy deployment, which are illustrated in Appendix E. We compare the performance of four policies that are trained respectively on:

1) 200 real-world demonstrations (**Collected 200**);
2) 200 generated demonstrations with only object pose augmentation (**Ours Pose-Only 200**);
3) 1800 generated demonstrations with only object pose augmentation (**Ours Pose-Only 1800**);
4) 3200 generated demonstrations with both lighting condition and object pose augmentation (**Ours 3200**).

As shown in Fig. 8, policies trained on augmented lighting condition achieve an average of over 80% success rate across *Pick Object*, *Close Drawer*, and *Pick-Place-Close* tasks, with an overall improvement over real-world baseline by 70%. Furthermore, our policy shows a significant edge over those trained solely on generated demonstrations with augmented object poses, justifying the validity of lighting condition augmentation pipeline.

2) Scene appearance

Similar to the experiment on lighting condition, we select five different scenarios for the ablation studies on scene appearance augmentation, which is illustrated in Appendix E. The four policies for ablation studies are trained in the same manner as Sec. V-D1, and the results are shown in Fig. 8. The policy trained on appearance augmentation demonstrates superior performance to all the baseline models, with an increase in success rate by over 70% in all three tasks. In particular, our policy achieves 100% success rate on the *Pick Object* task, showcasing strong robustness against various background appearance.

3) Camera view

We employ two different settings for camera view generalization: *novel view* and *moving view*. In *novel view* experiment, we select 30 poses for each camera, which are all different from each other and the training perspective. On the other hand, both of the cameras are kept moving in *moving view*

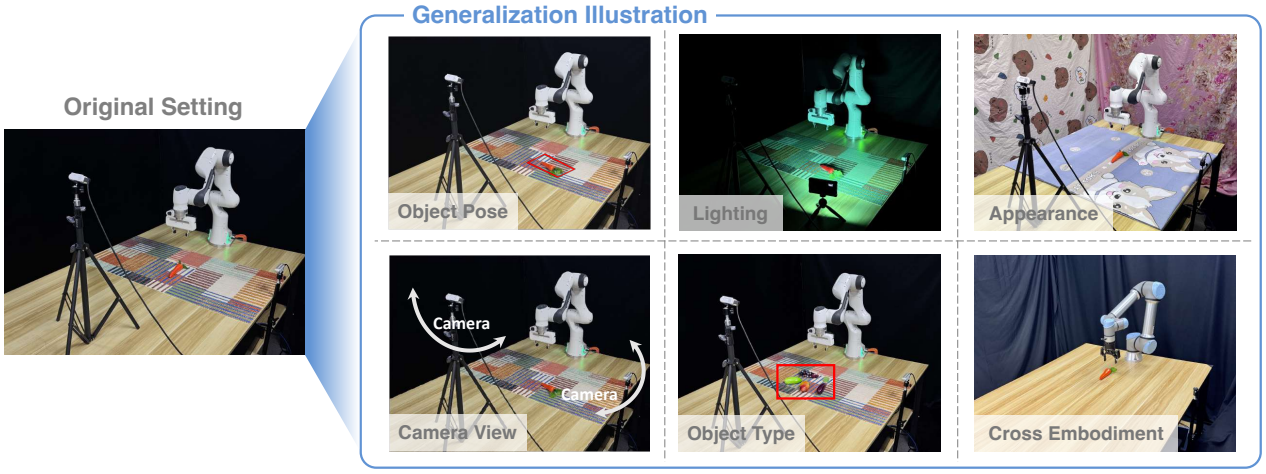


Fig. 7: **Illustration of real-world experiments for different generalization types.** The data is collected in the original setting. When deploying the trained policy, we modify object poses, lighting conditions, scene appearance, camera views, object types, and embodiments to evaluate the robustness in different scenarios.

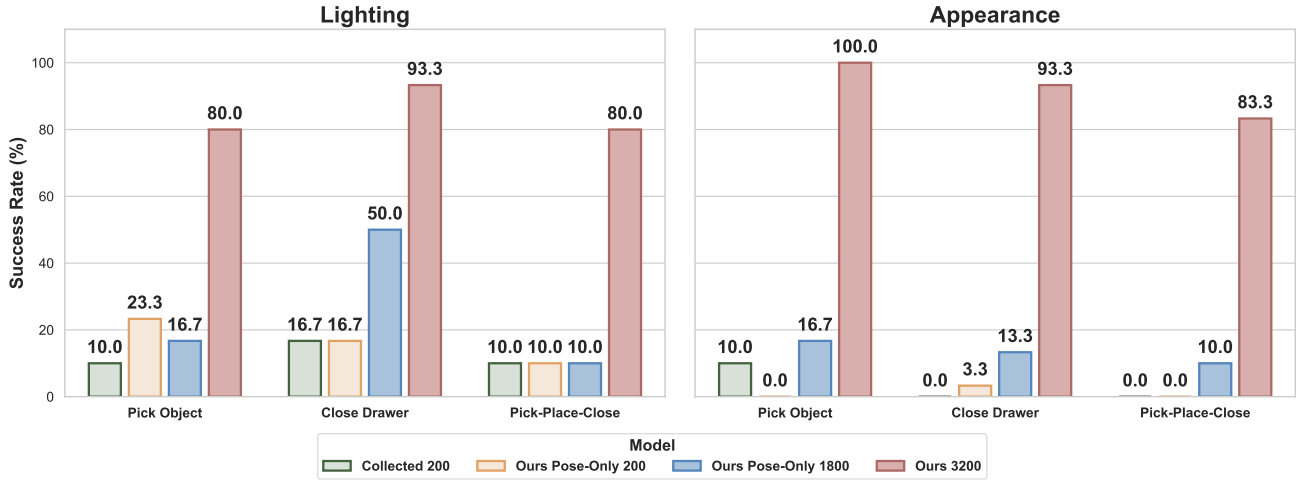


Fig. 8: **Performance when changing lighting conditions and appearance.** We report the success rate of different policies under various lighting conditions and appearance. The policies trained with generated demonstrations with corresponding augmentations manifest remarkable advance compared to baseline policies.

experiment. Similar to Sec. V-D1 and Sec. V-D2, we compare the performance of four policies that are trained respectively on 200 real-world demonstrations (**Collected 200**), 200 generated demonstrations (**Ours Pose-Only 200**), 1800 generated demonstrations (**Ours Pose-Only 1800**), and 3200 generated demonstrations with camera view augmentation (**Ours 3200**).

