Stem-OB: Generalizable Visual Imitation Learning with Stem-Like Convergent Observation through Diffusion Inversion

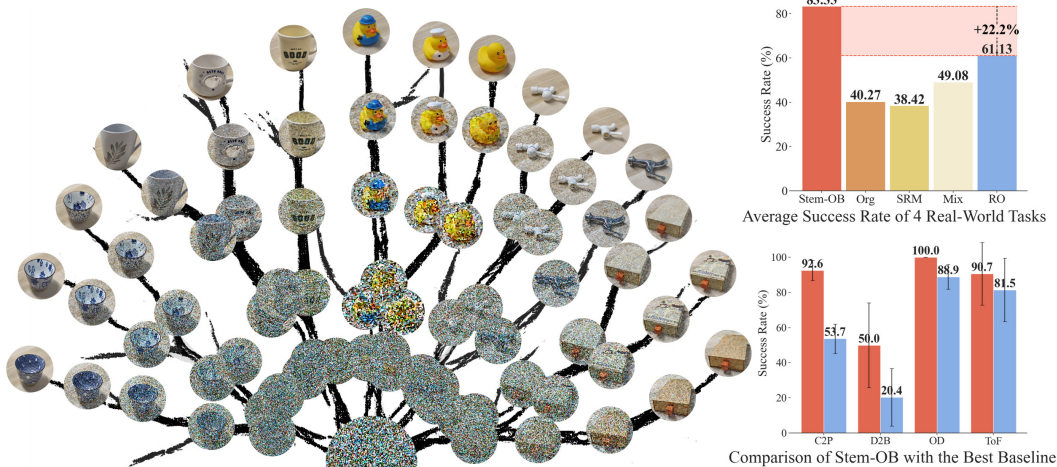**Anonymous authors**
Paper under double-blind review

Figure 1: **Left:** The tree of *Stem-OB* inversion is composed of different objects progressively inverted through a diffusion inversion process. Moving downward alone the tree's branches, objects of different textures, appearances, and categories gradually get closer, eventually converging into the same root of Gaussian noise, where they are completely indistinguishable. **Right:** Real-world tasks success rate, where *Stem-OB* showcases a significant improvement.

Abstract

Visual imitation learning methods demonstrate strong performance, yet they lack generalization when faced with visual input perturbations like variations in lighting and textures. This limitation hampers their practical application in real-world settings. To address this, we propose **Stem-OB** that leverages the inversion process of pretrained image diffusion models to suppress low-level visual differences while maintaining high-level scene structures. This image inversion process is akin to transforming the observation into a shared representation, from which other observations also stem. *Stem-OB* offers a simple yet effective plug-and-play solution that stands in contrast to data augmentation approaches. It demonstrates robustness to various unspecified appearance changes without the need for additional training. We provide theoretical insights and empirical results that validate the efficacy of our approach in simulated and real settings. *Stem-OB* shows an exceptionally significant improvement in real-world robotic tasks, where challenging light and appearance changes are present, with an average increase of **22.2%** in success rates compared to the best baseline. See our website for more videos.

1 Introduction

Visual Imitation Learning (IL), where an agent learns to mimic the behavior of the demonstrator by learning a direct mapping from visual observations to low-level actions, has gained popularity in recent real-world robot tasks (Chi et al., 2023; Zhao et al., 2023; Wang et al., 2023a; Chi et al., 2024;

Ze et al., 2024). Despite the versatility demonstrated by visual IL, learned policies are often brittle and fail to generalize to unseen environments, even minor perturbations such as altering lighting conditions or changing the texture of the object may lead to failure of the learned policy (Xie et al., 2023; Yuan et al., 2024b). The underlying reason is that the high-dimensional visual observation space is redundant with virtually infinite variations in appearance that are irrelevant to the task and hard to generalize.

As human beings, we can easily manipulate objects that have different appearances. For example, we can pick up a cup of coffee regardless of its color, texture, or the lighting condition of the room. This is partially because our visual system is capable of abstracting the high-level semantics of the scene, such as the silhouette of the object, the structure and arrangement of different objects, etc in a hierarchical manner (Hochstein & Ahissar, 2002), effectively merging scenes with perceptual differences to similar "meta" observations.

Augmentation techniques such as Spectrum Random Masking (SRM) (Huang et al., 2022) and Mixup (Zhang et al., 2018) remove details from observations to encourage the model to focus on structural features; however, they lack the ability to distinguish between low-level and high-level features. It is preferable if we can sweep the photometrical differences while maintaining the high-level structure for the scene. Achieving this requires a semantic understanding of the observations, and naively perturbing the data with Gaussian noise can lead to irreversible information loss.

Pretrained large image diffusion models, such as Stable Diffusion (Rombach et al., 2022; Esser et al., 2024), embed essential world knowledge for visual understanding. Apart from synthesizing new images from random noise, these models are capable to perform a reverse procedure called inversion (Song et al., 2022), which converts an image back to the space of random noises. A recent study (Yue et al., 2024) indicates that this inversion process selectively eliminates information from the image. Rather than uniformly removing information from different semantic hierarchies, it will push those images with similar structures closer in the early stages of the inversion process. Inversion is like the reprogramming of a differentiated cell back to a stem cell, which bears the totipotency to differentiate into any cell type. This characteristic aligns perfectly with our will of enhancing the robustness and generalizability of visual IL algorithms to visual variations. To distill such property into a visual IL policy, we propose an imitation learning pipeline which applies diffusion inversion to the visual observations. We name our method *Stem-OB* to highlight the similarity between the inversed observation and the stem cell in biology, as illustrated in Figure 1.

To be specific, our method is as simple as inverting the image for reasonable steps before sending them to the downstream visual IL algorithms. The number of steps is chosen empirically to balance removing irrelevant details without erasing essential high-level information. From this perspective, our approach fundamentally distinguishes from generative augmentation methods, which aim to enrich the training dataset with more unseen objects and appearances (Yu et al., 2023; Mandlekar et al., 2023). Moreover, *Stem-OB* is indifferent to many unspecified appearance changes, in contrast to augmentation-based methods that must concentrate on a few selected types of generalization, thereby introducing inevitable inductive biases.

We provide theoretical analysis and a user study to support our claim that *Stem-OB* can effectively merge scenes with perceptual differences to similar "stem observations". Empirical study demonstrates the effectiveness of our approach in a variety of simulated and real-world tasks and a range of different perturbations. *Stem-OB* proves to be particularly effective in real-world tasks where appearance and lighting changes hamper the other baselines, establishing an overall improvement in the success rate of **22.2%**. What's better, no inference time inversion is required for *Stem-OB* to take effect, making the deployment of our method virtually free of computational cost.

2 RELATED WORKS

2.1 VISUAL IMITATION LEARNING AND GENERALIZATION

Visual Imitation Learning (VIL) is a branch of Imitation learning (IL) that focuses on learning action mappings from visual observations. It typically follows two approaches: directly imitating expert policies, as in behavior cloning and DAgger (Ross et al., 2011), or inferring a reward function that aligns the agent's behavior with expert demonstrations, like inverse reinforcement learning (Ng &
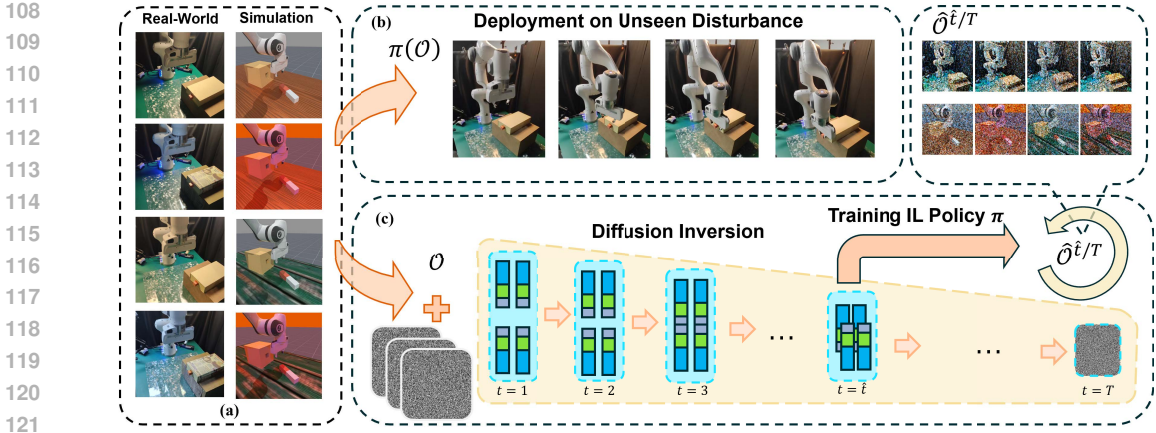
Figure 2: **Overview of _Stem-OB_**: **(a)**. _Stem-OB_ has been evaluated in both real-world and simulated environments. **(b)**. The trained visual IL policies are directly applied to the original observation space $\mathcal{O}$, demonstrating robustness to unseen environmental disturbances. **(c)**. We train the visual IL policy π on the diffusion-inversed latent space $\hat{\mathcal{O}}^{\hat{t}/T}$, where $\hat{t}$ denotes a specific inversion step out of a total of T. Each composite rectangle in the diffusion inversion process, made up of three smaller sections, represents the latent vector of an image, with the smaller section depict the finer attributes (gray). During the inversion process, finer attributes converge earlier than coarser ones.

Russell, 2000) and GAIL (Ho & Ermon, 2016). The former approach, favored in recent works due to its scalability and practicality in complex, real-world tasks, has led to several advancements. Notable methods include Diffusion Policy (Chi et al., 2023), which leverages a diffusion model (Ho et al., 2020) to maximize the likelihood of expert actions, and Action Chunk Transformer (Zhao et al., 2023) that uses a Transformer (Vaswani, 2017).

To enhance the generalization and robustness of visual imitation learning algorithms, various approaches have been explored (Cetin & Celiktutan, 2021). For instance, Li et al. (2023) leverage trajectory and step level similarity through an estimated reward function, while Wan et al. (2023) improve visual robustness by separating task-relevant and irrelevant dynamics models before applying the GAIL framework (Ho & Ermon, 2016). Zhang et al. (2023a) use mutual information constraints and adversarial imitation learning to create compact representations that generalize to unseen environments. However, these approaches are not directly applicable to methods like diffusion policy, which focus on imitation without reward functions or dynamics models. Zheng et al. (2023) propose to filter extraneous action subsequence, yet their focus is not on visual perturbations. Most relevant to our setting, several works in robust visual reinforcement learning have explored adding noise in image or frequency domain to improve generalizability (Huang et al., 2022; Lee & Hwang, 2024; Yuan et al., 2024a), however, they lack the semantic understanding of the augmentation process.

2.2 INVERSION OF DIFFUSION MODELS AND ITS APPLICATION

Diffusion model inversion aims to recover an initial noise distribution from a given image, enabling image reconstruction from the same noise via backward denoising. A common approach is DDIM inversion (Song et al., 2022), which estimates the previous noise using predictions from the current diffusion step, though the approximation will introduce cumulative errors. To address this issue, several methods employ learnable text embeddings (Mokady et al., 2023; Miyake et al., 2023), fixed-point iterations (Pan et al., 2023; Garibi et al., 2024), or gradient-based techniques (Samuel et al., 2024) to refine the result. Another approach, based on the stochastic DDPM scheduler (Ho et al., 2020), reconstructs noisy images at each diffusion step (Huberman-Spiegelglas et al., 2024; Brack et al., 2024). In contrast to the DDIM inversion methods, the noise $\tilde{\epsilon}_t$ of each step is statistically independent, making them ideal for our application since we can obtain the noisy image of a certain step without the need to recover other steps, greatly reducing the calculation cost.

Diffusion Inversion is the crucial part of diffusion-based image editing methods (Meng et al., 2021; Kawar et al., 2023), which typically involves first inverting the diffusion process to recover the noise latent, then denoise the latent with desired editing conditions. Recent works also explore to apply

attention control over the denoising process to improve the fidelity of the edited image Hertz et al. (2022); Tumanyan et al. (2023), and have shown promising application in robot learning tasks Gao et al. (2024). Beyond that, inversion is also used in tasks like concept extraction Huang et al. (2023) and personalization Gal et al. (2022). Most recently, Wang & Chen (2024) proposed the use of diffusion inversion to interpolate between image categories to improve classification performance. And Wang et al. (2023b) uses diffusion inversion to erase out sub-optimal trajectories from the dataset.

3 PROBLEM DEFINITION

Given a dataset of observation $\mathcal{O}$ and action $\mathcal{A}$ pairs, the goal of VIL is to learn a policy $\pi_\theta(\mathcal{A}|\mathcal{O})$ that maps observations to actions. The policy is typically parameterized by a neural network with parameters θ, and is trained to minimize the negative log-likelihood of the actions.

To achieve the goal of generalizing to unseen environments, we seek a method to preprocess or transform the observations $\mathcal{O}$ such that task-irrelevant details are suppressed while preserving the high-level semantic structure that is critical for the task. The problem can be transformed into learning a transformation $\mathcal{T}$ as the input to the policy $\pi_\theta(\mathcal{A}|\mathcal{T}(\mathcal{O}))$, where $\mathcal{T}(\mathcal{O})$ is the transformed observation emphasizing high-level semantics while removing irrelevant details.

