

## Supplementary Material

### A Pose and Trajectory L1 Metric

We define an L1-based metric for pose error that combines the positional and rotational components in a single scalar. The position error is computed as the L1 norm of the difference between predicted and ground truth translation vectors. The rotation error is measured in degrees, and we normalize both terms using the equivalence of  $1 \text{ cm} = 1^\circ$ . The overall pose L1 error is given by:

$$\text{mean taj. L1} = \frac{1}{N} \sum_{i=1}^N \left( \|\mathbf{t}_i - \hat{\mathbf{t}}_i\|_1 + \angle \left( \hat{R}_i^{-1} R_i \right) \right) \quad (1)$$

where:

- $N$  is the number of poses in the trajectory.
- $\mathbf{t}_i, \hat{\mathbf{t}}_i \in \mathbb{R}^3$  are the predicted and ground truth translation vectors for sample  $i$ .
- $R_i, \hat{R}_i \in \text{SO}(3)$  are the predicted and ground truth rotation matrices.
- $\angle(\cdot)$  denotes the angle (in degrees) of the relative rotation.

### B Simulation Setup

We make use of the Maniskill3 simulation environment [13]. For each interaction, we predict two waypoints corresponding to the grasping and gripper opening position. These are converted into a trajectory by adding a raising and destination alignment movement. These are then executed using an inverse kinematics planning system. The reward is computed by comparing the L2 norm between the object’s current and goal poses. This quantity is mapped to the unit range by normalizing it with the initial distance between the initial pose and the goal and computing  $1 - \text{this quantity}$ . The resulting rewards are clipped to the unit range. Success rates are computed using a reward threshold of 0.75.

$$\text{reward} = \text{clamp} \left( 1 - \frac{\|\mathbf{p}_A - \mathbf{p}_{\text{goal}}\|}{\|\mathbf{p}_A^{\text{init}} - \mathbf{p}_{\text{goal}}\|}, [0, 1] \right) \quad (2)$$

where:

- $\mathbf{p}_A \in \mathbb{R}^3$  is the current position of object,
- $\mathbf{p}_A^{\text{init}} \in \mathbb{R}^3$  is the initial position of the object at the start of the episode,
- $\mathbf{p}_{\text{goal}} \in \mathbb{R}^3$  is the target goal position for object

### C Objaverse Asset Curation

Objaverse assets were created by a mix of artists, the sizes of the objects are not scaled canonically, and the provided category label is not fine-grained enough to generate descriptions. We follow the following procedure to create a large dataset of diverse high-quality models with good, concise text descriptions. We start by subsampling the dataset of 1M shapes to 600k. Then, to obtain the relevant meta-information of model scale as well as text description, we start by using GPT-4V to produce long-form descriptions including a guess of the object’s dimensions, based on several rendered perspectives. We do an intermediate filtering step where each object is filtered by size: a) only objects with all side lengths between 0.01 m and 0.20 m are retained, and b) objects with a side length ratio exceeding 5 (i.e., too elongated) are excluded. After filtering, objects are rescaled such that the smallest side in the x-y plane is at most 0.07 m, ensuring compatibility with the gripper size (0.08 m), including margin.

515 To further obtain concise descriptions, we use GPT-4 to summarize the long-form descriptions. In  
 516 the final verification step, the short form descriptions were evaluated using SigLIP [50]. Specifically,  
 517 we compare the embeddings of the images against the short descriptions. Further, we use SigLIP to  
 518 evaluate the alignment of the short descriptions with the captions “grayscale image” and “cartoon  
 519 low-poly model”. Eventually, we only select objects for our asset set where the distance between  
 520 SigLIP embeddings passes a threshold, resulting in a final object set of around 7k models.