We present the results in Table II. Our policy is able to perform *Pick Object* task and *Pick-Place-Close* task with success rates of over 80% and 50% respectively, while the baseline policies can barely accomplish the task. Notably, our policy achieves nearly 100% success rate on *Close Drawer* task, manifesting strong robustness against novel camera views and moving cameras.

4) Object type

In order to demonstrate the effectiveness of our method in augmenting extensive object types, we compare the performance of three different policies that are respectively trained on:

1) 400 real-world demonstrations with 5 real-world objects (**Collected Five-Object 400**);
2) 1800 generated demonstrations with 5 reconstructed real-world objects (**Ours Five-Object 1800**);
3) 6400 generated demonstrations with 50 object types, which are generated with 3D Content Generation as mentioned in Sec. IV-C2 (**Ours 6400**).

During deployment, we select five real-word objects that are different from all the objects covered in training process. We report the result in Fig. 9. The policy trained on 50 object

TABLE II: **Performance when changing camera view.** We compare the success rate of different policies under two circumstances: novel camera view and moving camera view. The policies trained with camera-view-augmented demonstrations showcase significant improvement over baseline policies.

Data Source	Pick Object		Close Drawer		Pick-Place-Close		Average
	Novel View	Moving View	Novel View	Moving View	Novel View	Moving View	
Collected 200	6.7	0.0	16.7	13.3	0.0	0.0	6.1
Ours Pose-Only 200	0.0	0.0	13.3	6.7	0.0	0.0	5.0
Ours Pose-Only 1800	0.0	0.0	26.7	30.0	0.0	0.0	9.5
Ours 3200	**90.0**	**86.7**	**100.0**	**96.7**	**53.3**	**56.7**	**80.6**

types showcases better adaptability to novel object types, improving the success rate of baseline models by over 40%. This demonstrates the effectiveness of our data generation pipeline in utilizing off-the-shelf 3D Content Generation models to generalize policy to novel objects.

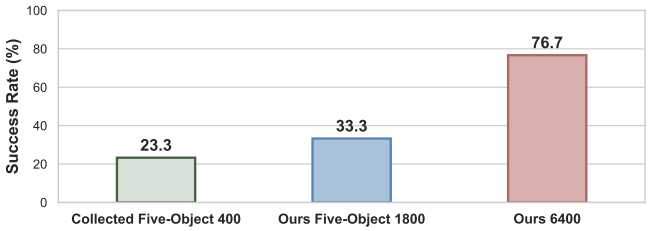


Fig. 9: **Performance on novel object types.** We evaluate policies on five unseen objects. The policy trained with objects from 3D Content Generation shows a salient edge over baseline policies.

5) Embodiment type

Our method supports generation of demonstrations across different embodiment types as mentioned in Sec. IV-C4. To prove that, we generate novel demonstrations for a UR5e robot equipped with a Robotiq 2F-85 gripper and deploy the learned policy directly in the real world. It is worth noting that policies trained on Franka Research 3 robot demonstrations fail to be deployed on UR5e robot due to frequent safety violations. Hence, we do not compare the performance of policies trained on embodiment-augmented demonstrations with any baseline models.

We present the performance of policies trained on augmented demonstrations in Fig. 10. The learned policy achieves nearly 100% success rate on an embodiment different from the one used for demonstration collection, demonstrating superior performance in cross-embodiment transfer.

VI. Limitations

Due to the limitations of naive 3D Gaussian Splatting, it is incapable of handling deformable objects. Additionally, the pipeline lacks physical constraints, making it unsuitable for contact-rich and dynamic tasks. However, recent advancements in Gaussian Splatting [57, 1, 62, 42] provide promising opportunities to address these challenges. Future work could apply these techniques to generate data for a wider range of tasks.

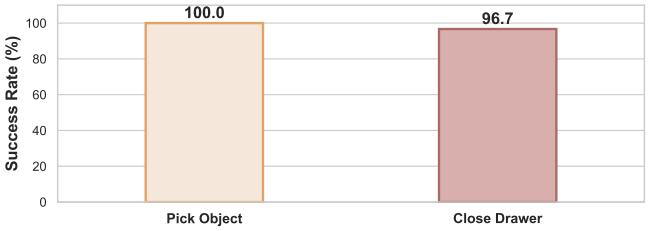


Fig. 10: **Performance on UR5e robot.** We evaluate the learned policy directly on the UR5e robot and achieve a nearly 100% success rate.

VII. Conclusion

In this work, we introduce **SplatForge**, a novel demonstration generation approach that requires only a single collected demonstration and generates diverse and high-quality data for policy learning. Comprehensive real-world experiments show that our approach significantly enhances the robustness of visuomotor policies when encountering various disturbances.

Acknowledgments

We sincerely thank Yang Tian, and Xiao Chen for fruitful discussions. This work is supported by the National Key R&D Program of China (2022ZD0160201), Shanghai Artificial Intelligence Laboratory, and China Postdoctoral Science Foundation (2023M741848).

