

479 **A Experiment Details**

480 **A.1 New Object Generalization**

481 In Fig. 5, we show pictures of seen / unseen objects. For all the new object generalizations, we
482 evaluate policies on multiple objects despite each policy being trained on a single object. Across
483 tasks, we showcase the new objects that involve variations in color and geometry.



Figure 5: Overview of objects used in real-robot experiments. In each image, the single object on the **left** side is used during data collection, and **all** the objects on the right side are not seen during training. They are used for the evaluation of new object generalization in each task.

484 **B Additional Implementation Details**

485 We describe all the details of our model implementation aside from the ones mentioned in main text.

486 **B.1 Model Details**

487 **Neural Network Details.** We use a standard Transformer [21] architecture in our paper. We use 4
488 layers of transformer encoder layers, and 6 heads of the multi-head self-attention modules. For the
489 two-layered fully connected networks, we use 1024 hidden units for each layer. For GMM output
490 head, we choose the number of modes for the Gaussian Mixture Model to be 5, which is the same
491 as in Mandlkar et al. [1].

492 **Temporal Positional Encoding.** For computing temporal positional encoding, we follow the
493 equation for each dimension i in the encoding vector at a temporal position pos :

$$PE(pos, 2i) = \sin\left(\frac{pos}{10^{2i/D}}\right)$$

$$PE(pos, 2i + 1) = \cos\left(\frac{pos}{10^{(2i+1)/D}}\right)$$

494 We choose the frequency of positional encoding to be 10 which is different from the one in the
495 original transformer paper. This is because our input sequence is much shorter than those in natu-
496 ral language tasks, hence we choose a smaller value to have sufficiently distinguishable positional
497 features for input tokens.

498 **Training Details** For point masking, we use a masking ratio of 0.6 in simulation, and 0.75 for
499 real world. Because of the limited field-of-view, the occlusion of objects in simulation is severe.
500 To properly evaluate policies in simulation, we add an eye-in-hand camera that only captures close-
501 distance depth (the depth observation is clipped to the range of gripper tips). This design choice
502 allows the policies to learn while preventing policies from relying entirely on eye-in-hand cameras.

503 In all our experiments of GROOT, we train for 100 epochs. We use a batch size of 16 and a
504 learning rate of 10^{-4} . We use negative log-likelihood as the loss function for action supervision
505 loss since we use a GMM output head. As we notice that validation loss doesn't correlate with
506 policy performance [1], we adopt a pragmatic way of saving model checkpoint as in Zhu et al. [3],
507 which is to save the checkpoint that has the lowest loss over all the demonstration data at the end of
508 training. We apply a gradient clip at 100 across all the experiments to prevent training from gradient
509 explosion.

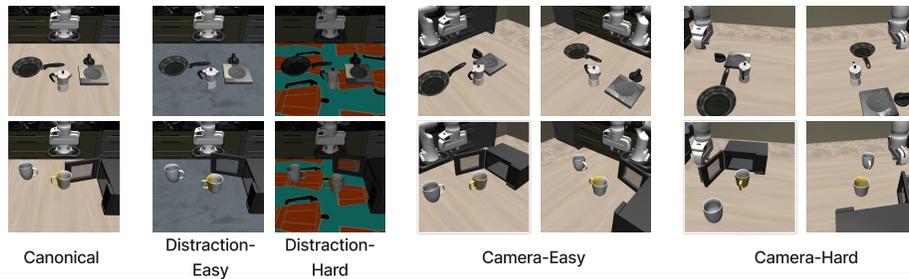


Figure 6: Screenshots of simulation tasks, for both Canonical, Background (Easy), Background (Hard), Camera (Easy), Camera (Hard)

510 C Environment Details

511 Here we describe more about our environment designs.

512 **Baseline Implementations** As we mainly focus on comparing the effectiveness of representations,
 513 all the transformer-based baselines (VIOLA and MAE-POLICY) use the same architecture for a fair
 514 comparison. Since none of the baselines were proposed for learning with RGB-D observations, we
 515 implemented them with minimal changes to accommodate the RGB-D observations. For BC-RNN,
 516 we encode depth images with an additional resnet encoder, and concatenate the features along with
 517 the other features as inputs to the RNN backbone. For VIOLA, we extract the task-agnostic pro-
 518 posals and back project each proposal into point clouds, giving VIOLA a fair comparison with our
 519 approach. As for MAE-POLICY, we patchify both RGB and depth images and pass the unmasked
 520 patches into the transformer architecture.

521 **Generalization Settings in Simulation** Fig. 6 shows the initial conditions of simulation tasks.
 522 Note that “Put the moka pot on the stove” and “Put the frying pan on the stove” share the same
 523 initial distributions, so we only visualize one of the tasks for showing the initialization settings.
 524 Background (Hard) is the hard level as we changed both the lighting conditions, and add the
 525 table cloth that has object patterns. Camera (Hard) is harder than Camera (Easy), as the cam-
 526 eras are rotated with 40 more degrees. Such a wild change in camera viewpoints results in a very
 527 different perspective on objects. Challenges the generalization abilities of policies.

528 **Real-Robot Setup** We use a 7-DoF Franka Emika Panda arm in all tasks. For real robot end-
 529 effector control, we use the Operational Space Controller [58] implemented from Deoxys [3]. The
 530 controller operates at 20Hz alongside a binary gripper control. We use Intel Realsense D435i as the
 531 workspace camera.

532 **Success Conditions of Real-Robot Task** To quantify the policy performance, we explain the
 533 success conditions for all the tasks as follows:

- 534 • “Pick Place Cup”: The cup is placed on the coaster upright.
- 535 • “Stamp The Paper”: The robot stamps on the paper and put the stamp back to the table
- 536 • “Take The Mug”: The mug is taken from the coaster, and placed on the table steadily.
- 537 • “Put the Mug On The Coaster”: The mug is put on the coaster steadily.
- 538 • “Roll the Stamp”: The robot successfully rolls the stamp for half of the paper length.

539 **Data Collection** We use a 3Dconnexion SpaceMouse to collect 50 human-teleoperated demon-
 540 strations for every real-world task. As for simulation, the simulation environments directly provide
 541 50 high-quality teleoperated demonstrations, so we directly leverage them for policy learning.

542 **Evaluation Horizons** For policy evaluation, we limit the decision horizons to 600 timesteps.