cVLA: Towards Efficient Camera-Space VLAs

Max Argus, Jelena Bratulić, Houman Masnavi, Maxim Velikanov
Nick Heppert, Abhinav Valada, Thomas Brox
University of Freiburg, Germany
argusm@cs.uni-freiburg.de

Abstract: Vision-Language-Action (VLA) models offer a compelling framework for tackling complex robotic manipulation tasks, but they are often expensive to train. In this paper, we propose a novel VLA approach that leverages the competitive performance of Vision Language Models (VLMs) on 2D images to directly infer robot end-effector poses in image frame coordinates. Unlike prior VLA models that output low-level controls, our model predicts trajectory waypoints, making it both more efficient to train and robot embodiment agnostic. Despite its lightweight design, our next-token prediction architecture effectively learns meaningful and executable robot trajectories. We further explore the underutilized potential of incorporating depth images, inference-time techniques such as decoding strategies, and demonstration-conditioned action generation. Our model is trained on a simulated dataset and exhibits strong sim-to-real transfer capabilities. We evaluate our approach using a combination of simulated and real data, demonstrating its effectiveness on a real robotic system.

Keywords: VLAs, Manipulation, Imitation Learning

1 Introduction

Vision-language-action (VLA) models integrate visual understanding with actionable decision-making by jointly learning from visual, linguistic, and interaction data. These methods enable fine-grained perception and action generation, allowing them to solve a diverse range of tasks [1, 2, 3]. When trained on large-scale datasets of robot demonstrations, VLAs can generalize across a variety of robots and environments [4]. However, for further advancement of VLAs, we identified several key constraints: a) Computational costs: Training VLAs demands significant computational resources, making experimentation challenging. b) Data limitations: Collecting high-quality, multimodal real-world datasets that pair all three modalities —vision, language, and interaction data —is expensive and time-consuming. c) Evaluation and benchmarking: Standardized benchmarks for assessing VLAs performance often rely on real-world rollouts, making consistent comparisons difficult. This work addresses these constraints by proposing a lightweight VLA system trained on a controllable synthetic dataset and designed for broad applicability across different domains.

Evaluating and potentially training in simulation may be a key method to address these problems. Its use is already widespread in other robotic areas, such as learning navigation and locomotion [5, 6, 7], but this is not yet the case for VLAs. This may be due to the high precision required for control, the large number of degrees of freedom, and complex contact interactions [8, 9] and scene complexity. Bridging this gap remains a key research area in robotic manipulation. To this end, we utilize training from simulations by constructing a curated dataset with a strong camera viewpoint and object variation. The simulations and augmentations are carefully constructed to enable sim-to-real transfer. Based on this data, we train a VLA based on the PaliGemma architecture [10]. We formulate our learning tasks as a one-step[1] prediction of end-effector keyposes, which allows efficient training on

[1]Note that the description as 1-step models is sometimes also used to refer to models having only a single observation time-step as input, instead of the history [11].

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

numerous scenes. We evaluate our VLA systems on the DROID dataset [12], simulations through ManiSkill [13], and a real robot set-up. Additionally, we explore the effect of defining keyposes in image frame coordinates instead of 6D end-effector poses. Given our architecture's similarity to standard VLMs, we investigate several inference-time strategies, such as input image cropping and multiple prediction generation, and evaluate their impact on the final model performance.

In this paper, rather than training a general-purpose foundation model, we focus on a narrow data distribution with a limited set of tabletop tasks, restricting ourselves to quasi-static manipulation and generating actions with low temporal resolution. We hope that our small-scale experiments can provide helpful insights into the factors affecting VLA performance and contribute to eventually scaling VLA systems. Our contributions can be summarized as:

- An efficient setup for training and evaluating VLA models, with a diverse curated collection of shapes and texts, including a lightweight 1-shot imitation system.
- An investigation into inference time prediction strategies for VLAs and their evaluation, including a new decoding algorithm called beam-search-NMS.
- Public release of code, datasets, and models at `available upon acceptance`.