References

[1] Jad Abou-Chakra, Krishan Rana, Feras Dayoub, and Niko Suenderhauf. Physically embodied gaussian splatting: A visually learnt and physically grounded 3d representation for robotics. In 8th Annual Conference on Robot Learning, 2024.

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. arXiv preprint arXiv:2303.08774, 2023.

[3] Ezra Ameperosa, Jeremy A Collins, Mrinal Jain, and Animesh Garg. Rocoda: Counterfactual data augmentation for data-efficient robot learning from demonstrations. arXiv preprint arXiv:2411.16959, 2024.

[4] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 5470–5479, 2022.

[5] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In Sensor fusion IV: control paradigms and data structures, volume 1611, pages 586–606. Spie, 1992.

[6] Ondrej Biza, Skye Thompson, Kishore Reddy Pagidi, Abhinav Kumar, Elise van der Pol, Robin Walters, Thomas Kipf, Jan-Willem van de Meent, Lawson LS Wong, and Robert Platt. One-shot imitation learning via interaction warping. arXiv preprint arXiv:2306.12392, 2023.

[7] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. arXiv preprint arXiv:2307.15818, 2023.

[8] Lawrence Yunliang Chen, Chenfeng Xu, Karthik Dharmarajan, Muhammad Zubair Irshad, Richard Cheng, Kurt Keutzer, Masayoshi Tomizuka, Quan Vuong, and Ken Goldberg. Rovi-aug: Robot and viewpoint augmentation for cross-embodiment robot learning. arXiv preprint arXiv:2409.03403, 2024.

[9] Zoey Chen, Sho Kiami, Abhishek Gupta, and Vikash Kumar. Genaug: Retargeting behaviors to unseen situations via generative augmentation. arXiv preprint arXiv:2302.06671, 2023.

[10] Zoey Chen, Zhao Mandi, Homanga Bharadhwaj, Mohit Sharma, Shuran Song, Abhishek Gupta, and Vikash Kumar. Semantically controllable augmentations for generalizable robot learning. The International Journal of Robotics Research, page 02783649241273686, 2024.

[11] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. The International Journal of Robotics Research, page 02783649241273668, 2023.

[12] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. arXiv preprint arXiv:2402.10329, 2024.

[13] Ethan Chun, Yilun Du, Anthony Simeonov, Tomas Lozano-Perez, and Leslie Kaelbling. Local neural descriptor fields: Locally conditioned object representations for manipulation. In 2023 IEEE International Conference on Robotics and Automation (ICRA), pages 1830–1836. IEEE, 2023.

[14] Murtaza Dalal, Min Liu, Walter Talbott, Chen Chen, Deepak Pathak, Jian Zhang, and Ruslan Salakhutdinov. Local policies enable zero-shot long-horizon manipulation. arXiv preprint arXiv:2410.22332, 2024.

[15] Linxi Fan, Guanzhi Wang, De-An Huang, Zhiding Yu, Li Fei-Fei, Yuke Zhu, and Anima Anandkumar. Secant: Self-expert cloning for zero-shot generalization of visual policies. arXiv preprint arXiv:2106.09678, 2021.

[16] Hao-Shu Fang, Chenxi Wang, Hongjie Fang, Minghao Gou, Jirong Liu, Hengxu Yan, Wenhai Liu, Yichen Xie, and Cewu Lu. Anygrasp: Robust and efficient grasp perception in spatial and temporal domains. IEEE Transactions on Robotics, 2023.

[17] Jian Gao, Chun Gu, Youtian Lin, Zhihao Li, Hao Zhu, Xun Cao, Li Zhang, and Yao Yao. Relightable 3d gaussians: Realistic point cloud relighting with brdf decomposition and ray tracing. In European Conference on Computer Vision, pages 73–89. Springer, 2025.

[18] Siddhant Haldar, Zhuoran Peng, and Lerrel Pinto. Baku: An efficient transformer for multi-task policy learning. arXiv preprint arXiv:2406.07539, 2024.

[19] Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In 2021 IEEE International Conference on Robotics and Automation (ICRA), pages 13611–13617. IEEE, 2021.

[20] Nicklas Hansen, Hao Su, and Xiaolong Wang. Stabilizing deep q-learning with convnets and vision transformers under data augmentation. Advances in neural information processing systems, 34:3680–3693, 2021.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[22] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. ACM Transactions on Graphics (ToG), 37(6):1–15, 2018.

[23] Alex Irpan, Alexander Herzog, Alexander Toshkov Toshev, Andy Zeng, Anthony Brohan, Brian Andrew Ichter, Byron David, Carolina Parada, Chelsea Finn, Clayton Tan, et al. Do as i can, not as i say: Grounding language in robotic affordances. In Conference on Robot Learning, number 2022, 2022.

[24] Mazeyu Ji, Ri-Zhao Qiu, Xueyan Zou, and Xiaolong Wang. Graspsplats: Efficient manipulation with 3d feature splatting. arXiv preprint arXiv:2409.02084, 2024.

[25] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. ACM Trans. Graph., 42(4):139–1, 2023.

[26] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics, 36(4), 2017.

[27] Georgios Kopanas, Thomas Leimkühler, Gilles Rainer, Clément Jambon, and George Drettakis. Neural point catacaustics for novel-view synthesis of reflections. ACM Transactions on Graphics (TOG), 41(6):1–15, 2022.

[28] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep

reinforcement learning from pixels. arXiv preprint arXiv:2004.13649, 2020.

[29] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. Advances in neural information processing systems, 33:19884–19895, 2020.

[30] Mara Levy, Siddhant Haldar, Lerrel Pinto, and Abhinav Shirivastava. P3-po: Prescriptive point priors for visuo-spatial generalization of robot policies. arXiv preprint arXiv:2412.06784, 2024.

