

422 Please see our project website at <https://view-invariant-policy.github.io/> for videos
423 and additional visualizations.

424 A Details of Model Implementations

425 All novel view synthesis methods that we consider generate novel views at a resolution of 256×256
426 given RGB images at a resolution of 256×256 . The synthesized images are later downsampled for
427 policy training.

428 A.1 ZeroNVS

429 We use the implementation and pretrained checkpoint provided by Sargent et al. [8]. As mentioned
430 in Section A, although ZeroNVS largely produces reasonable views even zero-shot, it can sometimes
431 produce images with significant visual artifacts. To filter these out, we reject and resample images
432 that have a LPIPS [35] distance larger than a hyperparameter η from the input image. We set $\eta = 0.5$
433 for all simulated experiments and $\eta = 0.7$ for real experiments. We do not extensively tune this
434 hyperparameter. If the model fails to produce an image with distance $< \eta$ after 5 tries, the original
435 image is returned.

436 ZeroNVS also requires as input a scene scale parameter. To determine the value of the scene
437 scale for simulated experiments, we perform view synthesis using the pretrained ZeroNVS check-
438 point on a set of 100 test trajectories for the `lift` and `threading` environments, and compute the
439 LPIPS score between the ZeroNVS rendered images and ground truth simulator renders for values
440 $\{0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95\}$. We find that the lowest error across
441 the tasks is achieved at 0.6 and thus use 0.6 for all environments, including the real robot experi-
442 ments.

443 While the behavior of the ZeroNVS model is somewhat sensitive to scene scale, we believe this
444 may be alleviated by selecting a wider viewpoint randomization radius at training time, which is
445 corroborated by our real robot experiments.

446 When sampling, we perform 250 DDIM steps and use a DDIM η of 1.0. We use a field of view
447 (FOV) of 45 degrees for simulated experiments (obtained from the simulator camera parameters)
448 and FOV of 70 degrees for the real world experiments (obtained from the Zed 2 camera datasheet).

449 It takes on average 8.7 seconds to generate a single 256×256 image with ZeroNVS using these
450 settings on a single NVIDIA RTX 3090 GPU.

451 A.2 Depth estimation + Reprojection baseline

452 This baseline represents a geometry-based approach that leverages depth estimation models trained
453 on large-scale, diverse data.

454 First, we use ZoeDepth (ZoeD_NK) [40], an off-the-shelf model, to perform metric depth esti-
455 mation on the input RGB image. Next, we deproject the images into pointclouds using a pin-
456 hole camera model. We rasterize an image from the points using the Pytorch3D point raster-
457 izer [43], setting each point to have a radius of 0.007 and 8 points per pixel. Finally, we use
458 a publicly available Stable Diffusion inpainting model ([https://huggingface.co/runwayml/](https://huggingface.co/runwayml/stable-diffusion-inpainting)
459 [stable-diffusion-inpainting](https://huggingface.co/runwayml/stable-diffusion-inpainting)) to inpaint regions that are empty after rasterization. We use 50
460 denoising steps as per the defaults.

461 It takes on average 2.8 seconds to generate a single 256×256 image with this baseline on a single
462 NVIDIA RTX 3090 GPU.

463 A.3 PixelNeRF

464 For PixelNeRF [22], we use the implementation from the original authors at [https://github.](https://github.com/sxyu/pixel-nerf)
465 [com/sxyu/pixel-nerf](https://github.com/sxyu/pixel-nerf). We use a pretrained model trained on the same datasets as ZeroNVS [8].