4 PRELIMINARY

We begin by outlining the fundamentals of Diffusion Inversion. A diffusion model operates with two passes: a backward denoising pass, which generates an image from noise, and a forward pass, where noise is incrementally added to an image until it becomes pure Gaussian noise. This forward process is a Markov chain that starts with $\boldsymbol{x}_0$, and gradually adds noise to obtain latent variables $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T$. Each step in this process is a Gaussian transition following the common form

$$\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sigma_t \boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{x}_t|\sqrt{\alpha_t}\boldsymbol{x}_t, \sigma_t^2 \boldsymbol{I}) \tag{1}$$

where $\alpha_t \in (0,1)$ represents the scheduler parameter at each step t, while σ_t characterizes the variance of the Gaussian noise $\boldsymbol{\epsilon}_t$ introduced at each step. In Denoising Diffusion Probabilistic Models (DDPM) (Ho et al., 2020), $\sigma_t = \sqrt{1 - \alpha_t}$. Consequently, equation Eq. (1) can be reformulated as Eq. (2) by applying the cumulative product $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{x}_t|\sqrt{\bar{\alpha}_t}\boldsymbol{x}_{t-1}, (1 - \bar{\alpha}_t)\boldsymbol{I}) \tag{2}$$

Diffusion inversion is similar to the forward process in that they both maps an image to a noise, however, inversion tries to preserve the image's information and obtain the specific noise that can reconstruct the image during a backward denoising process.

DDPM inversion. We follow the DDPM inversion proposed in Huberman-Spiegelglas et al. (2024),

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1 - \bar{\alpha}_t}\tilde{\boldsymbol{\epsilon}}_t \tag{3}$$

The DDPM inversion we consider here differs slightly from Eq. (2), as $\tilde{\boldsymbol{\epsilon}}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ are mutually independent distributions, in contrast to the highly correlated $\boldsymbol{\epsilon}_t$ in Eq. (2). As mentioned by Huberman-Spiegelglas et al. (2024), the independence of $\tilde{\boldsymbol{\epsilon}}_t$ results in a sequence of latent vectors where the structures of $\boldsymbol{x}_0$ are more strongly imprinted into the noise maps. An error reduction step is conducted in reverse order after the diffusion forward process to improve image reconstruction accuracy during the denoising process:

$$\boldsymbol{z}_t = \boldsymbol{x}_{t-1} - \hat{\mu}(\boldsymbol{x}_t)/\sigma_t, \quad \boldsymbol{x}_{t-1} = \hat{\mu}(\boldsymbol{x}_t) + \sigma_t \boldsymbol{z}_t \tag{4}$$

DDIM inversion. We follow the DDIM inversion proposed in Song et al. (2022), where at each forward diffusion step

$$\boldsymbol{x}_t = \sqrt{\frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}}\boldsymbol{x}_{t-1} + \Big(\sqrt{\frac{1}{\bar{\alpha}_t} - 1} - \sqrt{\frac{1}{\bar{\alpha}_{t-1}} - 1}\Big)\boldsymbol{\epsilon_\theta}(\boldsymbol{x}_{t-1}, t, \mathcal{C}) \tag{5}$$

Note that the noise $\boldsymbol{\epsilon_\theta}(\boldsymbol{x}_{t-1}, t, \mathcal{C})$ is now generated by a network trained to predict the noise based on the previous vector $\boldsymbol{x}_{t-1}$ and the text embedding $\mathcal{C}$ which, in our case, is $\emptyset$.

5 METHOD

In this section, we introduce the intuition and implementation of our framework. We first propose the intuition of applying inversion on observations through theoretical analysis based on attribute loss, a diffusion-based measurement of image semantic similarity. Then, we conduct an illustrative experiment and a user study to validate our intuition. Finally, we explain how to practically implement *Stem-OB* and incorporate diffusion inversion into a visual imitation learning framework.

5.1 INTUITION DEVIATION BY ATTRIBUTE LOSS

Intuitively, as the diffusion inversion process moves forward, a source image and another variation of it become increasingly indistinguishable. The variation here could be low-level changes like lightning conditions, but also includes semantic changes such as replacing an object. If there are two different variations, we want to show that as the inversion step increases, the pair with minor alterations will become indistinguishable sooner than the pair with larger and structural changes. We borrow the definition of attribute loss from Yue et al. (2024) to quantify the semantic overlapping of the two images at time step t during a inversion process:

$$loss(\boldsymbol{x}_0, \boldsymbol{y}_0, t) = \frac{1}{2}\mathrm{OVL}(q(\boldsymbol{x}_t|\boldsymbol{x}_0), q(\boldsymbol{y}_t|\boldsymbol{y_0})) \tag{6}$$

where $\boldsymbol{x}_0$ and $\boldsymbol{y}_0$ are the latent variables of the two images, and OVL is the overlapping coefficient quantifying the overlapping area of two probability density functions . For an inversion process where each step follows a Gaussian transition, it takes the form $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sigma\epsilon \sim \mathcal{N}(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, \sigma^2\mathbf{I})$. The OVL can be further calculated as the overlapping area of two Gaussian distributions, i.e.,

$$loss(\boldsymbol{x}_0, \boldsymbol{y}_0, t) = \frac{1}{2}\Big[1 - \mathrm{erf}(\frac{||\sqrt{\bar{\alpha}_t}(\boldsymbol{y}_0 - \boldsymbol{x}_0)||}{2\sqrt{2}\sigma})\Big] \tag{7}$$

where erf is the error function, which is strictly increasing. Given a source image $\boldsymbol{x}_0$ and its variations $\hat{\boldsymbol{x}}_0$ and $\tilde{\boldsymbol{x}}_0$, with $\tilde{\boldsymbol{x}}_0$ undergoing a larger variation than $\hat{\boldsymbol{x}}_0$, the following conclusion can be easily observed under the same diffusion scheduling:

$$\tau(\boldsymbol{x_0}, \hat{\boldsymbol{x}}_0, \rho) < \tau(\boldsymbol{x_0}, \tilde{\boldsymbol{x}}_0, \rho), \ s.t. ||\hat{\boldsymbol{x}}_0 - \boldsymbol{x}_0|| < ||\tilde{\boldsymbol{x}}_0 - \boldsymbol{x}_0|| \tag{8}$$

Here, $\tau(\boldsymbol{x}_0, \boldsymbol{y}_0, \rho) = \inf\{t > 0 \mid \mathrm{loss}(\boldsymbol{x}_0, \boldsymbol{y}_0, t) > \rho\}$ represents the earliest step where the loss between $\boldsymbol{x}_0$ and $\boldsymbol{y}_0$ exceeds the threshold ρ, and $||\cdot||$ measures the difference between an image and its variation. Eq. (8) provides a theoretical grounding for our intuition: images with fine-grained attribute changes tend to become indistinguishable sooner than those with coarse-grained modifications under identical diffusion schedules.

We can further derive the attribute loss for DDPM inversion

$$loss_{DDPM}(\boldsymbol{x}_0, \boldsymbol{y}_0, t) = \frac{1}{2}\Big[1 - \mathrm{erf}(\frac{||\sqrt{\bar{\alpha}_t}(\boldsymbol{y}_0 - \boldsymbol{x}_0)||}{2\sqrt{2(1 - \bar{\alpha}_t)}})\Big] \tag{9}$$

Additionally, we derive that the attribute loss for DDIM inversion exhibits a similar form under certain assumptions. The detailed derivation can be found in Appendix A.2.

$$loss_{DDIM}(\boldsymbol{x}_0, \boldsymbol{y}_0, t) = \frac{1}{2}\left[1 - \mathrm{erf}\Big(\frac{||(\boldsymbol{y}_0 - \boldsymbol{x}_0)||}{2\sqrt{2\sum_{i=1}^{t}\frac{1}{\bar{\alpha}_i}\Big(\sqrt{\frac{1}{\bar{\alpha}_i} - 1} - \sqrt{\frac{1}{\bar{\alpha}_{i-1}} - 1}\Big)^2}}\Big)\right] \tag{10}$$

Because $\bar{\alpha}_t \in (0, 1)$ is strictly decreasing, the attribute loss tends to increase as the time step increases. Furthermore, as discussed in Yue et al. (2024), this attribute loss is equivalent to how likely the DM falsely reconstruct $\boldsymbol{x}_t$ sampled from $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ closer to $\boldsymbol{y}_0$ instead of $\boldsymbol{x}_0$, and vise versa.

5.2 ILLUSTRATIVE EXPERIMENT

In Sec. 5.1, we made a key assumption that semantically similar images are closer in latent space of the diffusion models , leading to the conclusion that such images exhibit higher attribute loss at a given inversion step (Eq. (8)). While this assumption is backed by recent study on the zero-shot semantic correspondence ability on diffusion latents (Tang et al., 2023; Zhang et al., 2023b) , we further conducted an illustrative experiment to further support it. To validate this assumption, we used a set of images, denoted as $\mathcal{I}$, from the real-world task variations described in Sec. 6.1.1. Specifically, we selected the 4 generalization objects in the real-world tasks from 5 categories and calculated the pairwise distance between images on their latent representations, both for intra-category and cross-category image pairs. The results are presented in Tab. 1, where each entry represents the average distance between two categories. It is clear that the diagonal entries indicating intra-category similarity exhibit lower loss compared to cross-category images, which justifies our claim.

We then conducted a user study to validate that similar images exhibit higher attribute loss at a given inversion step. We recruited 51 participants and presented each with image pairs from $\mathcal{I}$ after specific steps of diffusion inversion. The inversion steps were systematically sampled from 15 to 45, out of a total 50 steps inversion and with intervals of 5. Each participant was asked to determine whether the image pairs depicted the same object, and we recorded the proportion of incorrect responses.

The experimental results in Fig. 3 show that at inversion step $t_1 = 20$, the incorrect response rate within the same category starts to increase. In contrast, the error rate for objects from different categories only started to rise at the inversion step $t_2 = 35$. This indicates that inversion makes objects within the same category harder to distinguish earlier than affecting the distinction between objects from different categories. Therefore, these results support our claim that similar images exhibit higher attribute loss at a given inversion step.

Table 1: **By-category image semantic similarity.**

Categories	Bowl	Cup	Drawer	Duck	Faucet
Bowl	156.65	172.20	172.81	172.82	167.73
Cup	172.20	154.63	165.43	167.83	167.37
Drawer	172.81	165.43	144.63	161.82	161.57
Duck	172.82	167.83	161.82	140.51	147.00
Faucet	167.73	167.37	161.57	147.00	145.34
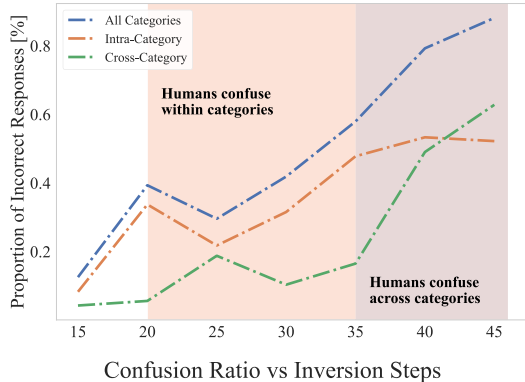
Figure 3: **User study confusion proportions.**

5.3 IMPLEMENTATION OF STEM-OB

Following the theoretical analysis and experimental validation, we propose a practical implementation of our framework. Our method applies diffusion inversion to converge different observations during training, and we find that the model gains robustness improvement in the original observation space during test time. The detailed implementation is described below.

Training. The training process begins by applying diffusion inversion to each observation. We define the following partial diffusion inversion process for an observation $\boldsymbol{o}_i \in \mathcal{O}$:

$$\hat{\boldsymbol{o}}_i^{t/T} = \boldsymbol{f}(\boldsymbol{o}_i, t, T) \tag{11}$$

Here, $\hat{\boldsymbol{o}}_i^{t/T} \in \hat{\mathcal{O}}^{t/T}$ denotes the observation after t out of T steps of diffusion inversion from the original observation $\boldsymbol{o}_i \in \mathcal{O}$. The function $\boldsymbol{f}(\cdot)$ applies t steps of diffusion inversion to $\boldsymbol{o}$ using any inversion methods. We select the DDPM inversion method for its efficiency and effectiveness in our experiments, and the selection of t and total inversion step T is discussed in Sec. 6.5. The visual imitation learning algorithms are then trained on the inversion-altered space $\hat{\mathcal{O}}^{t/T}$. Note that we empirically find the error reduction step of DDPM inversion in Eq. (4) is not significant for the performance, so we omit it and approximate the partial inversion process with regard to Eq. (3) only. In this way, we avoid the full inversion process or reverse-order corrections that involve extensive Diffusion Model inference. This approach significantly reduces preprocessing time with minimal performance impact, achieving an average time for preprocessing of 0.2s per image.
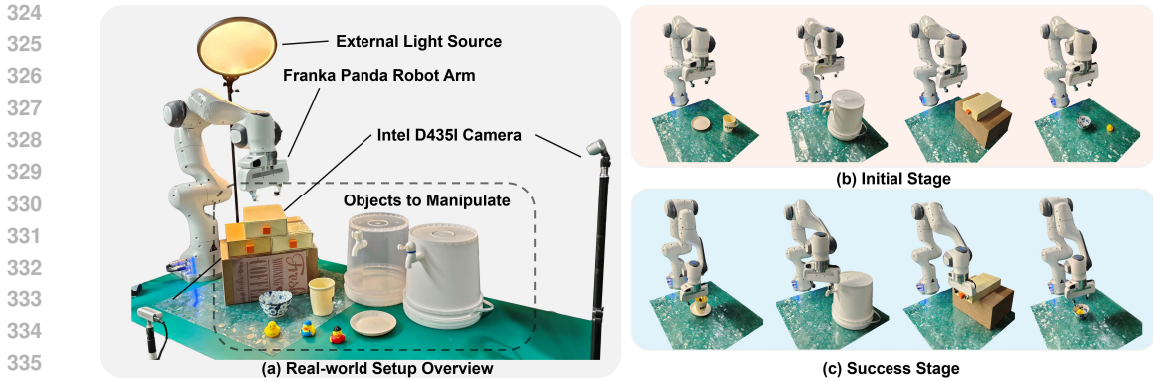
Figure 4: **Realworld Setup**: (a) Overview of the whole setup. (b)(c) These tasks are performed by the robot in a real-world environment, from left to right: ***Cup2Plate***, ***Turn on Faucet***, ***Open Drawer***, and ***Duck2Bowl***. The figure showcases the initial and final states of the tasks.