[31] Xinhai Li, Jialin Li, Ziheng Zhang, Rui Zhang, Fan Jia, Tiancai Wang, Haoqiang Fan, Kuo-Kun Tseng, and Ruiping Wang. Robogsim: A real2sim2real robotic gaussian splatting simulator. arXiv preprint arXiv:2411.11839, 2024.

[32] Zhihao Liang, Qi Zhang, Ying Feng, Ying Shan, and Kui Jia. Gs-ir: 3d gaussian splatting for inverse rendering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 21644–21653, 2024.

[33] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13, pages 740–755. Springer, 2014.

[34] Guanxing Lu, Shiyi Zhang, Ziwei Wang, Changliu Liu, Jiwen Lu, and Yansong Tang. Manigaussian: Dynamic gaussian splatting for multi-task robotic manipulation. In European Conference on Computer Vision, pages 349–366. Springer, 2025.

[35] Zhao Mandi, Homanga Bharadhwaj, Vincent Moens, Shuran Song, Aravind Rajeswaran, and Vikash Kumar. Cacti: A framework for scalable multi-task multi-scene visual imitation learning. arXiv preprint arXiv:2212.05711, 2022.

[36] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. arXiv preprint arXiv:2108.03298, 2021.

[37] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. arXiv preprint arXiv:2310.17596, 2023.

[38] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In Proceedings of Robotics: Science and Systems, Delft, Netherlands, 2024.

[39] Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, et al. Open x-embodiment: Robotic learning datasets and rt-x models. arXiv preprint arXiv:2310.08864, 2023.

[40] Mohammad Nomaan Qureshi, Sparsh Garg, Francisco Yandun, David Held, George Kantor, and Abhisesh Silwal. Splatsim: Zero-shot sim2real transfer of rgb manipulation policies using gaussian splatting. arXiv preprint arXiv:2409.10161, 2024.

[41] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, et al. Grounded sam: Assembling open-world models for diverse visual tasks. arXiv preprint arXiv:2401.14159, 2024.

[42] Boxiang Rong, Artur Grigorev, Wenbo Wang, Michael J Black, Bernhard Thomaszewski, Christina Tsalicoglou, and Otmar Hilliges. Gaussian garments: Reconstructing simulation-ready clothing with photorealistic appearance from multi-view video. arXiv preprint arXiv:2409.08189, 2024.

[43] Hyunwoo Ryu, Hong-in Lee, Jeong-Hoon Lee, and Jongeun Choi. Equivariant descriptor fields: Se (3)-equivariant energy-based models for end-to-end visual robotic manipulation learning. arXiv preprint arXiv:2206.08321, 2022.

[44] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[45] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In European Conference on Computer Vision (ECCV), 2016.

[46] Younggyo Seo, Junsu Kim, Stephen James, Kimin Lee, Jinwoo Shin, and Pieter Abbeel. Multi-view masked world models for visual robotic manipulation. In International Conference on Machine Learning, pages 30613–30632. PMLR, 2023.

[47] Ola Shorinwa, Johnathan Tucker, Aliyah Smith, Aiden Swann, Timothy Chen, Roya Firoozi, Monroe Kennedy III, and Mac Schwager. Splat-mover: Multi-stage, open-vocabulary robotic manipulation via editable gaussian splatting. arXiv preprint arXiv:2405.04378, 2024.

[48] Anthony Simeonov, Yilun Du, Andrea Tagliasacchi, Joshua B Tenenbaum, Alberto Rodriguez, Pulkit Agrawal, and Vincent Sitzmann. Neural descriptor fields: Se (3)-equivariant object representations for manipulation. In 2022 International Conference on Robotics and Automation (ICRA), pages 6394–6400. IEEE, 2022.

[49] Stephen Tian, Blake Wulfe, Kyle Sargent, Katherine Liu, Sergey Zakharov, Vitor Guizilini, and Jiajun Wu. View-invariant policy learning via zero-shot novel view synthesis. arXiv preprint arXiv:2409.03685, 2024.

[50] Yang Tian, Sizhe Yang, Jia Zeng, Ping Wang, Dahua Lin, Hao Dong, and Jiangmiao Pang. Predictive inverse

dynamics models are scalable learners for robotic manipulation. arXiv preprint arXiv:2412.15109, 2024.

[51] Marcel Torne, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. arXiv preprint arXiv:2403.03949, 2024.

[52] Pietro Vitiello, Kamil Dreczkowski, and Edward Johns. One-shot imitation learning: A pose estimation perspective. arXiv preprint arXiv:2310.12077, 2023.

[53] Vitalis Vosylius and Edward Johns. Instant policy: In-context imitation learning via graph diffusion. arXiv preprint arXiv:2411.12633, 2024.

[54] Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation, 2023.

[55] Yuxuan Wu, Lei Pan, Wenhua Wu, Guangming Wang, Yanzi Miao, and Hesheng Wang. Rl-gsbridge: 3d gaussian splatting based real2sim2real method for robotic manipulation learning. arXiv preprint arXiv:2409.20291, 2024.

[56] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. arXiv preprint arXiv:2412.01506, 2024.

[57] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4389–4398, 2024.

[58] Jingyun Yang, Zi-ang Cao, Congyue Deng, Rika Antonova, Shuran Song, and Jeannette Bohg. Equibot: Sim (3)-equivariant diffusion policy for generalizable and data efficient learning. arXiv preprint arXiv:2407.01479, 2024.

[59] Jingyun Yang, Congyue Deng, Jimmy Wu, Rika Antonova, Leonidas Guibas, and Jeannette Bohg. Equivact: Sim (3)-equivariant visuomotor policies beyond rigid object manipulation. In 2024 IEEE International Conference on Robotics and Automation (ICRA), pages 9249–9255. IEEE, 2024.