2 Related work

Vision-Language-Action Systems Recent VLA models integrate visual perception, language understanding, and action generation to achieve generalist robotic skills [1, 2, 3]. A0 [14] introduces affordance-aware representations for cross-platform manipulation. TraceVLA [15] enhances spatial-temporal reasoning through visual trace prompts. GR00T N1 [16] scales VLA systems to humanoid robots by employing a dual deliberative and reactive system design, achieving strong generalization across different embodiments. Together, these works highlight progress toward unified high-level understanding and low-level control. Molmo [17] and RoboPoint [18] take an intuitive but different view on the problem by introducing the concept of pointing directly in image space.

Trajectory Prediction and Waypoint Representations are critical for robust robot control. Inferred keyposes have been successfully applied to solve complex robotic manipulation tasks [19]. Extending the concept, HDP [20] suggested connecting keyposes through diffusing low-level control actions. Most recently, PPI [21] introduces hybrid 6-DoF keyposes and pointflows to maintain spatial precision while supporting flexible closed-loop control, enabling superior two-arm manipulation. Such mid-level waypoint structures blend discrete and continuous cues, improving planning and execution across complex tasks.

Training from Simulation for VLA models enables scalable learning but faces challenges in sim-to-real transfer. DexGraspVLA [22] combines a pre-trained vision-language planner with a diffusion-based controller, using a mix of real and simulated data to achieve robust zero-shot dexterous grasping. Robot manipulation benchmarks like CALVIN [23] and RLBench [24] provide simulated tasks to support large-scale model training. Advances in simulation domain randomization, heterogeneous datasets, and real-world alignment are key to bridging the sim-to-real gap.

Auxiliary Visual Tasks help VLAs ground their predictions spatially. LLARVA [25] uses 2D trace supervision to align vision and action, improving task success rates. Gemini Robotics-ER [26] leverages auxiliary outputs such as keypoint detection and motion sketching to enhance multistep reasoning and manipulation. Incorporating segmentation, depth estimation, and affordance prediction further improves generalization to unseen scenarios. 3D-VLA [27] defines multiple auxiliary tasks and also takes depth values as inputs.

Evaluation of VLAs is investigated in a number of recent works. This includes evaluating VLAs trained primarily on real data through the use of aligned simulations called real-to-sim evaluations [28, 29]. Long-horizon trajectory prediction is performed in VLA models for autonomous driving [30], and in some cases, also considering metrics sensitive to the diversity of predictions [31].

Few-Shot Imitation from Demonstrations remains a key challenge in robotics, particularly when scaling efficiently to new tasks. Early approaches introduced meta-learning for one-shot imita-

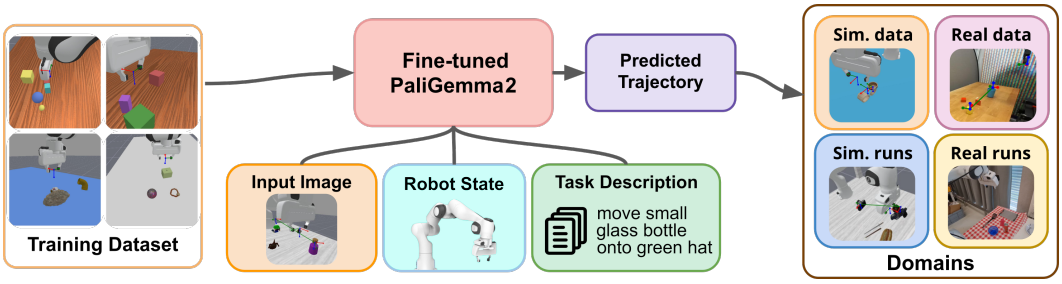


Figure 1: **Overview of cVLA.** Our lightweight method is based on fine-tuning a PaliGemma2 [40] model for trajectory prediction using our curated dataset with a single image, robot state, and task description as inputs. Our synthetic training dataset is built from different simulations of pick-and-place tasks, which enables easy scaling and an efficient training pipeline. The approach shows good generalization across different application domains, including simulation, real data, and real robot setups, and offers a simpler setup for experimental research and development of VLAs.