Testing. Despite training is entirely conducted on the inversion-altered space, the imitation learning algorithm of our choice showcases surprising zero-shot adaptation ability to the original observation space. We adhere to the original observation space $\mathcal{O}$ during testing, which means essentially no changes are made to the downstream visual imitation learning algorithms. This approach demonstrates improved robustness to environmental variations without any inference-time overhead, and is suitable for any other test-time augmentation techniques.

6 EXPERIMENTS

In this section, we present a comprehensive set of experiments using Diffusion Policy (DP) (Chi et al., 2023) as our visual imitation learning backbone. Since Stable Diffusion is pretrained on real-world images (Chi et al., 2023; Rombach et al., 2022), and a increasing interest in deploying visual imitation learning in real-world scenarios in the community (Paolo et al., 2024), we focus primarily on evaluating *Stem-OB* on real-world tasks. We extend the testing to simulated environments as well for further benchmarking. To assess the robustness of our method against visual appearance variations, we design experiments featuring different object textures and lighting conditions. We compare *Stem-OB* against several baselines to validate its effectiveness.

6.1 EXPERIMENT SETUP

6.1.1 REAL-WORLD TASKS

We conduct real-world experiments using a Franka Emika Panda Arm, and two RealSense D435I cameras positioned at different angles for RGB inputs. The setup is shown in Fig. 4. Our experiments focus on four tasks, as described below:

> ***Cup to Plate*** (C2P): The robot arm picks up a cup and places it on a plate, with variation introduced by changing tablecloth patterns.
> ***Duck to Bowl*** (D2B): The robot grasps a toy duck and places it into a bowl, with variations introduced by altering the duck's appearance.
> ***Open Drawer*** (OD): The robot arm grabs a drawer handle and pulls it open, with variations introduced by modifying the drawer's visual characteristics.
> ***Turn on Faucet*** (ToF): The robot arm turns on a faucet, with variations introduced by altering the appearance of the faucet and bucket.

In addition to the visual variations mentioned above, all four tasks above involve changes in *lighting conditions*, i.e., cool vs. warm light. All the variations only happen during testing time, with the training set contain only a basic setting. The object locations in training set are randomly initialized within a specified area, and 100 demonstrations are collected per task. For testing, nine predefined target positions are used. Further details of the environmental variations can be found in Appendix C.1.

6.1.2 SIMULATION TASKS

Our simulation experiments consider different tasks within two frameworks: a photorealistic simulation platform SAPIEN 3 (Xiang et al., 2020) and a less realistic framework MimicGen (Mandlekar et al., 2023).

SAPIEN 3 provides physical simulation for robots, rigid bodies, and articulated objects. It delivers high-fidelity visual and physical simulations that closely approximate real-world conditions. We leverage the ManiSkill 3 dataset (Gu et al., 2023; Tao et al., 2024), collected on SAPIEN 3, for benchmarking. Specifically, we select four tasks from ManiSkill for evaluation:

PushCube: The robot arm pushes a cube to a target location.
PegInsertionSide: The robot arm inserts a peg into a hole.
PickCube: The robot arm picks up a cube and places it on a target location.
StackCube: The robot arm stacks one cube over the other.

During testing, we vary the background and lighting conditions in all tasks to generate different visual appearances. 50 episodes are tested for each setting. Details of the environmental variations can be found in Appendix C.2.
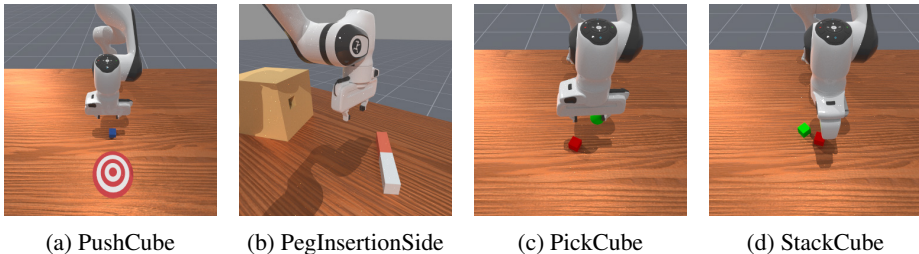

(a) PushCube   (b) PegInsertionSide   (c) PickCube   (d) StackCube

Figure 5: **SAPIEN 3 environments.** The figure showcases the visual appearance and task configuration of each setting.

MimicGen is a system for generating large diverse datasets from a small number of human demonstrations. We utilize the benchmark to evaluate the performance of our approach and conduct experiments on a variety of tasks: *MugCleanup*, *Threading*, *ThreePieceAssembly* and *Coffee*. MimicGen offers numerous task variations, each characterized by different initial distributions. We adopt the default initial distribution (D0) for both training data and test environments. For evaluation, we employ a single image as the input to the policy, using 500 samples out of a total of 1000 demos for training. We alter the table texture and object appearances to create different test environments. 300 episodes are tested for each setting of all the tasks. Detailed descriptions of the environmental variations can be found in Appendix C.3.

6.2 BASELINES

We compare *Stem-OB* with several data augmentation methods aimed at improving the generalizability of visual imitation learning. The baselines include **SRM** (Huang et al., 2022), which modifies images by adding random noise in the frequency domain, **Mix-Spectrum** (Mix) (Lee & Hwang), which enhances the agent's focus on semantic content by blending original images with randomly selected reference images from the same dataset in the frequency domain, and **original images** (Org) without any modification. Additionally, as highlighted in (Yuan et al., 2024b), **Random Overlay** (RO) improves generalization in real-world experiments by blending original images with random real-world photos in the image domain. Therefore, we include an additional study using Random Overlay in our real-world experiments.

6.3 EXPERIMENT RESULTS

6.3.1 REAL-WORLD EXPERIMENTS

In this section, we report the average success rate across all predefined test positions for each task. As shown in Tab. 2, *Stem-OB* consistently outperforms the baseline models in all settings, demon-

Table 2: **Evaluation of real-world experiments.** Train.: evaluations in the same settings as the training dataset. Gen.: evaluations under different visual perturbations for generalizability analysis. All: evaluations including both Train. and Gen. The tasks are *C2P*, *D2B*, *OD*, and *ToF*. We report the mean and standard deviation of the success rate (%) over 6 settings for each task, and the best results are highlighted in **bold**.

Task \	*C2P*			*D2B*			*OD*			*ToF*		
Algorithm	Train.	Gen.	All	Train.	Gen.	All	Train.	Gen.	All	Train.	Gen.	All
Stem-OB	89.0	**93.3**$_{\pm 6.1}$	**92.6**$_{\pm 5.7}$	78.0	**44.4**$_{\pm 22.2}$	**50.0**$_{\pm 24.1}$	100.0	**100.0**$_{\pm 0}$	**100.0**$_{\pm 0}$	100.0	**88.9**$_{\pm 19.2}$	**90.7**$_{\pm 17.8}$
Org	89.0	77.8$_{\pm 26.1}$	79.6$_{\pm 23.7}$	56.0	13.3$_{\pm 14.4}$	20.4$_{\pm 21.6}$	89.0	13.3$_{\pm 18.3}$	25.9$_{\pm 34.9}$	100.0	22.2$_{\pm 19.2}$	35.2$_{\pm 36.1}$
SRM	56.0	73.3$_{\pm 9.9}$	70.4$_{\pm 11.5}$	44.0	8.9$_{\pm 9.3}$	14.8$_{\pm 16.7}$	89.0	15.6$_{\pm 21.7}$	27.8$_{\pm 35.7}$	89.0	31.1$_{\pm 30.8}$	40.7$_{\pm 36.3}$
Mix	89.0	55.6$_{\pm 13.6}$	61.1$_{\pm 18.3}$	44.0	22.2$_{\pm 17.6}$	25.9$_{\pm 18.1}$	33.0	35.6$_{\pm 5.0}$	35.2$_{\pm 4.5}$	100.0	68.9$_{\pm 30.8}$	74.1$_{\pm 30.4}$
RO	56.0	53.3$_{\pm 9.3}$	53.7$_{\pm 8.4}$	44.0	15.6$_{\pm 12.7}$	20.4$_{\pm 16.4}$	89.0	88.9$_{\pm 7.9}$	88.9$_{\pm 7.0}$	78.0	82.2$_{\pm 20.2}$	81.5$_{\pm 18.1}$

Table 3: **Evaluations in simulated environments.** We compare the performance of *Stem-OB*, Org, SRM, and Mix in simulated environments. The left side of the table shows the results on the SAPIEN 3 environment, while the right side is about the MimicGen environment. The results are reported as the mean and standard deviation of the success rate(%) over their own various test settings. The performance of *Stem-OB* is highlighted in orange for better visibility. The best performance of each task is highlighted in **bold**.

Task \	SAPIEN 3				MimicGen			
Algorithm	*PushCube*	*PegInsertionSide*	*PickCube*	*StackCube*	*MugCleanup*	*Threading*	*ThreePieceAssembly*	*Coffee*
Stem-OB	**99.1**$_{\pm 1.8}$	0.4$_{\pm 0.8}$	**25.1**$_{\pm 20.9}$	**27.2**$_{\pm 31.3}$	19.5$_{\pm 12.5}$	16.4$_{\pm 12.8}$	13.1$_{\pm 6.9}$	**50.5**$_{\pm 19.6}$
Org	61.8$_{\pm 33.1}$	0.0$_{\pm 0.0}$	8.6$_{\pm 18.2}$	4.3$_{\pm 19.2}$	16.4$_{\pm 12.0}$	18.1$_{\pm 12.8}$	**14.8**$_{\pm 8.4}$	43.3$_{\pm 17.3}$
SRM	62.5$_{\pm 30.5}$	0.0$_{\pm 0.0}$	8.4$_{\pm 16.1}$	0.7$_{\pm 3.4}$	15.4$_{\pm 0.08}$	**25.8**$_{\pm 11.7}$	13.6$_{\pm 7.6}$	39.4$_{\pm 12.0}$
Mix	97.2$_{\pm 3.35}$	**0.6**$_{\pm 0.1}$	12.9$_{\pm 16.3}$	8.2$_{\pm 18.0}$	**22.3**$_{\pm 11.8}$	18.9$_{\pm 8.5}$	14.2$_{\pm 6.6}$	41.8$_{\pm 9.9}$

strating superior generalization capability. Under the training conditions, all methods achieve relatively high success rates, with our method performing slightly better than the baselines. However, in generalization testing, baseline methods exhibit significant performance drop, while our approach maintains a high success rate, showcasing the superior adaptability of *Stem-OB* to complex and noisy real-world visual disturbances.

The experiment results demonstrate that previous visual augmentation methods, such as SRM and Mix-Spectrum, struggle to generalize in real-world scenarios, which could be attribute to the complexity of real-world environments. Real images contain more redundant information, complicating the frequency domain and potentially leading to the failure of these augmentation methods. The light disturbance introduced in our experiments is a typical example, where wide-range but low-intensity noise is added to the images. Our approach effectively handles real-world noise by extracting high-level structures from the appearances, resulting in better generalization. Even in challenging scenarios like D2B, where baseline methods mostly fail, our method maintains a high success rate.

Interestingly, DP without any image modification (Org) outperforms the baselines in some tasks, such as OD and ToF, suggesting that DP has inherent generalization capabilities. However, this generalizability is inconsistent across tasks. For instance, in C2P, we observe that object appearance (cup and plate) had little impact on DP's performance, while the tablecloth pattern significantly affects it. In D2B, the duck's appearance is critical, whereas the table cloth variation is more influential. In contrast, our method exhibits consistent generalization across diverse scenarios.

6.4 SIMULATION EXPERIMENTS

Sapien 3 The evaluation results on Sapien 3 are presented on the left side of Tab. 3. The complexity of the tasks poses challenges for DP in generalizing across various conditions. However, our method achieves a higher success rate than the baseline methods on most of the tasks. In *PushCube*, both Mix-Spectrum and our method perform well, but our approach is more robust and reach nearly 100% success. These results demonstrate that in more photorealistic simulation environments, our method generalizes more effectively across diverse tasks and conditions.

MimicGen. The evaluation results on the MimicGen benchmark are presented on the right side of Tab. 3. At first glance, the results seem unanticipated, asresults seem unanticipated, as *Stem-OB* does not perform best in most settings. This can be attributed to the fact that is can be attributed to the fact that MimicGen evironments are less photorealistic, with nearly texturexture-free and -ow-resolutions, limiting DP from fully leveraging images. Consequently, diffusion inversion processing complicates the observations, limiting DP from fully leveraging the advantages of *Stem-OB*. It is im-

Table 4: **Ablation study on diffusion steps.** On the left side of the table, we increase the inversion step with fixed total number of steps, to intensify the effect of the diffusion process. Additionally, we compare the performance of the model with fixed ratio of steps, where the intensity of inversion is approximately the same. The results are reported as mean and standard deviation over 21 kinds of settings. The best performance is highlighted in orange and the second best is highlighted in pink.