[60] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10371–10381, 2024.

[61] Sizhe Yang, Yanjie Ze, and Huazhe Xu. Movie: Visual model-based policy adaptation for view generalization. Advances in Neural Information Processing Systems, 36, 2024.

[62] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 20331–20341, 2024.

[63] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. In European Conference on Computer Vision, pages 162–179. Springer, 2025.

[64] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 12–22, 2023.

[65] Tianhe Yu, Ted Xiao, Austin Stone, Jonathan Tompson, Anthony Brohan, Su Wang, Jaspiar Singh, Clayton Tan, Jodilyn Peralta, Brian Ichter, et al. Scaling robot learning with semantically imagined experience. arXiv preprint arXiv:2302.11550, 2023.

[66] Zhecheng Yuan, Tianming Wei, Shuiqi Cheng, Gu Zhang, Yuanpei Chen, and Huazhe Xu. Learning to manipulate anywhere: A visual generalizable framework for reinforcement learning. arXiv preprint arXiv:2407.15815, 2024.

[67] Xinyu Zhang and Abdeslam Boularias. One-shot imitation learning with invariance matching for robotic manipulation. arXiv preprint arXiv:2405.13178, 2024.

[68] Yuhang Zheng, Xiangyu Chen, Yupeng Zheng, Songen Gu, Runyi Yang, Bu Jin, Pengfei Li, Chengliang Zhong, Zengmao Wang, Lina Liu, et al. Gaussiangrasper: 3d language gaussian splatting for open-vocabulary robotic grasping. arXiv preprint arXiv:2403.09637, 2024.

APPENDIX

A. Applying Transformation and Scaling to 3D Gaussians

This section outlines how to apply transformations (translation, rotation) and scaling to 3D Gaussians.

The Gaussian primitive typically possesses three core properties: 1) a center position in three-dimensional space; 2) an orientation that specifies the tilt of its principal axes, commonly represented as a quaternion; 3) a scale indicating its width or narrowness. Additionally, Gaussian primitives can be enhanced with Spherical Harmonics (SH) to capture complex, direction-dependent color features.

When applying a transformation to the Gaussian primitive, the following steps should be taken: 1) update the center position by scaling, rotating, and then adding the translation offset; 2) update the orientation by combining the existing rotation with the new rotation; 3) adjust the scale by multiplying by the scaling factor; 4) rotate the Spherical Harmonics coefficients by using the Wigner D matrices. The code is shown in Listing 1.

B. Details of Demonstration Augmentation Process

We expand on the details of the demonstration augmentation process in this section. An illustration of augmented demonstrations is provided in Fig. 11.

1) Object pose

As mentioned in Sec. IV-C1, we transform the end-effector poses at key frames equivariantly according to the transformation that is applied to the target object. However, considering the symmetry of the gripper, we perform post-processing on the transformed end-effector pose.

Suppose the rotation of the transformed end-effector pose can be expressed as (r_x, r_y, r_z) in the format of XYZ Euler angles. We replace r_z with r'_z, which can be calculated as Equation 2,

$$r'_z = \begin{cases} r_z & -\frac{\pi}{2} \leqslant r_z \leqslant \frac{\pi}{2} \\ r_z + \pi & r_z < -\frac{\pi}{2} \\ r_z - \pi & r_z > \frac{\pi}{2}. \end{cases} \tag{2}$$

The resulting Euler angles (r_x, r_y, r'_z) form the final rotation of the end-effector, which prevents the end-effector from performing redundant rotation along its z-axis.

2) Camera view

As aforementioned in Sec. V-D3, we enumerate the hyperparameters of camera view augmentations and their range of randomization in Table III. Suppose the camera view in the expert demonstration has target point $O_c^{\text{expert}} = (x_c^0, y_c^0, z_c^0)$ and corresponding spherical coordinates $(r^0, \theta^0, \varphi^0)$. Thereby, the target point $O_c = (x_c, y_c, z_c)$ and corresponding spherical coordinates (r, θ, φ) are sampled from uniform distributions, ranging between $(x_c^0 \pm \Delta x_c, y_c^0 \pm \Delta y_c, z_c^0 \pm \Delta z_c, r^0 \pm \Delta r, \theta^0 \pm \Delta\theta, \varphi^0 \pm \Delta\varphi)$.

TABLE III: **Camera view augmentation hyperparameters and their range of randomization.**

Hyperparameter	Value
Δx_c	0.1(m)
Δy_c	0.1(m)
Δz_c	0.1(m)
Δr	0.2(m)
$\Delta\theta$	$\frac{\pi}{6}$
$\Delta\varphi$	$\frac{\pi}{6}$

3) Lighting condition

We present the hyperparameters of lighting condition augmentation in this section. First, we normalize the RGB values of each pixel with minimum value 0 and maximum value 1. Then, we stipulate that the hyperparameters are sampled from the following distributions:

$$(\Delta_r, \Delta_g, \Delta_b) \sim \mathcal{N}(\mathbf{0}, 0.1^2\mathbf{I}), \tag{3}$$

$$s_r, s_g, s_b \sim \text{Uniform}(0.3, 1.8), \tag{4}$$

$$o_r, o_g, o_b \sim \text{Uniform}(-0.3, 0.3). \tag{5}$$

C. Policy Architecture

As illustrated in Fig. 12, the policy processes two types of inputs: images and robot states. We use different encoders to tokenize each modality accordingly. For image inputs, the images are first passed through a ResNet-18 vision encoder to generate visual embeddings. We employ a linear layer to extract compact visual features. For the robot state, we encode it into state tokens using a multi-layer perceptron (MLP).