## 521 D Simulation and Real Experiments

522 We use different versions of the training and evaluation datasets in our pipeline. Training datasets are  
 523 curated using ManiSkill3 simulation environment [13], where the objects in the scene come from  
 524 either CLEVR [44] or Objaverse [43] datasets. We further introduce two difficulty levels of the  
 525 training datasets *CLEVR/Objaverse-easy* and *CLEVR/Objaverse-hard*, where the difference is in the  
 526 scene and camera field-of-view randomization between them. Fig. 6 shows examples from the harder  
 527 version of the training datasets with both CLEVR and Objaverse assets.

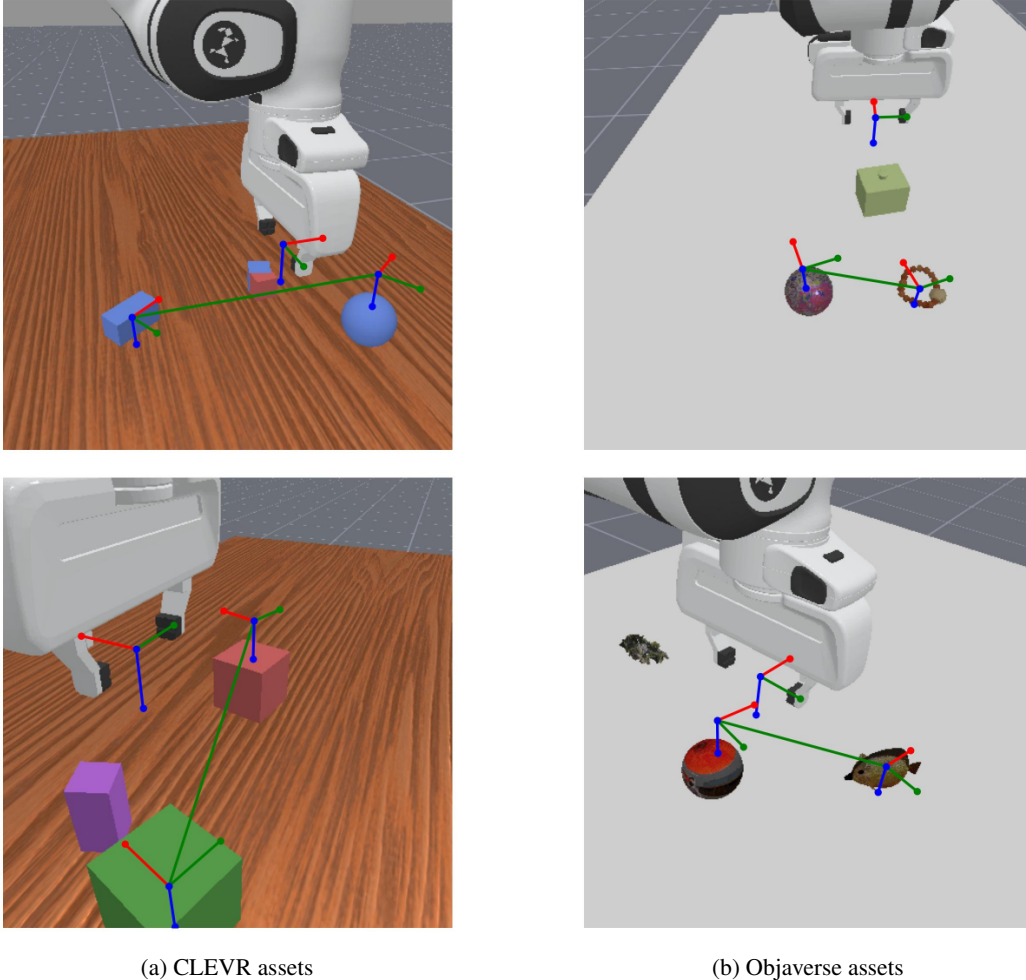


Figure 6: **Qualitative examples of our training data.** Our training data consists of different 3D objects placed into the scene. We distinguish between 2 different object groups - CLEVR-like assets and Objaverse assets.