Task \	Fixed Total Number of Steps (50)						Fixed Ratio of Steps (30%)		
Settings	0/50	5/50	10/50	15/50	20/50	25/50	9/30	15/50	30/100
PushCube	$61.8_{\pm33.1}$	$97.1_{\pm4.0}$	$91.1_{\pm18.9}$	$99.1_{\pm1.8}$	$98.5_{\pm3.2}$	$96.3_{\pm3.8}$	$98.1_{\pm3.8}$	$99.1_{\pm1.8}$	$98.3_{\pm2.9}$
PickCube	$8.6_{\pm18.2}$	$12.0_{\pm20.0}$	$28.0_{\pm26.3}$	$25.1_{\pm20.9}$	$12.7_{\pm14.0}$	$8.29_{\pm6.2}$	$23.7_{\pm24.4}$	$25.1_{\pm20.9}$	$25.6_{\pm22.3}$
StackCube	$4.3_{\pm19.2}$	$20.7_{\pm29.6}$	$27.2_{\pm31.3}$	$19.6_{\pm24.9}$	$11.6_{\pm12.8}$	$2.0_{\pm3.6}$	$19.5_{\pm20.5}$	$19.6_{\pm24.9}$	$20.2_{\pm27.0}$

portant to emphasize that our method is primarily designed for real-world scenarios, where diverse noise is inevitable, in contrast to the controlled controlled Gen environment.

6.5 ABLATION

In this section, we test with several design choices of *Stem-OB*, providing a better understanding of their impact on the final performance. We consider the following choices: the number of inversion steps with fixed total steps, the number of total steps with the constant proportion of inversion steps to the total steps, and the selection of inversion methods (DDPM or DDIM inversion). The experiments are conducted on SAPIEN 3. We choose *PushCube*, *PickCube* and *StackCube*, since they have a more significant performance variance in the main experiments.

Number of Inversion Steps. We compare the performance of *Stem-OB* with varying numbers of inversion steps, fixing the total steps to 50 and adjusting the inversion steps from 5 to 25 in increments of 5. The results, shown in Tab. 4, indicate that performance generally increases up to 15 inversion steps before declining. This can be attributed to insufficient removal of low-level appearance features with too few inversion steps, while excessive steps eliminate high-level structural information, hindering task performance. Optimal performance is observed around 10 to 15 inversion steps, depending on the complexity of the tasks.

Number of Total Steps. With the proportion of inversion steps to total steps fixed, we varied the total number of steps to 30, 50, and 100. The results, shown on the right side of Tab. 4, indicate that performance remains consistent regardless of the total number of steps. This suggests that the proportion of inversion steps is more critical to performance than the total number of steps.

Diffusion Inversion Methods. We compare the performance of two inversion methods: DDPM inversion and DDIM inversion, with results presented in Tab. 5. For DDIM inversion, we use 5/50 inversion steps instead of 15/50 due to differences in noise scheduling compared to DDPM inversion. The results show that DDPM inversion outperforms DDIM inversion on the benchmark datasets, supporting our choice of DDPM inversion as the primary diffusion inversion method.

Task	DDPM	DDIM
PushCube	$\mathbf{99.1}_{\pm\mathbf{1.8}}$	$81.3_{\pm15.7}$
PickCube	$\mathbf{25.1}_{\pm\mathbf{20.9}}$	$5.4_{\pm12.0}$

Table 5: **Ablation study on different inversion methods.** We compare the performance of DDPM and DDIM on the tasks of *PushCube* and *PickCube*. The results are reported as the success rate (%) of the task. The best performance is highlighted in **bold**.

7 CONCLUSION

In this work, we propose *Stem-OB*, a straightforward preprocessing method for visual IL. By inverting the observation in the diffusion latent space for certain steps, we effectively converge different observation variations to the node it stems from, Making it invariant to unseen low-level appearance changes in the observation space. Though we only test our method on the diffusion policy method, our method is general and compatible with any visual IL baselines in theory, and the plug-and-play nature of our method makes it easy to integrate. We plan to test our method with other visual IL baselines in simulation and real tasks in the future.

Reproducibility: The main algorithm of our method is simple as applying the open-sourced DDPM inversion method on the dataset before training. We've provided the code for our method in the supplementary material.

REFERENCES

Manuel Brack, Felix Friedrich, Katharia Kornmeier, Linoy Tsaban, Patrick Schramowski, Kristian Kersting, and Apolinário Passos. Ledits++: Limitless image editing using text-to-image models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8861–8870, 2024.

Edoardo Cetin and Oya Celiktutan. Domain-robust visual imitation learning with mutual information constraints, 2021. URL `https://arxiv.org/abs/2103.05079`.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric A. Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *ArXiv*, abs/2303.04137, 2023. URL `https://api.semanticscholar.org/CorpusID:257378658`.

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.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024.

Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. *arXiv preprint arXiv:2208.01618*, 2022.

Jialu Gao, Kaizhe Hu, Guowei Xu, and Huazhe Xu. Can pre-trained text-to-image models generate visual goals for reinforcement learning? *Advances in Neural Information Processing Systems*, 36, 2024.

Daniel Garibi, Or Patashnik, Andrey Voynov, Hadar Averbuch-Elor, and Daniel Cohen-Or. Renoise: Real image inversion through iterative noising, 2024. URL `https://arxiv.org/abs/2403.14602`.

Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiaing Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, Xiaodi Yuan, Pengwei Xie, Zhiao Huang, Rui Chen, and Hao Su. Maniskill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations*, 2023.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning, 2016. URL `https://arxiv.org/abs/1606.03476`.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arxiv:2006.11239*, 2020.

Shaul Hochstein and Merav Ahissar. View from the top: Hierarchies and reverse hierarchies in the visual system. *Neuron*, 36(5):791–804, 2002. ISSN 0896-6273. doi: https://doi.org/10.1016/S0896-6273(02)01091-7. URL `https://www.sciencedirect.com/science/article/pii/S0896627302010917`.

Yangru Huang, Peixi Peng, Yifan Zhao, Guangyao Chen, and Yonghong Tian. Spectrum random masking for generalization in image-based reinforcement learning. *Advances in Neural Information Processing Systems*, 35:20393–20406, 2022.

Ziqi Huang, Tianxing Wu, Yuming Jiang, Kelvin CK Chan, and Ziwei Liu. Reversion: Diffusion-based relation inversion from images. *arXiv preprint arXiv:2303.13495*, 2023.

Inbar Huberman-Spiegelglas, Vladimir Kulikov, and Tomer Michaeli. An edit friendly ddpm noise space: Inversion and manipulations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12469–12478, 2024.

Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6007–6017, 2023.

Jeong Woon Lee and Hyoseok Hwang. Mix-spectrum for generalization in visual reinforcement learning. *Available at SSRN 4818827*.

Jeong Woon Lee and Hyoseok Hwang. Mix-spectrum for generalization in visual reinforcement learning. *Available at SSRN 4818827*, 2024.

Siyuan Li, Xun Wang, Rongchang Zuo, Kewu Sun, Lingfei Cui, Jishiyu Ding, Peng Liu, and Zhe Ma. Robust visual imitation learning with inverse dynamics representations, 2023. URL `https://arxiv.org/abs/2310.14274`.

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. In *7th Annual Conference on Robot Learning*, 2023.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021.

Daiki Miyake, Akihiro Iohara, Yu Saito, and Toshiyuki Tanaka. Negative-prompt inversion: Fast image inversion for editing with text-guided diffusion models, 2023. URL `https://arxiv.org/abs/2305.16807`.

Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6038–6047, 2023.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation, 2022. URL `https://arxiv.org/abs/2203.12601`.

Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, 2000. URL `https://api.semanticscholar.org/CorpusID:266239314`.

Zhihong Pan, Riccardo Gherardi, Xiufeng Xie, and Stephen Huang. Effective real image editing with accelerated iterative diffusion inversion, 2023. URL `https://arxiv.org/abs/2309.04907`.

Giuseppe Paolo, Jonas Gonzalez-Billandon, and Balázs Kégl. A call for embodied ai. *arXiv preprint arXiv:2402.03824*, 2024.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, June 2022.

Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning, 2011. URL `https://arxiv.org/abs/1011.0686`.

Dvir Samuel, Barak Meiri, Nir Darshan, Shai Avidan, Gal Chechik, and Rami Ben-Ari. Regularized newton raphson inversion for text-to-image diffusion models, 2024. URL `https://arxiv.org/abs/2312.12540`.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. URL `https://arxiv.org/abs/2010.02502`.

Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent correspondence from image diffusion. *Advances in Neural Information Processing Systems*, 36: 1363–1389, 2023.

Stone Tao, Fanbo Xiang, Arth Shukla, Chen Bao, Nan Xiao, Rui Chen, Tongzhou Mu, Tse-Kai Chan, Xander Hinrichsen, Xiaodi Yuan, Xinsong Lin, Xuanlin Li, Yuan Gao, Yuzhe Qin, Zhiao Huang, and Hao Su. Maniskill 3 (beta). 2024. GitHub repository.

Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1921–1930, 2023.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Shenghua Wan, Yucen Wang, Minghao Shao, Ruying Chen, and De-Chuan Zhan. Semail: Eliminating distractors in visual imitation via separated models, 2023. URL https://arxiv.org/abs/2306.10695.

Chen Wang, Linxi Fan, Jiankai Sun, Ruohan Zhang, Li Fei-Fei, Danfei Xu, Yuke Zhu, and Anima Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play. *arXiv preprint arXiv:2302.12422*, 2023a.

Yanghao Wang and Long Chen. Improving diffusion-based data augmentation with inversion spherical interpolation. *arXiv preprint arXiv:2408.16266*, 2024.

Yunke Wang, Minjing Dong, Bo Du, and Chang Xu. Imitation learning from purified demonstration. *arXiv preprint arXiv:2310.07143*, 2023b.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Annie Xie, Lisa Lee, Ted Xiao, and Chelsea Finn. Decomposing the generalization gap in imitation learning for visual robotic manipulation. *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3153–3160, 2023. URL https://api.semanticscholar.org/CorpusID:259375845.

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.

Zhecheng Yuan, Tianming Wei, Shuiqi Cheng, Gu Zhang, Yuanpei Chen, and Huazhe Xu. Learning to manipulate anywhere: A visual generalizable framework for reinforcement learning, 2024a. URL https://arxiv.org/abs/2407.15815.

Zhecheng Yuan, Sizhe Yang, Pu Hua, Can Chang, Kaizhe Hu, and Huazhe Xu. Rl-vigen: A reinforcement learning benchmark for visual generalization. *Advances in Neural Information Processing Systems*, 36, 2024b.

Zhongqi Yue, Jiankun Wang, Qianru Sun, Lei Ji, Eric I Chang, Hanwang Zhang, et al. Exploring diffusion time-steps for unsupervised representation learning. *arXiv preprint arXiv:2401.11430*, 2024.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

Haoran Zhang, Yinghong Tian, Liang Yuan, and Yue Lu. Invariant adversarial imitation learning from visual inputs. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2023a. doi: 10.1109/ICASSP49357.2023.10096070.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization, 2018. URL https://arxiv.org/abs/1710.09412.

Junyi Zhang, Charles Herrmann, Junhwa Hur, Luisa Polania Cabrera, Varun Jampani, Deqing Sun, and Ming-Hsuan Yang. A tale of two features: Stable diffusion complements dino for zero-shot semantic correspondence. 2023b.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL https://arxiv.org/abs/2304.13705.

Ray Chen Zheng, Kaizhe Hu, Zhecheng Yuan, Boyuan Chen, and Huazhe Xu. Extraneousness-aware imitation learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2973–2979, 2023. doi: 10.1109/ICRA48891.2023.10161521.

A APPENDIX

A.1 DERIVATION OF DDPM INVERSION ATTRIBUTE LOSS

The derivation of DDPM inversion is straightforward by simply letting $\sigma = \sqrt{1 - \bar{\alpha}_t}$ in Eq. (7).