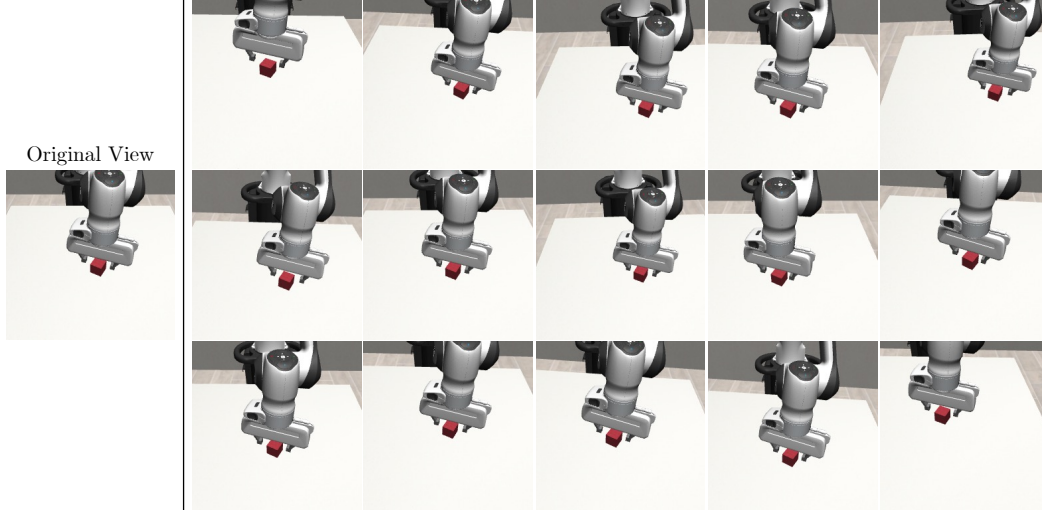


Figure 7: Example ground truth viewpoints from the “perturbation” distribution for the Lift task, rendered by the simulator.

It takes on average 5.8 seconds to generate a single 256×256 with PixelNeRF on a single NVIDIA RTX 3090 GPU.

B Simulated Experimental Details

Here we provide details regarding the simulated experimental setup. As a high level goal, we aim to minimize differences from our setup from existing robotic learning pipelines to demonstrate how this augmentation technique can be generally and easily applied across setups.

B.1 Simulation Environments and Datasets

Our simulated experiments use environments created in the MuJoCo simulator and packaged by the robosuite [44] and MimicGen [39] frameworks.

For the Lift, PickPlaceCan, and Nut Assembly tasks, the training datasets are the Proficient-Human datasets for those tasks from robomimic [36] and consist of 200 expert demonstrations each. For all MimicGen tasks (Threading, Hammer Cleanup, Coffee, Stack) the datasets consist of the first 200 expert trajectories for the “core” MimicGen-generated datasets, downloaded from https://github.com/NVlabs/mimicgen_environments.

B.2 Details for Training and Test Viewpoints

We use the same distribution of viewpoints at training time for augmenting the dataset and when testing the policies. Note, however, that images generated by novel view synthesis models are not guaranteed to actually be from the target viewpoint – only the oracle that uses the simulator to render the scene satisfies this.

B.2.1 Perturbations

This set of viewpoints are representative of incremental changes, for instance, that of a physical camera drifting over time or subject to unintentional physical disturbance. Specifically, this distribution is parameterized by a random translation $\Delta t \sim \mathcal{N}(0, \text{diag}(\sigma_t^2))$ and rotation around a uniformly randomly sampled 3D axis, where the magnitude is sampled from $\mathcal{N}(0, \sigma_r^2)$. Specifically, we set $\sigma_t = 0.03$ m and $\sigma_r = 0.075$ rad. Samples of observations taken from viewpoints drawn from this distribution are shown in Figure 7.

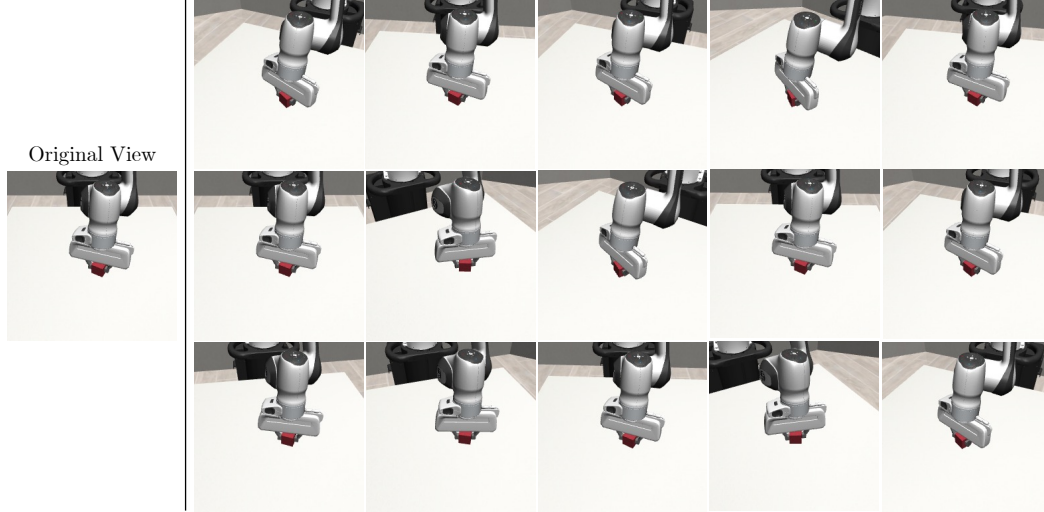


Figure 8: Example ground truth viewpoints from the “quarter circle arc” distribution for the Lift task, rendered by the simulator.