528 We evaluated our models on four different application domains - simulated data, real data, simulations,  
 529 and real robot setups. Evaluations on simulated data and simulations were performed in the same  
 530 setup as the training data was curated, but with new environments and scenes.

For the real data evaluations, we used different subsets of the DROID dataset [12]. This data first needs to be filtered as the quality of the extrinsic calibration is very variable, something that has been noted and systemically addressed in recent work [here](#). For our small-scale evaluation sets, we manually filtered the data using the projection of the end-effector position into the image. For our testing purposes, after manual filtering based on calibration, we selected text prompts containing the word “block” and took 160 episodes without background clutter, which made our *DROID-hard* subset. This subset contains images of blocks as well as a few visually confounding items. The presence of these leads to false predictions and makes it suitable to evaluate predictions of multiple trajectories. To make predictions without confounding objects, we created *DROID-easy* subset, which blurred out the confounding objects, see Fig. 8 for example.

For real robot evaluations, we used a Franka Panda robotic arm. The real robot was run using a ROS setup, the inverse kinematics planning was done using `bio_ik` package. We used the version of our network without depth inputs, as we did not implement any depth augmentation, to compensate for the missing depth we projected the grasp point onto the closes valid depth value along the position ray.

## E Training Details

In Tab. 4, we provide an overview of the hyperparameters that we use in our experiments. We leverage the Hugging Face library and the pretrained PaliGemma2 [40] model for fine-tuning.

Hyperparameter	Value
Learning rate	3e-5
Learning rate scheduler	cosine
Warmup ratio	0.05
Optimizer	Adafactor
Batch size train	32
Training epochs	1
Training iterations	4687 (150k samples)
Trainable layers	Self-attention layers only

Table 4: Training hyperparameters used throughout in our experiments.

## F One-shot Imitation from Demonstrations

Here we provide more information about extending our system to support few-shot imitation from demonstrations. We provide examples of the used prompts and visualize the predictions from simulations and real-data evaluations.

Our imitation extension requires a specific prompt format which follows the template of `<demo img> + <demo robot state> + <demo trajectory> + <live img> + <live robot state> → <estimated trajectory>`. We further provide the information of the robot state after the images. An example of one prompt is shown in Fig. 7, note that no explicit text description is given, only the tokens of the demonstration trajectory.

To create demonstration - live image pairs, we implemented a look-up table in the dataset. We sample demonstration-live image pairs such that each unique combination is used only once and never repeated during training. For the evaluation, we are using a hold-out validation dataset with the same distribution as the training, but in new environments. We again repeat the sampling process and conduct an evaluation over 10k pre-sampled combinations. When evaluating in simulation, we use a hold-out validation dataset to fetch a demonstration pair that corresponds to the given simulation task. Fig. 8 shows predictions of the imitation model trained on CLEVR-hard dataset version. Our imitation model generalizes well to different application domains, especially to the Objaverse dataset, where the imitation is performed with completely unseen objects.

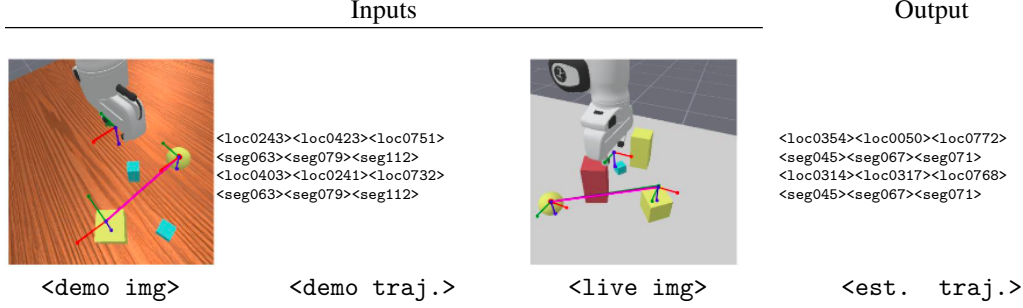


Figure 7: **One-shot prompt example.** Prompts consist of a demonstration image-trajectory path and a live image for which the model should predict the trajectory of the task represented with demonstration data. No language description is provided, yet the model is able to imitate the task. The given prompt represents the task, "move large yellow sphere onto large yellow cube". The pink line in the first image presents the demonstration trajectory, while the pink line in the second image presents the predicted trajectory. Ground truth of the live image trajectory is presented with a green line, but it is not visible due to overlapping. Robot state inputs not included for brevity.