A.2 DERIVATION OF DDIM INVERSION ATTRIBUTE LOSS

We first rewrite Eq. (5) as a linear combination of $\boldsymbol{x}_0$ and a noise variable. To do so, we expand the recursive equation in Eq. (5). Note that we assume that $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}, t, \mathcal{C})$ is a Gaussian distribution with mean $\boldsymbol{\mu}_t$ and use $\boldsymbol{\epsilon}_t$ for simplicity in the following derivation.

$$
\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}} \boldsymbol{x}_{t-1} + \left(\sqrt{\frac{1}{\bar{\alpha}_t} - 1} - \sqrt{\frac{1}{\bar{\alpha}_{t-1}} - 1} \right) \boldsymbol{\epsilon}_t \\
&= \sqrt{\frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}} \left(\sqrt{\frac{\bar{\alpha}_{t-1}}{\bar{\alpha}_{t-2}}} \boldsymbol{x}_{t-2} + \left(\sqrt{\frac{1}{\bar{\alpha}_{t-1}} - 1} - \sqrt{\frac{1}{\bar{\alpha}_{t-2}} - 1} \right) \boldsymbol{\epsilon}_{t-1} \right) + \left(\sqrt{\frac{1}{\bar{\alpha}_t} - 1} - \sqrt{\frac{1}{\bar{\alpha}_{t-1}} - 1} \right) \boldsymbol{\epsilon}_t \\
&= \sqrt{\frac{\bar{\alpha}_t}{\bar{\alpha}_{t-2}}} \left(\sqrt{\frac{\bar{\alpha}_{t-2}}{\bar{\alpha}_{t-3}}} \boldsymbol{x}_{t-3} + \left(\sqrt{\frac{1}{\bar{\alpha}_{t-2}} - 1} - \sqrt{\frac{1}{\bar{\alpha}_{t-3}} - 1} \right) \boldsymbol{\epsilon}_{t-2} \right) \\
&\qquad + \sqrt{\frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}} \left(\sqrt{\frac{1}{\bar{\alpha}_{t-1}} - 1} - \sqrt{\frac{1}{\bar{\alpha}_{t-2}} - 1} \right) \boldsymbol{\epsilon}_{t-1} + \left(\sqrt{\frac{1}{\bar{\alpha}_t} - 1} - \sqrt{\frac{1}{\bar{\alpha}_{t-1}} - 1} \right) \boldsymbol{\epsilon}_t \\
&\quad \dots \\
&= \sqrt{\bar{\alpha}_t} \boldsymbol{x}_0 + \sum_{i=1}^{t} \sqrt{\frac{\bar{\alpha}_t}{\bar{\alpha}_i}} \left(\sqrt{\frac{1}{\bar{\alpha}_i} - 1} - \sqrt{\frac{1}{\bar{\alpha}_{i-1}} - 1} \right) \boldsymbol{\epsilon}_i \\
&= \sqrt{\bar{\alpha}_t} \boldsymbol{x}_0 + \sum_{i=1}^{t} \sqrt{\frac{\bar{\alpha}_t}{\bar{\alpha}_i}} \left(\sqrt{\frac{1}{\bar{\alpha}_i} - 1} - \sqrt{\frac{1}{\bar{\alpha}_{i-1}} - 1} \right) \boldsymbol{\mu_t} + \sqrt{\sum_{i=1}^{t} \frac{\bar{\alpha}_t}{\bar{\alpha}_i} \left(\sqrt{\frac{1}{\bar{\alpha}_i} - 1} - \sqrt{\frac{1}{\bar{\alpha}_{i-1}} - 1} \right)^2} \boldsymbol{\epsilon}
\end{aligned}
\tag{12}
$$

where the last equality holds considering the property of sum of gaussian distributions, i.e., $\mathcal{N}(\boldsymbol{\mu}_1, \sigma_1^2 \boldsymbol{I}) + \mathcal{N}(\boldsymbol{\mu}_2, \sigma_2^2 \boldsymbol{I}) \sim \mathcal{N}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2, (\sigma_1^2 + \sigma_2^2) \boldsymbol{I})$. And $\sum_{i=1}^{t} \sqrt{\frac{\bar{\alpha}_t}{\bar{\alpha}_i}} \left(\sqrt{\frac{1}{\bar{\alpha}_i} - 1} - \sqrt{\frac{1}{\bar{\alpha}_{i-1}} - 1} \right) \boldsymbol{\mu_t}$ represents the mean shift resulting from the mean bias of model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$. For two similar images, the biases are approximately equal and thus cancel out when substituted into Eq. (7). Consequently, we can approximately derive the attribute loss for DDIM inversion.

$$
\begin{aligned}
loss_{DDIM}(\boldsymbol{x}_0, \boldsymbol{y}_0, t) &\approx \frac{1}{2} \left[1 - \text{erf}\left(\frac{\|\sqrt{\bar{\alpha}_t}(\boldsymbol{y}_0 - \boldsymbol{x}_0)\|}{2\sqrt{2\sum_{i=1}^{t} \frac{\bar{\alpha}_t}{\bar{\alpha}_i} \left(\sqrt{\frac{1}{\bar{\alpha}_i} - 1} - \sqrt{\frac{1}{\bar{\alpha}_{i-1}} - 1} \right)^2}} \right) \right] \\
&= \frac{1}{2} \left[1 - \text{erf}\left(\frac{\|(\boldsymbol{y}_0 - \boldsymbol{x}_0)\|}{2\sqrt{2\sum_{i=1}^{t} \frac{1}{\bar{\alpha}_i} \left(\sqrt{\frac{1}{\bar{\alpha}_i} - 1} - \sqrt{\frac{1}{\bar{\alpha}_{i-1}} - 1} \right)^2}} \right) \right]
\end{aligned}
\tag{13}
$$

B TRAINING DETAILS

The hyperparameters for Diffusion Policy across all experiments are listed in Tab. 6. We use DDPM inversion in all the basic experiments, with the specific inversion steps for each experiment detailed in Tab. 7.

Hyperparameter	value	Hyperparameter	value
epoch_every_n_steps	100	ddim_num_train_timesteps	100
seq_length	15	ddim_num_inference_timesteps	10
frame_stack	2	ddim_beta_schedule	squaredcos_cap_v2
batch_size	128	ddim_clip_sample	TRUE
num_epochs	1500	ddim_set_alpha_to_one	TRUE
learning_rate_initial	0.0001	ddim_steps_offset	0
learning_rate_decay_factor	0.1	ddim_prediction_type	epsilon
regularization	L2/0.0	VisualCore_feature_dimension	64
observation_horizon	2	VisualCore_backbone_class	ResNet18Conv
action_horizon	8	VisualCore_pool_class	SpatialSoftmax
prediction_horizon	16	VisualCore_pool_num_kp	32
unet_diffusion_embed_dim	256	VisualCore_pool_temperature	FALSE
unet_down_dims	[256, 512,1024]	VisualCore_pool_noise_std	1
unet_kernel_size	4	obs_randomizer_class	CropRandomizer
unet_n_groups	8	num_crops	1
ema_power	0.75		

Table 6: **Hyperparameters used for Diffusion Policy.** We use the same hyperparameters of diffusion policy for all the experiments.

Task	Step	Task	Step	Task	Step
Cup2Plate	15/50	PushCube	15/50	MugCleanup	10/50
Duck2Bowl	15/50	PegInsertionSide	15/50	Threading	5/50
OpeningDrawer	15/50	PickCube	15/50	ThreePieceAssembly	5/50
Turn on Faucet	15/50	StackCube	10/50	Coffee	5/50

Table 7: **Tasks and inversion steps used for training the Diffusion Policy.** We all use DDPM as the inversion method.

C EXPERIMENT DETAILS

C.1 REALWORLD DETAILS

Fig. 6 shows the camera views of the real-world environments. Each task is represented by two rows, corresponding to different camera angles. The columns represent task settings, with the first being the default. As lighting conditions vary across tasks, images are grouped in pairs: the odd columns are under warm light and the even columns are under cold light. The low-level object appearance changes along each row.

C.2 SAPIEN ENVIRONMENTS SETTINGS

Each task includes 21 distinct testing environments, as shown in Fig. 7. We alternate between seven unique tablecloth textures and three lighting conditions: white, yellow, and red light.

C.3 MIMICGEN ENVIRONMENTS SETTINGS

Fig. 9 demonstrates the four MimicGen environments used in the experiments, and each row in Fig. 8 represents a task. In *MugCleanup*, the robot arm opens a drawer, retrieves a mug, and places it back. *Threading* involves the precise insertion of a stick into a hole. *ThreePieceAssembly* requires collecting and assembling three components in a specific order. In *Coffee*, the robot opens the lid of the coffee machine and places a cup inside. Eight different table textures are used across all tasks (first eight columns). Additionally, *MugCleanup*, *Threading*, and *ThreePieceAssembly* employ alternating object texture, as indicated in the last column, creating distinct testing environments.
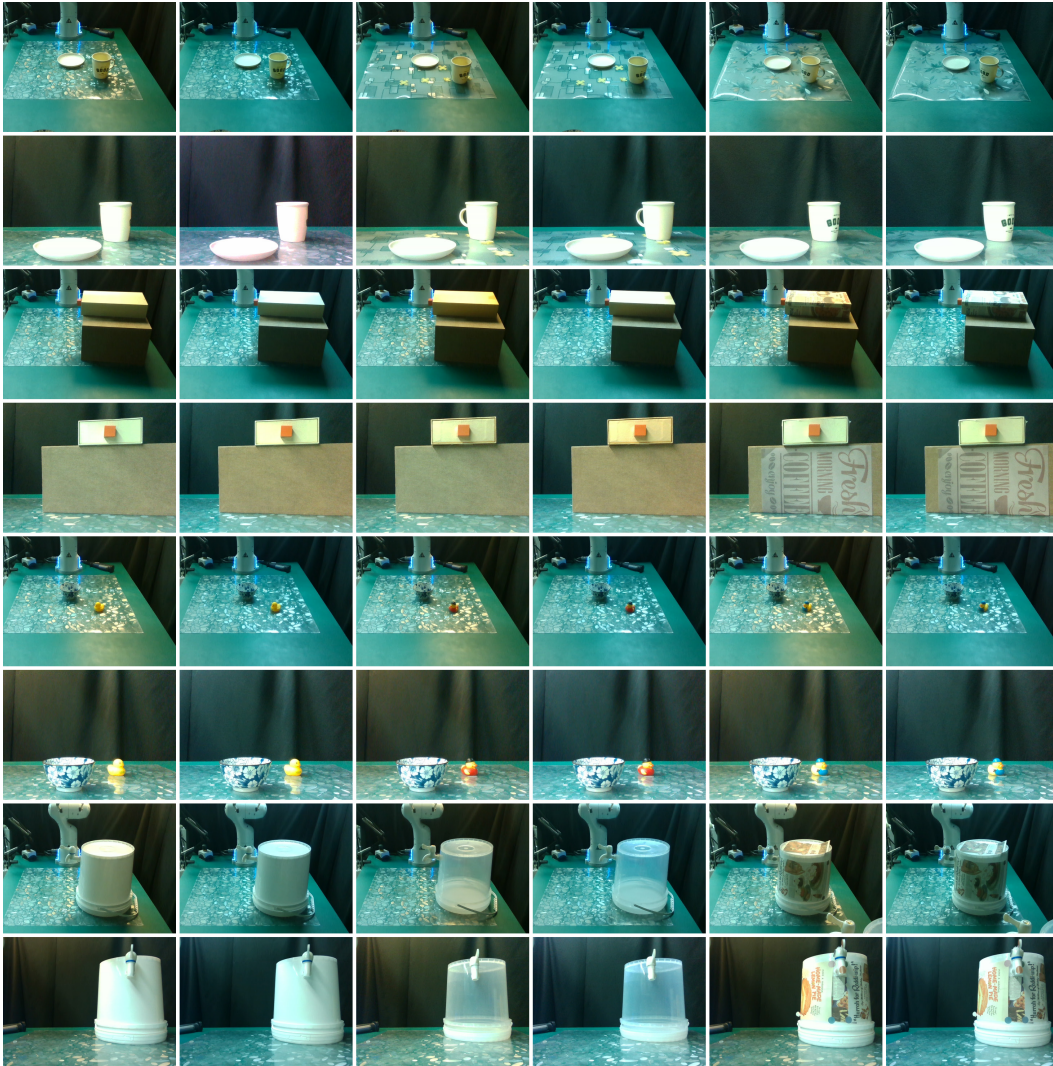
Figure 6: **Realworld Camera Views.** Each task is represented by two rows, with each representing a different camera view angle. For each task, we perform experiments on three instances with different object appearances, and two light conditions, as displayed along the columns. The first column represents the training setup.

Figure 7: **Sapien Testing Settings.** Each task consists of 21 distinct settings, arranged in two rows. The images are grouped in threes, with each group sharing the same tablecloth texture but differing in lighting conditions: white, yellow, and red light. A total of seven unique tablecloth textures are used. The training environment is the first setting in the first row.
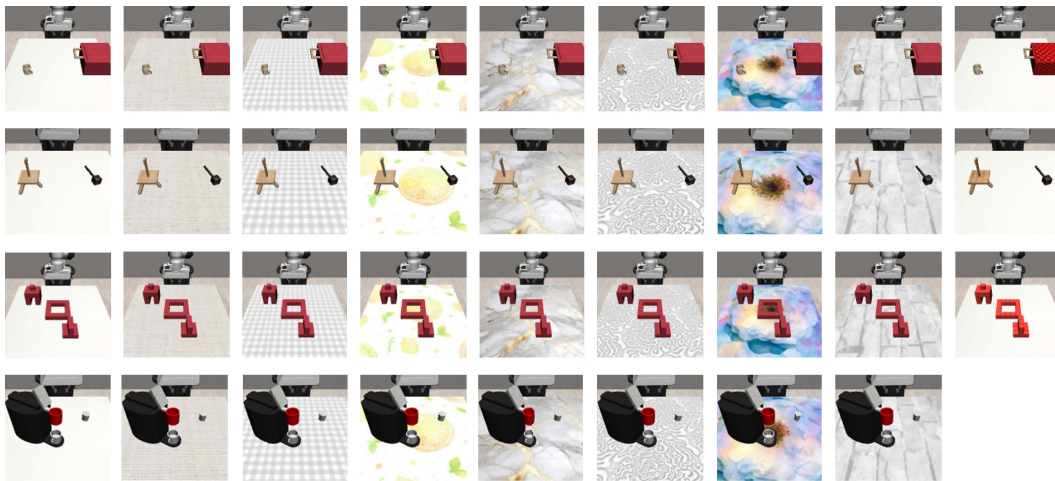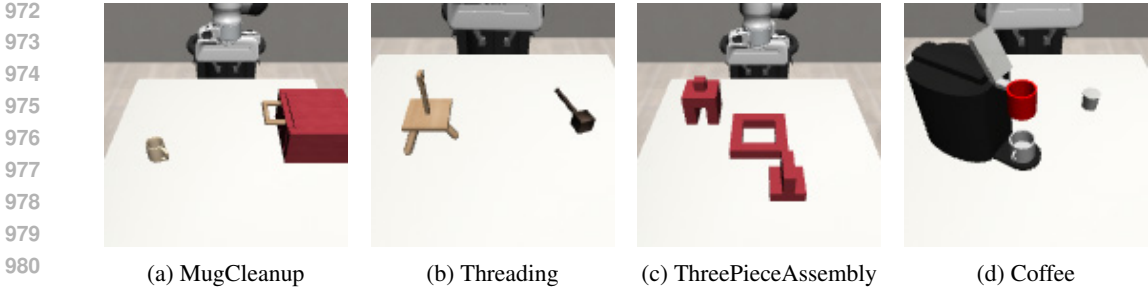

Figure 8: **MimicGen Testing Settings.** Each row represents a specific task, with eight different table textures across the columns. The training environment is the first column. Additionally, *Mug-Cleanup*, *Threading*, and *ThreePieceAssembly* employ alternate object textures in the last column.

| (a) MugCleanup | (b) Threading | (c) ThreePieceAssembly | (d) Coffee |

Figure 9: **MimicGen environments.** These tasks are based on MimicGen benchmark. (a) The agent must open the drawer, pick up the mug, and place it into the drawer. (b) This task requires the agent to thread a string through a hole. (c) The agent must assemble three pieces together. (d) The agent is required to open the lid, and then place the coffee cup inside it.