The multi-modal encoder in our model is based on a GPT-2 style transformer architecture. Before feeding the sequential image and state tokens into the transformer, we append readout tokens [ACT] to the end. These readout tokens attend to embeddings from different modalities, serving as action latents used for action prediction.

Encoded by the multi-modal encoder, the action latents generated by the [ACT] tokens are fed into the readout decoders to predict actions. The action decoder utilizes an MLP to transform the action latent into the action vector. We predict a chunk of 10 future actions. Compared to single-step action prediction, predicting multiple steps provides temporal action consistency and robustness to idle actions [11].

D. Training Details

During training, the input at each timestep consists of two images captured from two eye-on-base cameras, along with the robot state. The robot state includes both the arm state and the gripper state. The gripper state is binary, indicating whether the gripper is open or closed. For the Franka FR3 robot, the arm state is 7-dimensional, while for the UR5e robot, it is 6-dimensional.

The policy operates with a history length of 1, and the size of the action chunk is set to 10. During inference, we utilize temporal ensemble techniques to compute a weighted average of the multi-step actions.

The policy is trained using a single NVIDIA RTX 4090 GPU, with a batch size of 256 and a learning rate of 1e-4. Depending on the number of demonstrations, the policy is trained for varying numbers of epochs. The hyperparameters used during training are detailed in Table IV.

TABLE IV: **Policy training hyperparameters.**

Batch Size	256
Learning Rate	1e-4
Training Epochs	1400 (100 demonstrations) 1000 (200 demonstrations) 800 (400 demonstrations) 700 (800 demonstrations) 500 (1800 demonstrations) 300 (3200 demonstrations) 200 (6400 demonstrations)
Image Size	128*128
Optimizer	AdamW
History Length	1
Action Chunk Length	10

E. Illustration of Real-World Experiment Settings

We illustrate the experiment settings on lighting condition generalization in Fig. 13. The flashing light alternates between red and blue light at a frequency of 4Hz. Every lighting condition takes up 6 trials in a single experiment.

Besides, we present the real-world settings on appearance generalization in Fig. 14. Each scenario accounts for 5 trials in a single experiment.

```python
from torch import einsum
from pytorch3d.transforms import quaternion_to_matrix, matrix_to_quaternion
from e3nn import o3
import einops

def transform_shs(shs_feat, rotation_matrix):
    ## rotate shs
    P = torch.tensor([[0.0, 0.0, 1.0], [1.0, 0.0, 0.0], [0.0, 1.0, 0.0]]).cuda() # switch axes: yzx -> xyz
    permuted_rotation_matrix = torch.linalg.inv(P) @ rotation_matrix @ P
    rot_angles = o3._rotation.matrix_to_angles(permuted_rotation_matrix)

    # Construction coefficient
    D_1 = o3.wigner_D(1, rot_angles[0], - rot_angles[1], rot_angles[2])
    D_2 = o3.wigner_D(2, rot_angles[0], - rot_angles[1], rot_angles[2])
    D_3 = o3.wigner_D(3, rot_angles[0], - rot_angles[1], rot_angles[2])

    #rotation of the shs features
    one_degree_shs = shs_feat[:, 0:3]
    one_degree_shs = einops.rearrange(one_degree_shs, 'n␣shs_num␣rgb␣->␣n␣rgb␣shs_num')
    one_degree_shs = einsum(
        "...␣i␣j,␣...␣j␣->␣...␣i",
        D_1,
        one_degree_shs,
    )
    one_degree_shs = einops.rearrange(one_degree_shs, 'n␣rgb␣shs_num␣->␣n␣shs_num␣rgb')
    shs_feat[:, 0:3] = one_degree_shs

    two_degree_shs = shs_feat[:, 3:8]
    two_degree_shs = einops.rearrange(two_degree_shs, 'n␣shs_num␣rgb␣->␣n␣rgb␣shs_num')
    two_degree_shs = einsum(
        "...␣i␣j,␣...␣j␣->␣...␣i",
        D_2,
        two_degree_shs,
    )
    two_degree_shs = einops.rearrange(two_degree_shs, 'n␣rgb␣shs_num␣->␣n␣shs_num␣rgb')
    shs_feat[:, 3:8] = two_degree_shs

    three_degree_shs = shs_feat[:, 8:15]
    three_degree_shs = einops.rearrange(three_degree_shs, 'n␣shs_num␣rgb␣->␣n␣rgb␣shs_num')
    three_degree_shs = einsum(
        "...␣i␣j,␣...␣j␣->␣...␣i",
        D_3,
        three_degree_shs,
    )
    three_degree_shs = einops.rearrange(three_degree_shs, 'n␣rgb␣shs_num␣->␣n␣shs_num␣rgb')
    shs_feat[:, 8:15] = three_degree_shs

    return shs_feat

def transform_gaussian(gaussian, T, scale=1.0, transform_sh=False):
    # transform xyz
    xyz = gaussian.get_xyz
    xyz = xyz * scale
    xyz = xyz @ T[:3, :3].T + T[:3, 3]
    gaussian._xyz = xyz
    # tranform rotation
    rotation = gaussian.get_rotation
    rotation = quaternion_to_matrix(rotation)
    rotation = T[:3, :3] @ rotation
    rotation = matrix_to_quaternion(rotation)
    gaussian._rotation = rotation
    # transform scale
    gaussian._scaling = torch.log(torch.exp(gaussian._scaling) * scale)

    if transform_sh:
        # transform sh
        sh = gaussian._features_rest
        sh = transform_shs(sh, T[:3, :3])
        gaussian._features_rest = sh

    return gaussian
```