## 567 G Input Image Cropping

568 We evaluate the effect of different crop strategies on performance by choosing different crop centers  
569 and applying zero padding, as shown in Fig. 3. When cropping, the region of interest around the  
570 task-relevant objects is enlarged before processing. The center is set to: 1) image center, 2) start object,  
571 3) middle point between start and end objects. The crop of size  $w \times w$  is taken without zero-padding  
572 (valid mode) or with zero-padding (non-valid mode) and resized to model image resolution  $224 \times 224$ .  
573 Absence of padding doesn't preserve aspect ratio. However, we emphasize that solving object scale  
574 sensitivity is not the primary focus of this work, and we leave more general solutions (e.g., multiscale  
575 feature extraction or higher-resolution processing) to future research.

## 576 H Beam-Search-NMS Implementation

577 We propose a variant of beam-search that also does non-maximum suppression over a span of  
578 spatially contiguous tokens. This is done in the following manner: a Point  $x$  is a local maximum  
579 if  $p(x) \geq p(x') \forall x' \in [x - w, x + w]$ . With a noisy distribution,  $w$  should be larger; however,  
580 with too large  $w$ , we will suppress all maxima except the global one. Thus, we find that  $w = 100$   
581 was sufficiently large. Our decoding procedure is beam search with  $n = 3$ . After processing the  
582 coordinate or angle distribution with NMS and suppressing all non-maxima by setting the token  
583 log-probability to  $-\infty$ , the next top  $n$  tokens are selected.

584 We compare our beam search-NMS with other decoding strategies using trajectory mean L1 error  
585 (Tab. 3). Additionally, we plot the beam score (log-probability) vs. the mean L1 error of the trajectory.  
586 Both standard beam search and beam search-NMS show correlation between beam log-probability  
587 and error (Fig. 9), which enables us to compute a precision-recall curve in object detection style by  
588 replacing IoU with L1 measure (Fig. 10). Our NMS approach has an mAP of 0.31 on original size  
589 DROID-hard images, while standard beam search has an mAP of 0.11.

## 590 I mAP Calculation

591 To evaluate the generation of multiple trajectories, we adapt the mean Average Precision (mAP)  
592 metric from the COCO object detection challenge [51]:

- 593 1. Instead of bounding boxes, we compare trajectories.
- 594 2. For each episode, we have one ground-truth trajectory and several predictions, for the  
595 confidence of predictions, we use the log-probability of the beam, see Fig. 9.

596       3. We replace intersection over union (IoU) with the mean L1 error over keyposes, a prediction  
597       is considered a false positive if the L1 error is larger than a threshold. See Fig. 10.

598   In this metric, we don't average of classes, only different values of L1 thresholds. Using this metric,  
599   we compared our beam search-NMS with standard beam search, as the latter is the second-best  
600   variant on the DROID-hard dataset Tab. 3. The mAP and precision-recall curves are shown in Fig. 10.