C.4 DETAILS OF EXPERIMENT RESULTS
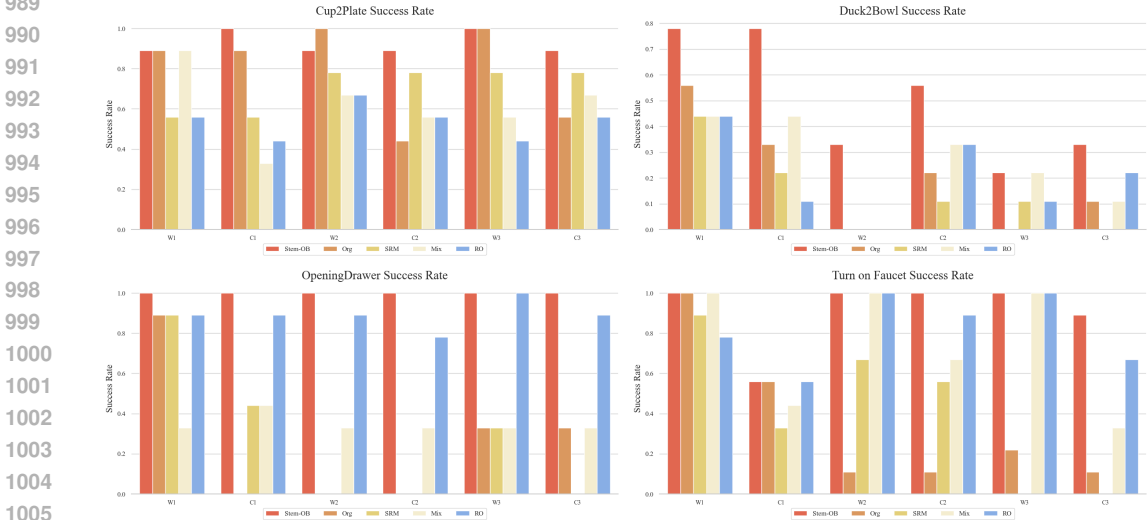
Figure 10: **Realworld Success Rate.** This figure presents the success rates for real-world tasks. The groups are labeled as W1, C1 through W3, C3. W1 to W3 are conducted under the same warm lighting condition as the training set with different object appearances. C1 to C3 are performed under identical cold lighting conditions, with varying object appearances. The order of the groups is consistent to that shown in Fig. 6.

Fig. 10, Fig. 11, and Fig. 12 illustrate the success rates of *Stem-OB* and baseline methods in real-world, SAPIEN 3, and MimicGen environments, respectively. These three figures display the performance of *Stem-OB* under each setting with more details.

C.5 DETAILS OF ILLUSTRATIVE EXPERIMENTS

Image Distance Calculation: We use the 2-norm to compute the image distance $D(\boldsymbol{x}, \boldsymbol{y})$ in the latent space, where a smaller distance indicates greater similarity between the two images.

Intra-Category Distance Calculation: the intra-category distance is calculated as the mean of the pairwise distances between images within the same category $\mathcal{I}$

$$D_{intra} = \frac{2}{N(N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} D(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{14}$$

where N is the number of images within $\mathcal{I}$ and $\boldsymbol{x}_i$ is the i-th image.
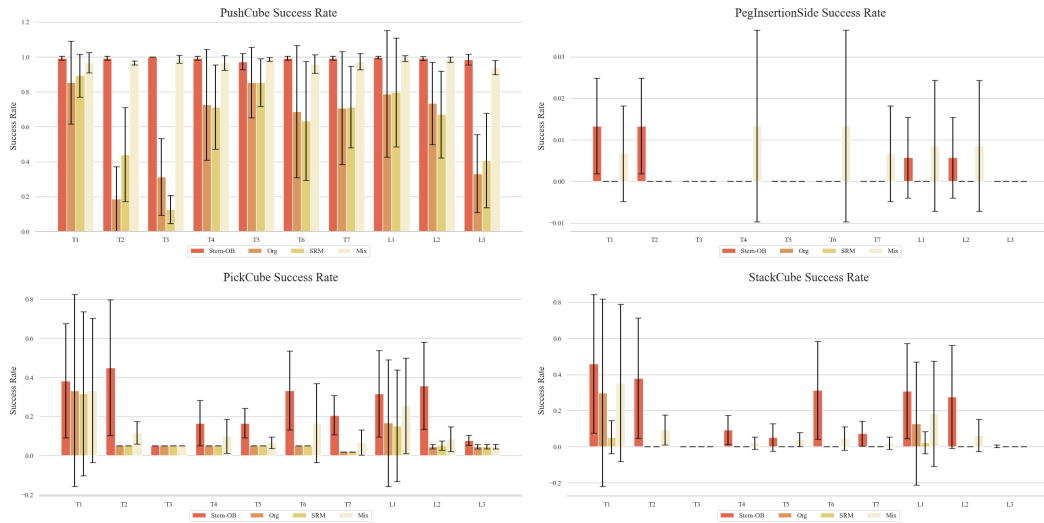
Figure 11: **Sapien Success Rate.** This figure presents the success rates for tasks of the SAPIEN 3 benchmark. The groups are labeled as T1 to T7 and L1 to L3. The settings are varied across 7 different tablecloth textures (T1 to T7) and 3 distinct lighting conditions (L1 to L3). For T1 to T7, the mean and standard deviation of the success rates are calculated over different lighting conditions with 3 lighting conditions combined for one texture. Conversely, for L1 to L3, the mean and standard deviation are computed over various tablecloth textures under a fixed lighting condition, with 7 textures evaluated for each light condition. The order of tablecloth textures and lighting conditions aligns with that presented in Fig. 7.
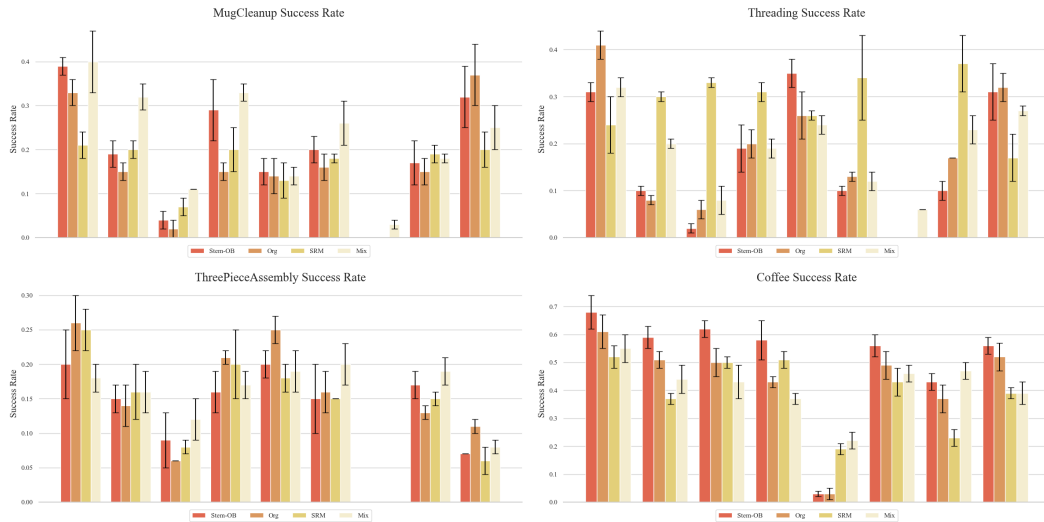

Figure 12: **MimicGen Success Rate.** This figure presents the success rates of *Stem-OB* and baselines across four distinct tasks. The mean and standard deviation are computed over 300 episodes. Each group of bars corresponds to one experimental setting. The bar order is consistent with the arrangement shown in Fig. 8.
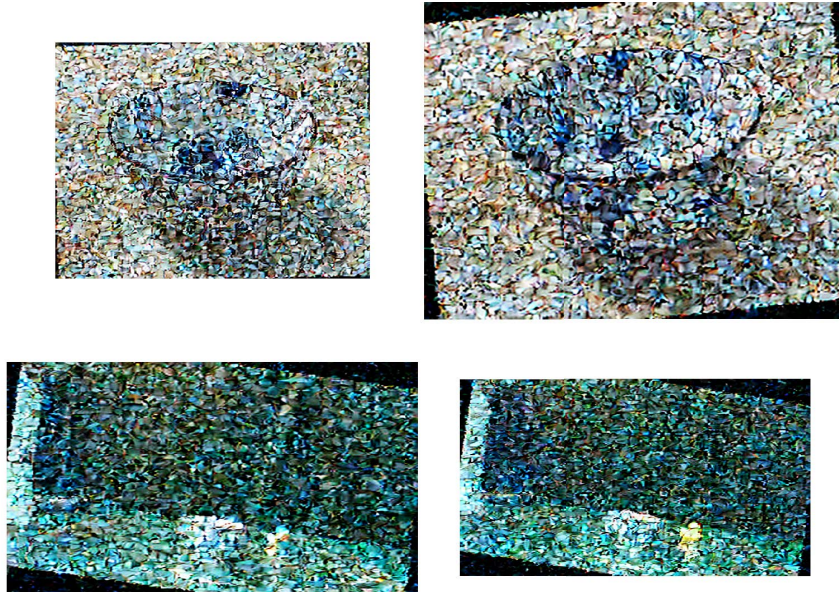

Figure 13: **Illustrative Experiment Objects.**

Cross-Category Distance Calculation: the cross-category distance is calculated as the mean of the pairwise distances between images from two different categories $\mathcal{I}_1$ and $\mathcal{I}_2$

$$D_{intra} = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} D(\boldsymbol{x}_i, \boldsymbol{y}_j) \qquad (15)$$

where M and N are the size of $\mathcal{I}_0$ and $\mathcal{I}_1$, respectively, $\boldsymbol{x}_i \in \mathcal{I}$ and $\boldsymbol{y}_i \in \mathcal{I}_2$.

C.6   DETAILS OF USER STUDY

Figure 14: **Example of User Study Questions.** The upper two images represent one example from the first section, and the lower two images represent one example from the second section. Each pair is generated through a randomly selected inversion step. To prevent users from making judgments based on image size or orientation, these images have been randomly rotated, cropped, and zoomed.

We conducted a user study to evaluate the quality of the inversion results through a questionnaire-based approach. In the first section, consisting of 42 questions, participants compare two images per question and are asked to determine whether the images depict the same object, different objects of the same category, or entirely different objects. Participants are informed that the images have been randomly rotated, cropped, and zoomed, so image size or orientation should not influence their judgment of object similarity. All images are snapshots of items like bowls and faucets that are used in the real-world experiment, with each pair inverted by a randomly selected inversion step.