Listing 1: **Code for transformation of 3D Gaussians.**

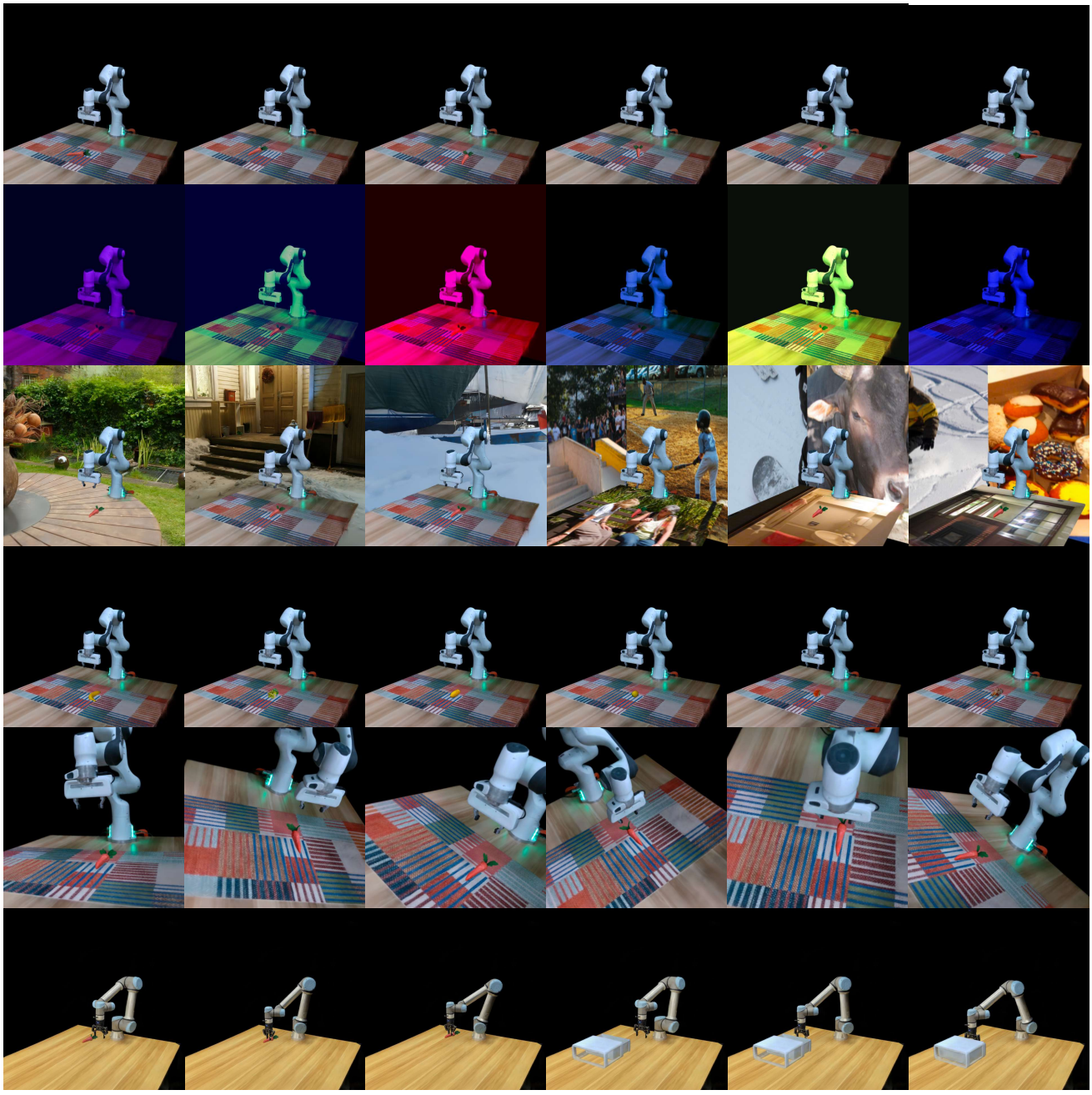


Fig. 11: **Illustration of augmented demonstrations.** Type of generalization from the top row to the bottom row: object pose, lighting condition, scene appearance, object type, camera view, and embodiment type.