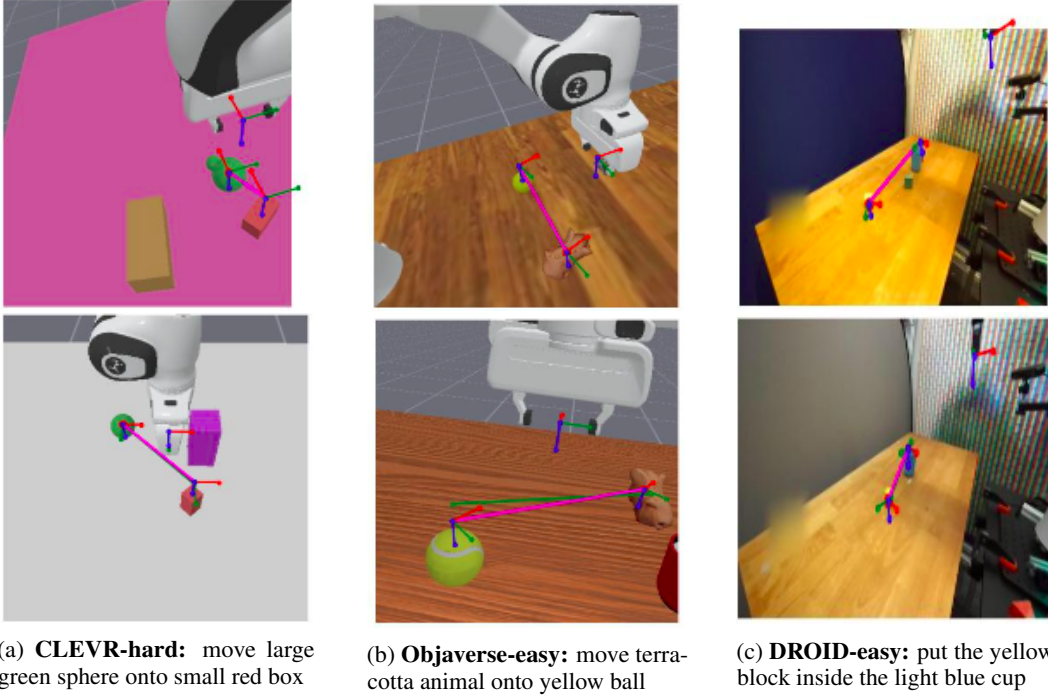


Figure 8: **Demonstration and live image with predictions.** Examples of demonstration image in the first row with visualized ground truth trajectory (pink line) and live image with both ground truth (green line) and predicted trajectory (pink line) in the second row for three different datasets - simulation data matching the training distribution, the Objaverse dataset, and DROID easy. Our imitation model performs well on all three application domains, showing good generalization to imitation with completely unseen objects in the Objaverse dataset.

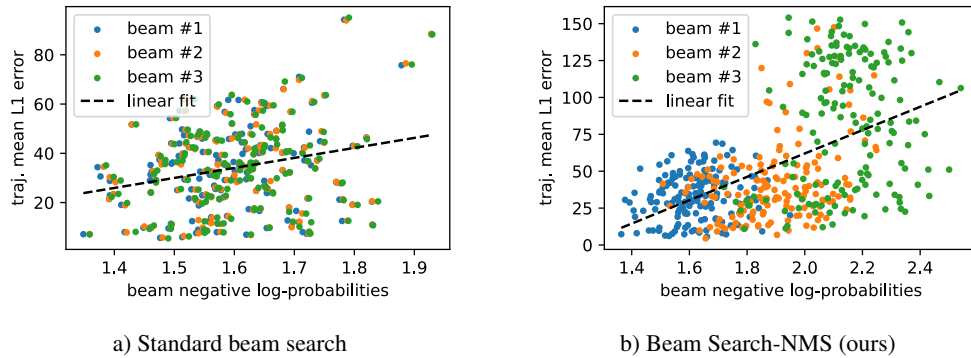


Figure 9: **Beam log-pobs correlate with L1 error.** Higher negative log-prob means lower confidence. Existence of correlation allows us to use log-probs as confidences. Results show an exploration-exploitation tradeoff. Standard beam search generates low error samples, these predictions are not very diverse, see a). Our approach generates more diverse predictions, allowing it to explore other possible trajectories. The spearman rank coefficients are 0.17 and 0.49 respectively.