The second section, consisting of 56 questions, presented images from real-world tasks, as seen in Fig. 6. In this section, participants evaluate whether the images represent the same task, different scenes from the same task (with variations such as lighting or object appearance), or entirely different tasks, where the target of the task changed (for example, a cup versus a drawer). Participants are informed that scene variations could include changes in environmental conditions, such as lighting, or the appearance of objects, like a change in a tablecloth or the texture of a drawer, but the underlying task or function remains the same. To avoid bias from image size or angle, we also randomly cropped, rotated, and zoomed the images. The example of images from the two sections is shown in Fig. 14. A total of 51 valid questionnaires were collected, with the results presented in Fig. 3.
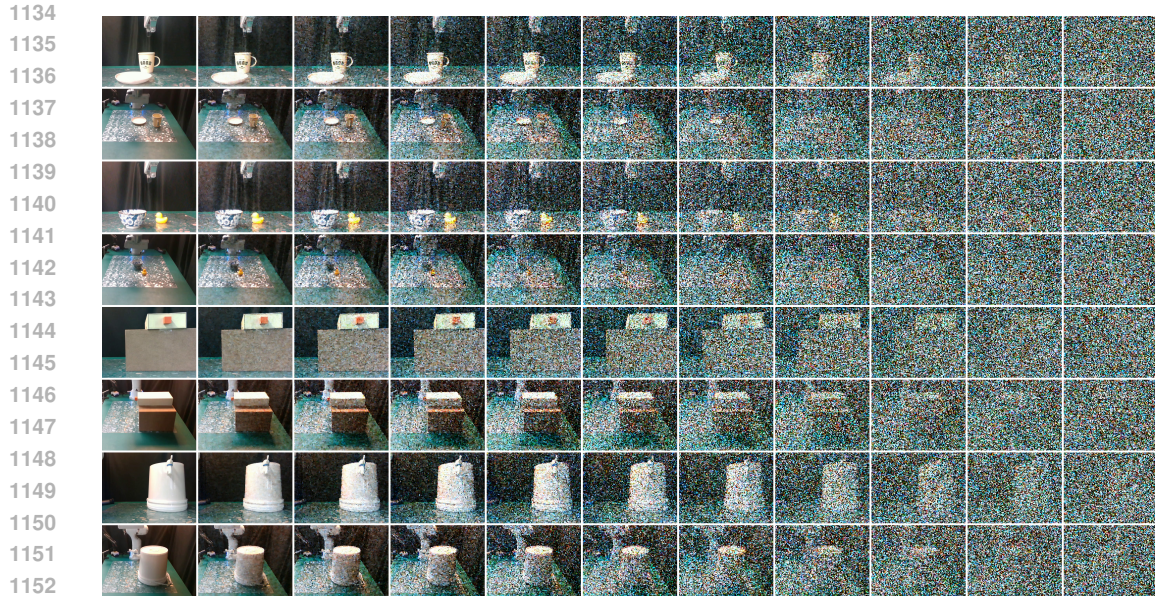
Figure 15: **Realworld Inversion Results.** Each task corresponds to two rows, with each representing a distinct camera view. The first column of each row is the original image, while the other columns are the DDPM inversion results. Starting from the second column, the diffusion steps are increased from 5 to 50, with the incremental step being 5.
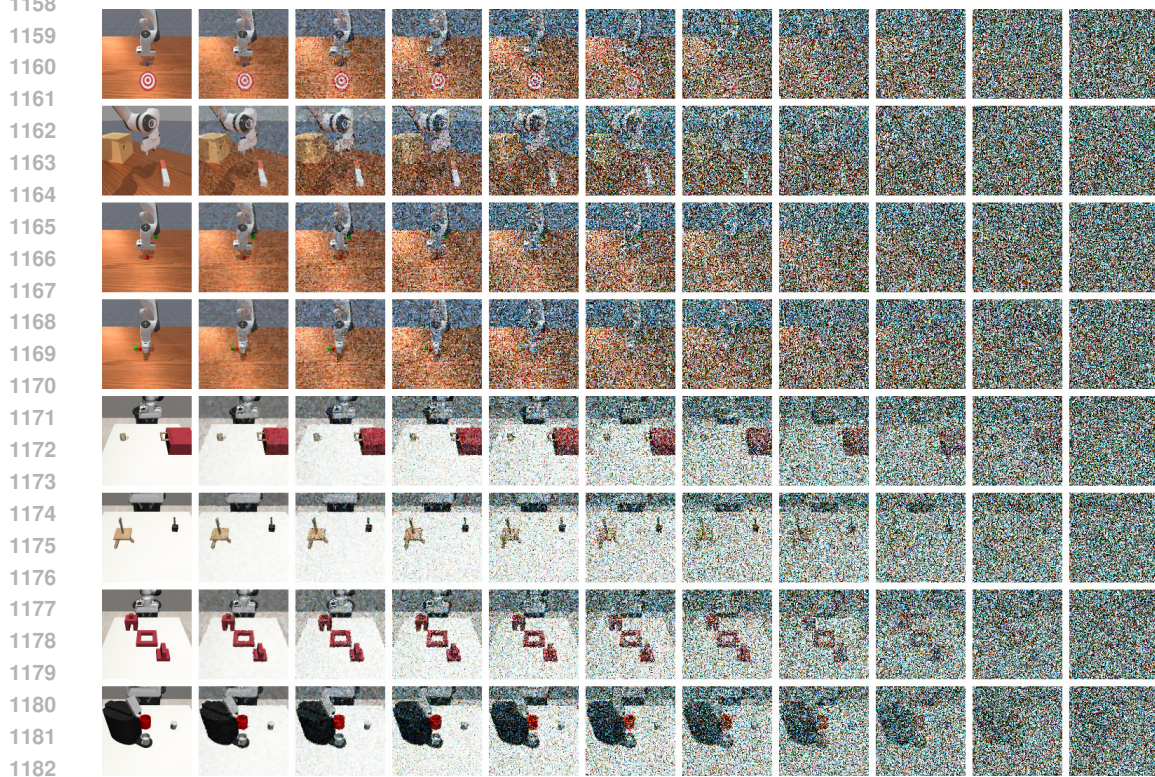
Figure 16: **Simulation Inversion Results.** The first four rows correspond to the tasks in SAPIEN 3, while the last four rows correspond to the tasks in MimicGen. Each row represents a task, with the first column displaying the original image and the subsequent columns illustrating the DDPM inversion results. Starting from the second column, the diffusion steps are increased from 5 to 50, with the incremental step being 5.

Table 8: **Success rate of SAPIEN tasks.** Train.: evaluations in the same settings as the training dataset. Gen.: evaluations under different visual perturbations for generalizability analysis. All: evaluations including both Train. and Gen. The tasks are *PushCube*, *PegInsertionSide*, *PickCube*, and *StackCube*. We report the mean and standard deviation of the success rate (%), and the best results are highlighted in **bold**. In the following tables, when "Train.", "Gen.", or "All" is mentioned, they carry the same meaning as defined above.

Task \	PushCube			PegInsertionSide			PickCube			StackCube		
Algorithm	Train.	Gen.	All	Train.	Gen.	All	Train.	Gen.	All	Train.	Gen.	All
Stem-OB	**100.0**	**99.1**$_{\pm1.8}$	**99.1**$_{\pm1.8}$	**2.0**	0.3$_{\pm0.7}$	0.4$_{\pm0.8}$	50.0	**23.9**$_{\pm20.6}$	**25.1**$_{\pm20.9}$	**90.0**	**24.1**$_{\pm28.7}$	**27.2**$_{\pm31.3}$
Org	**100.0**	59.9$_{\pm32.7}$	61.8$_{\pm33.1}$	0.0	0.0$_{\pm0.0}$	0.0$_{\pm0.0}$	**90.0**	4.6$_{\pm1.1}$	8.6$_{\pm18.2}$	**90.0**	0.0$_{\pm0.0}$	4.3$_{\pm19.2}$
SRM	**100.0**	60.6$_{\pm30.0}$	62.5$_{\pm30.5}$	0.0	0.0$_{\pm0.0}$	0.0$_{\pm0.0}$	80.0	4.8$_{\pm1.6}$	8.4$_{\pm16.1}$	16.0	0.0$_{\pm0.0}$	0.7$_{\pm3.4}$
Mix	**100.0**	97.1$_{\pm3.4}$	97.2$_{\pm3.4}$	**2.0**	**0.5**$_{\pm1.2}$	**0.6**$_{\pm0.1}$	75.0	9.8$_{\pm8.8}$	12.9$_{\pm16.3}$	84.0	4.4$_{\pm6.3}$	8.2$_{\pm18.0}$

Table 9: **Success rate of MimicGen tasks.** The tasks are *MugCleanup*, *Threading*, *ThreePieceAssembly*, and *Coffee*. We report the mean and standard deviation of the success rate (%), and the best results are highlighted in **bold**.

Task \	MugCleanup			Threading			ThreePieceAssembly			Coffee		
Algorithm	Train.	Gen.	All	Train.	Gen.	All	Train.	Gen.	All	Train.	Gen.	All
Stem-OB	38.7$_{\pm2.5}$	17.1$_{\pm11.1}$	19.5$_{\pm12.5}$	31.3$_{\pm2.5}$	14.5$_{\pm12.3}$	16.4$_{\pm12.8}$	19.7$_{\pm4.8}$	12.3$_{\pm6.7}$	13.1$_{\pm6.9}$	**68.0**$_{\pm5.7}$	**48.0**$_{\pm19.7}$	**50.5**$_{\pm19.6}$
Org	33.0$_{\pm2.8}$	16.4$_{\pm12.0}$	16.4$_{\pm12.0}$	**41.3**$_{\pm3.3}$	15.2$_{\pm10.4}$	18.1$_{\pm12.8}$	**26.3**$_{\pm4.5}$	13.4$_{\pm7.7}$	**14.8**$_{\pm8.4}$	61.0$_{\pm6.5}$	40.7$_{\pm16.8}$	43.3$_{\pm17.3}$
SRM	21.0$_{\pm2.9}$	15.6$_{\pm7.1}$	15.4$_{\pm0.1}$	24.3$_{\pm5.7}$	**26.0**$_{\pm12.2}$	**25.8**$_{\pm11.7}$	24.7$_{\pm2.9}$	12.3$_{\pm6.8}$	13.6$_{\pm7.6}$	52.3$_{\pm3.7}$	37.6$_{\pm11.7}$	39.4$_{\pm12.0}$
Mix	**39.7**$_{\pm6.9}$	**21.9**$_{\pm12.3}$	**22.3**$_{\pm11.8}$	32.0$_{\pm1.6}$	17.3$_{\pm7.5}$	18.9$_{\pm8.5}$	17.7$_{\pm1.7}$	**13.8**$_{\pm6.8}$	14.2$_{\pm6.6}$	55.3$_{\pm5.3}$	39.8$_{\pm8.8}$	41.8$_{\pm9.9}$

D VISUALIZATION

D.1 INVERSION VISUALIZATION

Fig. 15 and Fig. 16 display the inversion results for real-world and simulation environments, respectively. In the real-world environments, each task spans two rows, corresponding to the same camera view as in Fig. 6. In the simulation environments, each row represents a task, with the first four rows showing SAPIEN 3 environments and the last four showing MimicGen environments. The first column in each row is the original image, while subsequent columns present the inverted results. From the second column onward, the inversion results are generated by DDPM inversion, with 50 total inversion steps and 5 incremental steps from 5 to 50.

E DETAILS OF SUCCESS RATES

We separate the results for training and test variants to provide a clearer picture of the policy's performance. For SAPIEN tasks, we test 50 episodes for each task, while for MimicGen tasks, we test 300 episodes. As is shown in the Tab. 8 and Tab. 9, *Stem-OB* performs well in the generalizing settings, while remaining competitive in the training settings.
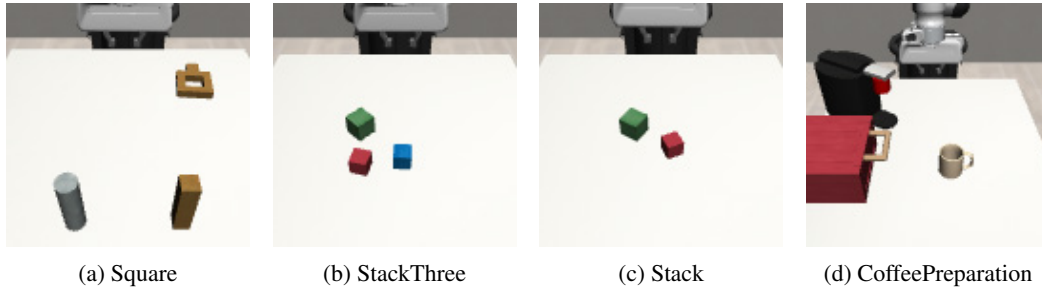
F ADDITIONAL MIMICGEN EXPERIMENTS

To further strengthen our evaluation, we add experiments for additional MimicGen tasks. We report results for an additional 4 tasks, including *Square*, *StackThree*, *Stack*, and *CoffeePreparation*. Fig. 17 shows the images for each task. In *Square*, the robot arm must pick up square nuts and place them onto the correct pole. In *StackThree*, the robot arm is asked to stack three cubes in the correct order. *Stack* is similar to *StackThree*, but with only two cubes. *CoffeePreparation* is a long-horizon task where the robot arm must prepare a cup of coffee by placing a mug under the coffee

Table 10: **Success rate of Additional MimicGen tasks.** The tasks are *Square*, *StackThree*, *Stack*, and *CoffeePreparation*. We report the mean and standard deviation of the success rate (%), and the best results are highlighted in **bold**.