B.2.2 Quarter Circle Arc

This is a more challenging distribution of camera poses with a real-world analogue to constructing another view of a given scene. We first compute a sphere centered at the robot base and containing the initial camera pose. We then sample camera poses on the sphere at the same z height and within a 90° azimuthal angle of the starting viewpoint. The radius of the sphere is further randomly perturbed with Gaussian noise with variance σ_r^2 . Specifically, the radius of the sphere is 0.7106 m for all simulated environments, which is the distance between the camera and the robot base in the Lift task, and $\sigma_r = 0.05$ m.

B.3 Finetuning ZeroNVS on MimicGen Datasets

We finetune the ZeroNVS model on datasets from the MimicGen data of 8 tasks: stack three, square, three piece assembly, mug cleanup, pick place can, nut assembly, kitchen, and coffee prep. For each environment, we take the first 200 trajectories of the “core” MimicGen dataset for that task with the maximum initialization diversity (e.g. if Square is available in variants D0, D1, and D2, we take D2) and simulate 10 random viewpoints from the quarter circle arc distribution for each image in the dataset. We supply this as training data to ZeroNVS, using the training settings from the original ZeroNVS paper but changing the optimizer from AdamW to Adam and decreasing the learning rate to $2.5e-5$, and decreasing the batch size to 512 due to computational constraints. We finetune the model for 5000 steps using four NVIDIA L40S GPUs.

B.4 Policy Learning

We use the same policy training settings for all simulated experiments, taken from the behavior cloning implementation in robomimic. The output of the policy network is a Gaussian mixture model. A brief overview of hyperparameters, corresponding directly to robomimic configuration file keys, are listed in Table 4. Note that we do not tune these hyperparameters and simply use them as sensible defaults. We train each policy using a single NVIDIA TITAN RTX GPU.

Hyperparameter	Value
Batch size	16
Optimizer steps per epoch	500
Training epochs	600
Input image resolution	84×84
Augmentation	Random crop ($84 \times 84 \rightarrow 76 \times 76$)
Optimizer	Adam
Learning rate	$1e-4$
Actor layer dimensions	1024, 1024
GMM num modes	5
GMM min std	0.0001
GMM std activation	softplus
Visual encoder backbone	Resnet18
Visual encoder feature dim	64
Visual encoder pooling	Spatial softmax
Spatial softmax num kp	32
Spatial softmax temperature	1.0

Table 4: Behavior cloning hyperparameters for simulated experiments.

C Real World Experimental Details

Next we provide details regarding the real world experimental setup. As a high level goal, we aim to minimize the differences from existing robotic learning pipelines to demonstrate how this augmentation technique can be generally and easily applied across setups.

C.1 Real World Robot Setup

We use a Franka Research 3 (FR3) robot in our real world experiments. The hardware setup is otherwise a replica of that introduced by Khazatsky et al. [7]. Specifically, the robot is mounted to a mobile desk base (although we keep it fixed in our experiments) and two ZED 2 stereo cameras provide observations for the robot. An overview of the real-world robot setup is shown in Figure 9.

We use a Meta Quest 2 controller (also as per the DROID hardware setup) to collect teleoperated expert demonstrations. We collect 150 human expert demonstrations of the place cup on saucer task, randomizing the position of the cup and saucer after each task. Each demonstration trajectory lasts approximately 15 seconds of wall clock time.

When performing evaluations, we score task completion based on two stages: 1) Reaching the cup in a grasp attempt based on determination by a human rater and 2) Completing the task, which means that the cup is above and touching the surface of the saucer at some point during the trajectory.

C.2 Finetuning ZeroNVS on the DROID Dataset

To finetune ZeroNVS on the DROID dataset, we first collect a random subset of 3000 trajectories from the DROID dataset. Then, for each trajectory, we uniformly randomly sample 10 timestamps from the du-

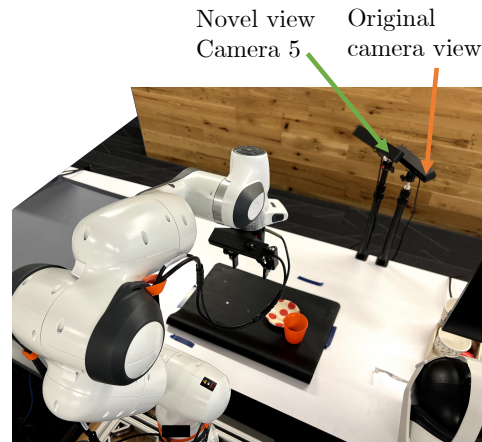


Figure 9: Experimental setup for real robot evaluation. Here we show the testing setup for one particular novel camera view, camera 5. The original camera view that data was collected using is shown by the orange arrow. We use the left camera of each stereo pair.

ration of the video, and consider the trajectory frozen at each of those times as a “scene”. Thus, we effectively have 30000 scenes. For each scene, we extract 4 views, which correspond to stereo images from the two external cameras. Although the DROID dataset does contain wrist camera data, we do not use it, as the wrist camera poses are much more challenging for synthesizing novel views.