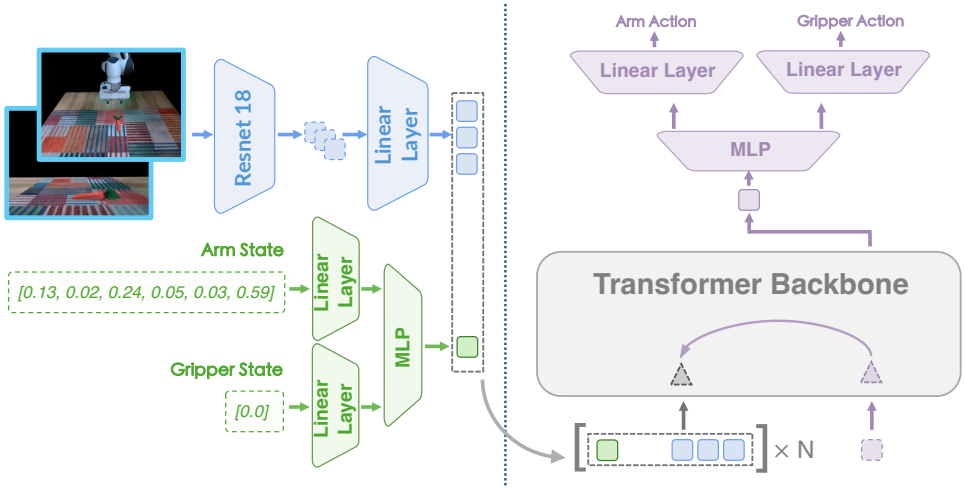
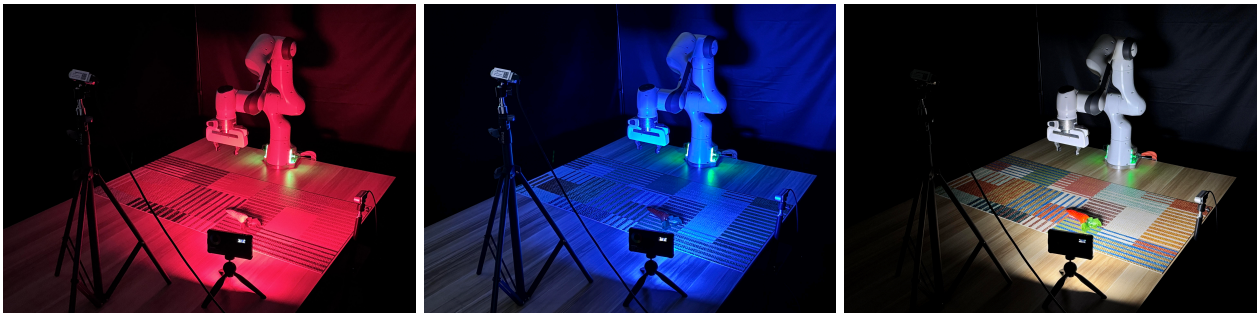
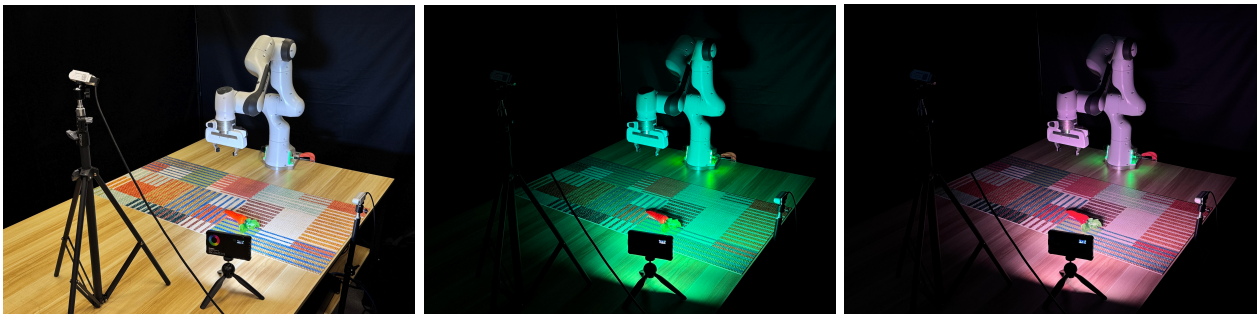


Fig. 12: **Policy architecture.**

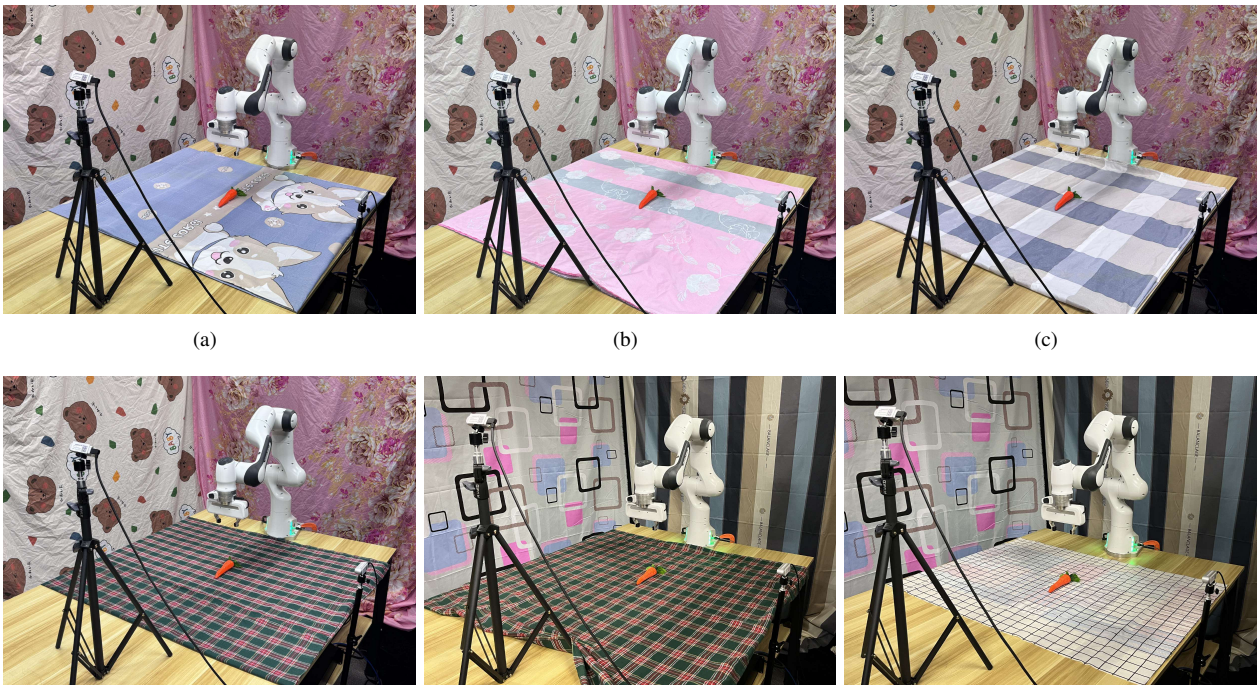


(a) Flashing light (Red) (b) Flashing light (Blue) (c) Dark light



(d) Bright light (e) Green light (f) Yellow light

Fig. 13: **Illustration of real-world experiment on lighting generalization.**



(a) (b) (c)

(d) (e) (f)

Fig. 14: **Illustration of real-world experiment on appearance generalization.**