Task \	Square			StackThree			Stack			CoffeePreparation		
Algorithm	Train.	Gen.	All	Train.	Gen.	All	Train.	Gen.	All	Train.	Gen.	All
Stem-OB	50.7$_{\pm1.7}$	28.3$_{\pm12.8}$	31.4$_{\pm14.0}$	41.3$_{\pm0.9}$	**27.6**$_{\pm12.3}$	**29.3**$_{\pm12.4}$	85.7$_{\pm0.9}$	70.7$_{\pm29.0}$	72.6$_{\pm27.5}$	14.7$_{\pm2.1}$	**13.2**$_{\pm6.4}$	**13.4**$_{\pm6.0}$
Org	53.3$_{\pm5.2}$	22.0$_{\pm12.5}$	25.9$_{\pm15.7}$	**44.0**$_{\pm5.1}$	25.7$_{\pm15.1}$	28.0$_{\pm15.5}$	**87.3**$_{\pm2.1}$	**74.5**$_{\pm29.3}$	**76.1**$_{\pm27.7}$	**18.7**$_{\pm5.0}$	10.1$_{\pm5.9}$	11.1$_{\pm6.5}$
SRM	51.7$_{\pm0.5}$	22.4$_{\pm13.0}$	26.1$_{\pm15.6}$	28.0$_{\pm0.0}$	20.4$_{\pm9.1}$	21.3$_{\pm8.9}$	75.3$_{\pm3.4}$	58.7$_{\pm24.8}$	60.8$_{\pm23.8}$	15.0$_{\pm2.2}$	9.5$_{\pm5.2}$	10.2$_{\pm5.2}$
Mix	**53.7**$_{\pm5.6}$	**37.5**$_{\pm15.4}$	**39.5**$_{\pm15.5}$	35.7$_{\pm2.5}$	18.3$_{\pm9.2}$	20.5$_{\pm10.4}$	79.7$_{\pm3.1}$	66.8$_{\pm27.6}$	68.4$_{\pm26.2}$	17.7$_{\pm2.5}$	11.9$_{\pm6.0}$	12.6$_{\pm6.0}$

(a) Square   (b) StackThree   (c) Stack   (d) CoffeePreparation

Figure 17: **Additional MimicGen environments.** These tasks are based on MimicGen benchmark. (a) The agent must pick up square nuts and place them onto the correct pole. (b) The agent is required to stack three cubes in the correct order. (c) The task also needs stacking but only with two cubes. (d) The agent need to place a mug on the machine, open the lid, open the drawer to get coffee pod and insert it into the machine.
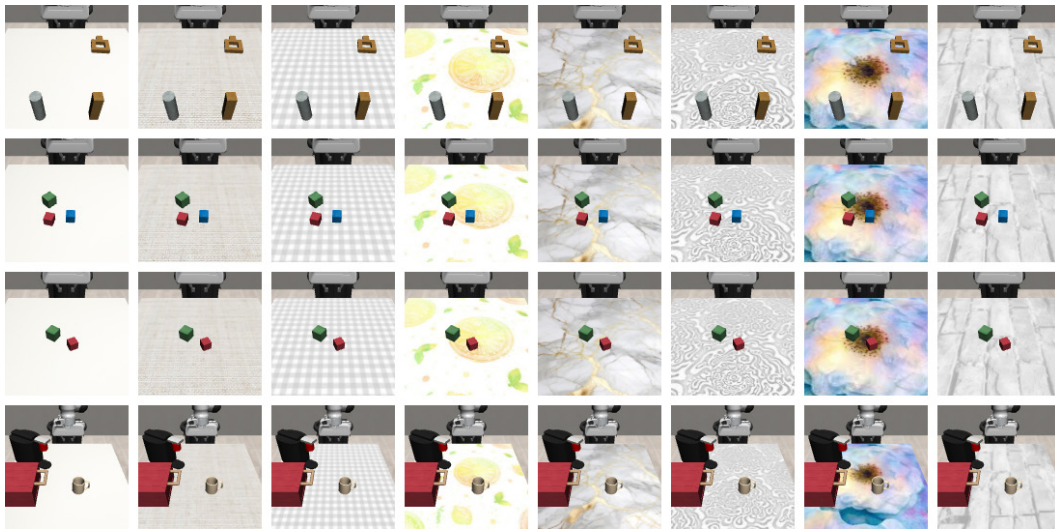
Figure 18: **Additional MimicGen Testing Settings.** Each row represents a specific task, with eight different table textures across the columns. The training environment is the first column.

Table 11: **Success rate of RO in SAPIEN.** The tasks are *PushCube*, *PickCube*, and *StackCube*. We report the mean and standard deviation of the success rate (%), and the best results are highlighted in **bold**. We constructed a simulation dataset by overlaying images from the MimicGen dataset to adapt the RO algorithm to simulation datasets.

Task \	*PushCube*			*PickCube*			*StackCube*		
Algorithm	Train.	Gen.	All	Train.	Gen.	All	Train.	Gen.	All
Stem-OB	**100.0**	**99.1**$_{\pm 1.8}$	**99.1**$_{\pm 1.8}$	50.0	**23.9**$_{\pm 20.6}$	25.1$_{\pm 20.9}$	**90.0**	**24.1**$_{\pm 28.7}$	**27.2**$_{\pm 31.3}$
RO	**100.0**	92.1$_{\pm 18.0}$	92.5$_{\pm 17.7}$	**86.0**	22.5$_{\pm 20.9}$	**25.5**$_{\pm 24.5}$	60.0	7.2$_{\pm 11.5}$	9.7$_{\pm 15.9}$

Table 12: **Evaluations in SAPIEN with normal map perturbation.** normal only: evaluations with normal map perturbation only. normal+texture: evaluations with normal map and texture perturbation. normal+light: evaluations with normal map and light perturbation. We report the mean and standard deviation of the success rate (%), and the best results are highlighted in **bold**.

Task \ Algorithm	PushCube			PickCube		
	normal only	normal+texture	normal+light	normal only	normal+texture	normal+light
Stem-OB	$98.0_{\pm2.3}$	$\mathbf{99.3_{\pm0.9}}$	$98.7_{\pm0.9}$	$\mathbf{30.0_{\pm21.5}}$	$\mathbf{2.3_{\pm0.8}}$	$\mathbf{17.7_{\pm21.5}}$
Org	$74.7_{\pm13.4}$	$39.0_{\pm1.5}$	$58.5_{\pm27.1}$	$2.0_{\pm0.0}$	$2.0_{\pm0.0}$	$2.0_{\pm0.0}$
SRM	$86.0_{\pm8.1}$	$8.3_{\pm2.7}$	$75.8_{\pm11.7}$	$2.3_{\pm0.8}$	$\mathbf{2.3_{\pm0.8}}$	$2.2_{\pm0.5}$
Mix	$\mathbf{99.0_{\pm1.0}}$	$97.3_{\pm1.9}$	$\mathbf{99.0_{\pm1.0}}$	$22.3_{\pm17.5}$	$2.0_{\pm0.0}$	$8.5_{\pm5.9}$

machine, opening the lid of the coffee machine, then retrieving the coffee pod from a drawer and finally inserting it in the machine. The test settings are shown in Fig. 18. Tab. 10 presents the success rates for these tasks. *Stem-OB* achieves competitive performance across all tasks, demonstrating its generalization capabilities.

G RANDOM OVERLAY BASELINE IN SAPIEN ENVIRONMENTS

The original random overlay baseline uses the Places365 dataset, which contains real-world images inconsistent with the simulation environments, thus we didn't take it into account in simulation tasks. To adapt it for simulation, we constructed a simulation dataset by overlaying images from the MimicGen dataset. Specifically, we randomly selected one of the seven MimicGen environments and picked an arbitrary image from a random trajectory to overlay onto the original image. Using this adapted RO dataset, we evaluated **RO** on *PushCube*, *PickCube* and *StackCube*. The results show that *Stem-OB* still outperforms **RO** in most tasks as shown in Tab. 11.
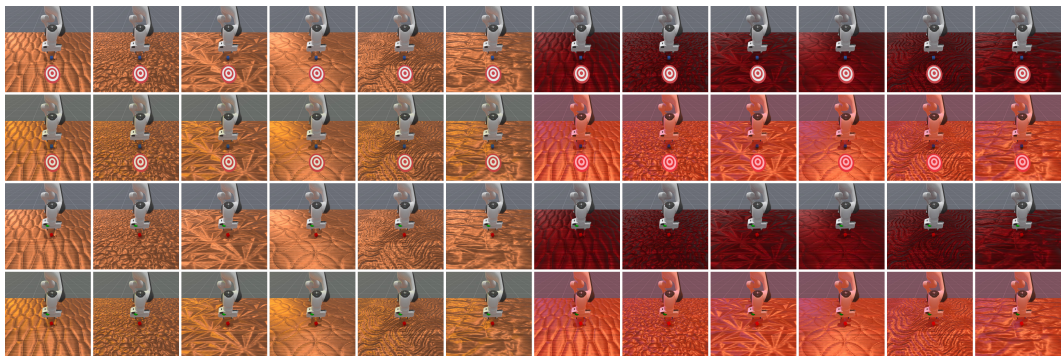
H MORE GENERALIZATION EXPERIMENTS

Figure 19: **Sapien Testing Settings.** Each task consists of 24 distinct settings, arranged in two rows. The first row of each task contains 12 settings, with the left half showing 6 *normal only* perturbation and the right half showing the *normal + texture* perturbation. The second row of each task represents 12 *normal+lighting* perturbation settings, with 2 lights combined with 6 normal maps.

We introduced a new type of visual variance in the SAPIEN environment by adding normal maps to the tabletop and combining this perturbation with other variations, such as lighting conditions and table textures. Fig. 19 provides a visualization of these new perturbations. We then evaluated *Stem-OB* and the baselines on the ***PushCube*** and ***PickCube*** tasks. The results are shown in Tab. 12. *Stem-OB* consistently outperforms the baselines, even when encountering previously unseen perturbations.

Table 13: **Success rate of R3M in SAPIEN.** The tasks are *PushCube*, *PickCube*. We report the mean and standard deviation of the success rate (%), and the best results are highlighted in **bold**. R3M(w/ freeze) is trained with the frozen encoder, while R3M(w/o freeze) is fine-tuning the encoder during training.

Task \	*PushCube*			*PickCube*		
Algorithm	Train.	Gen.	All	Train.	Gen.	All
Stem-OB	**100.0**	**99.1**$_{\pm 1.8}$	**99.1**$_{\pm 1.8}$	**50.0**	**23.9**$_{\pm 20.6}$	**25.1**$_{\pm 20.9}$
R3M(w/ freeze)	14.0	19.8$_{\pm 6.1}$	19.5$_{\pm 6.1}$	2.0	2.1$_{\pm 0.8}$	2.1$_{\pm 0.8}$
R3M(w/o freeze)	24.0	20.0$_{\pm 5.7}$	20.2$_{\pm 5.6}$	2.0	1.9$_{\pm 0.8}$	1.9$_{\pm 0.8}$

Table 14: **Success rate of SD1.4 in SAPIEN.** The tasks are *PushCube*, *PickCube*. We report the mean and standard deviation of the success rate (%), and the best results are highlighted in **bold**. We perform additional benchmarking on another version of stable-diffusion: SD1.4. For SAPIEN environment, the DDPM inverison steps are set to 15/50.

Task \	*PushCube*			*PickCube*		
Algorithm	Train.	Gen.	All	Train.	Gen.	All
Stem-OB (sd2.1)	**100.0**	99.1$_{\pm 1.8}$	99.1$_{\pm 1.8}$	**50.0**	**23.9**$_{\pm 20.6}$	**25.1**$_{\pm 20.9}$
Stem-OB (sd1.4)	**100.0**	**99.7**$_{\pm 0.7}$	**99.7**$_{\pm 0.7}$	36.0	18.3$_{\pm 14.6}$	19.1$_{\pm 14.7}$

I   COMPARISON WITH SELF-SUPERVISED REPRESENTATION LEARNING

Self-supervised representation learning (SSL) also aims to map raw observations into a unified representation space. We benchmark our method against **R3M** Nair et al. (2022) which pre-trains a visual representation using a human video dataset and incorporates a self-supervised loss term. This representation can then be used as a perception module for downstream policy learning. We simply replace our visual encoder (ResNet18) with the pretrained R3M ResNet18 model, and then experiment with two approaches: freezing the pretrained model and fine-tuning the encoder during training. We test **R3M** on *PushCube* and *PickCube*. The results are shown in the Tab. 13.

The experiment results show that the R3M pretrained encoder doesn't work well on the SAPIEN environment tasks. Since loading R3M is simple and we use the official parameters, we think this may be due to the heavy supervision imposed by R3M not suitable for diffusion policies.

J   GENERATIVE MODELS

Table 15: **Success rate of SD1.4 in MimicGen.** The tasks are *MugCleanup*, *Threading*, *ThreePieceAssembly*, and *Coffee*. We report the mean and standard deviation of the success rate (%), and the best results are highlighted in **bold**. We perform additional benchmarking on another version of stable-diffusion: SD1.4. For MimicGen environment, the DDPM inverison steps are set to 1/8.

Task \	*MugCleanup*			*Threading*			*ThreePieceAssembly*			*Coffee*		
Algorithm	Train.	Gen.	All	Train.	Gen.	All	Train.	Gen.	All	Train.	Gen.	All
Stem-OB (sd2.1)	**38.7**$_{\pm 2.5}$	**17.1**$_{\pm 11.1}$	**19.5**$_{\pm 12.5}$	31.3$_{\pm 2.5}$	**14.5**$_{\pm 12.3}$	**16.4**$_{\pm 12.8}$	19.7$_{\pm 4.8}$	12.3$_{\pm 6.7}$	13.1$_{\pm 6.9}$	**68.0**$_{\pm 5.7}$	**48.0**$_{\pm 19.7}$	**50.5**$_{\pm 19.6}$
Stem-OB (sd1.4)	35.3$_{\pm 1.7}$	15.9$_{\pm 10.0}$	18.0$_{\pm 11.2}$	**32.0**$_{\pm 0.8}$	13.4$_{\pm 10.7}$	15.5$_{\pm 11.6}$	**20.7**$_{\pm 3.9}$	**14.0**$_{\pm 6.2}$	**14.7**$_{\pm 6.4}$	60.0$_{\pm 6.5}$	37.3$_{\pm 16.0}$	40.2$_{\pm 16.9}$

We conduct the experiments on other generative models, other than Stable Diffusion 2.1, to find out whether the performance of *Stem-OB* is model-specific. We perform additional benchmarking on another version of stable-diffusion: SD1.4 in both MimicGen and SAPIEN environments. For MimicGen tasks, we use 1 / 8 as the inversion steps in DDPM inversion, and 15/50 for SAPIEN tasks. The results of the two environments are listed in the Tab. 14 and Tab. 15. The success rates of these two versions of stable-diffusion are similar. This supports our methods' feasibility regardless of the underlying pretrained diffusion model.