We then perform depth estimation for each image using a stereo depth model. We then center crop all images to be square, and resize them to 256×256 to fit the existing ZeroNVS models. We obtain camera extrinsics from the DROID dataset, and use simplified intrinsics assuming a camera FOV of 68 degrees for all cameras, which we obtained from a single randomly sampled camera in the dataset. In reality, the FOV differs slightly for each camera due to hardware differences, and slightly better results may be obtained by using per-camera intrinsics.

As in the simulated finetuning experiments, we again use the training settings from the original ZeroNVS paper but change the optimizer from AdamW to Adam and decrease the learning rate to $2.5e-5$, and decrease the batch size to 512 due to computational constraints. We use 29000 scenes for training and 1000 for validation. As an attempt to reduce overfitting, we mix in a single shard of 50 scenes each from the CO3D and ACID datasets which are sampled for each training sample with probability 0.025 each. DROID data is sampled with probability 0.95. We did not extensively validate the effect of this data mixing due to computational cost of finetuning the model repeatedly, and it is likely unnecessary. We finetune the model for 14500 steps using four NVIDIA L40S GPUs.

C.3 Policy Learning

Training augmentation viewpoints. For the real world experiments, we do not have access to the test viewpoint distribution. To sample viewpoints for ZeroNVS data augmentations for these experiments, we sample from a distribution parameterized in the same way as the “perturbation” range in the simulated experiments, but with a vastly increased variance in translation and rotation magnitude intending to cover a wide range of possible test viewpoints.

This distribution is parameterized by a random translation $\Delta t \sim \mathcal{N}(0, \text{diag}(\sigma_t^2))$ and rotation around a uniformly randomly sampled 3D axis, where the magnitude is sampled from $\mathcal{N}(0, \sigma_r^2)$. Specifically, we set $\sigma_t = 0.15$ m and $\sigma_r = 0.375$ rad.

Policy learning. For policy learning on the real robot, we train diffusion policies [37]. Specifically, we use the implementation from the evaluation in the DROID paper [7] with language conditioning removed. The input images are of size 128×128 , and both random color jitter and random crops (to 116×116) are applied to the images during training. We train all policies for 100 epochs (50000 gradient steps), using 2 NVIDIA RTX 3090 or RTX A5000 GPUs.

D Additional Experiment

D.1 Increasing Number of Augmented Transitions

In the experiments presented in Section 5, we perform data augmentation via novel view synthesis by doing offline preprocessing of the dataset, augmenting and replacing each transition with a single augmented transition. However, many random data augmentation strategies for neural network training perform augmentation “on-the-fly”, applying augmentations to each particular batch. This increases the effective dataset size. Augmentation with

Effect of Additional Augmented Trajectories:
Threading Quarter Circle Arc

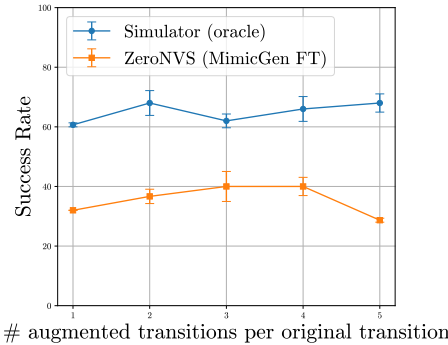


Figure 10: Results for ablation of number of augmented transitions per original dataset transitions. Overall, we see modest performance improvements from increasing the amount of augmented data, across both the oracle and learned NVS model.

591 novel view synthesis methods is too computationally expensive to apply per batch with our compu-
592 tational budget, but we are still interested in understanding how the performance of trained policies
593 is affected by increasing the number of augmented trajectories for each original dataset trajectory.

594 For the threading task with viewpoints sampled from the “quarter circle arc” distribution, we
595 trained policies on dataset containing 1, 2, 3, 4, and 5 augmented transitions per dataset transition
596 for the simulator (oracle) and ZeroNVS (MimicGen finetuned) models.

597 The results are shown in Figure 10. We find that increasing the number of augmented transitions
598 per original dataset transition yields modest improvements with both models, although there is a
599 surprising dip in performance when using 5 augmented transitions for the ZeroNVS (MimicGen
600 finetuned) model.