tion using task-specific demonstration-action pairs [32] and employing metric learning to embed demonstrations, enabling strong retrieval-based generalization [33]. Recent works explore data-efficient retrieval by selectively utilizing extensive unlabeled datasets [34] and enhancing retrieval with optical flow representations [35]. One-shot methods [36, 37] achieve success with a single unannotated demonstration, while Di Palo and Johns [38] demonstrate a few-shot visual imitation through in-context learning with pre-trained transformers. Parallel to us, Fu et al. [39] proposes an in-context imitation learning method which requires training and context data collected in the same environment.

3 Technical Approach

In the following, we detail our technical approach, summarized in Fig. 1. First, we describe our base model architecture in Sec. 3.1. Then, we introduce our novel action representations, combining image-frame coordinates and camera frame-poses in Sec. 3.1. Thirdly, we explain how depth information is incorporated into our model in Sec. 3.1. Finally, Sec. 3.2 outlines the extension of the base method to few-shot trajectory imitation set-up.

3.1 Base Model

We fine-tune a pre-trained vision language model (VLM), in our case PaliGemma2 [40]. By using an already pre-trained model, the result is an efficient and robust VLA system. Following standard VLA prompting conventions, we design our prompts as: `<live img(s)>` + `<robot state>` + `<task description>` $\rightarrow$ `<estimated trajectory>`, where `<live img(s)>` are an RGB and optional depth image (see Sec. 3.1), `<robot state>` is the current end-effector pose of the robot and `<task description>` an instruction in natural language. For `<estimated trajectory>`, unlike most other VLAs, we also make the following design decisions: a) instead of predicting a full trajectory, we predict trajectory keyposes, of which we predict only two. These are then converted into a robot trajectory using a low-level planner. Then b) we also make a one-step prediction, i.e., we predict the entire trajectory for a scene in one step. This choice has the drawback of restricting the flexibility of the system, however it has the advantage of making training more efficient.[2]

Our efficient VLA model is fine-tuned only on attention layer parameters, which, although a simple and lightweight modification, ensures creation of a strong trajectory prediction model for our use cases, which also allows investigating inference-time strategies. The training procedure and hyperparameters are described in detail in the appendix.

[2]Again here we do not aim to train a foundation model, however strategies like increasing temporal resolution is a possible extension.

Action Representations: Similar to RT-1 [41], we encode 6-DoF gripper poses using discretized tokens. PaliGemma2 [40] contains special tokens for image detection and segmentation, which we repurpose for pose prediction similar to [42]. Instead of encoding actions as end-effector deltas in robot frame coordinates, we encode actions as absolute positions in either the robot-base coordinates or image-frame coordinates, i.e., the normalized width, height, and distance from the camera. We extend the localization tokens (n=1024) to also predict depth, given the gripper position. Orientations are encoded using the segmentation tokens (n=128). The same scheme is used for both image-frame and robot-frame actions. Additionally, we experiment with using a smaller number of tokens to predict the position (n=512, 256, 128). This frees up the tokens to be used for predicting depth separately, an approach we also evaluate. An example of the logit distribution is shown in Fig. 4.

Depth Input: We extend our approach to utilize depth observations as input. To leverage the strong image encoders available for RGB images, we convert our depth maps into RGB using Matplotlib's viridis color map. These images are then processed with the same pre-trained image encoder as the natural images.

3.2 Robotic Imitation

We extend our approach to few-shot imitation learning by conditioning trajectory prediction on demonstration image-trajectory pairs instead of a natural language text description. The system infers the task from the given demonstration image-trajectory pair and must apply it to a new scene image - similar to in-context imitation learning, but in our case, we train the model to learn how to do imitation. We do not perform any fine-tuning on the novel scene image during inference time.

Our extended approach now introduces a multi-step reasoning process: given the context template of `<demo img>` + `<demo trajectory>` + `<live img>` $\rightarrow$ `<estimated trajectory>`, the model must infer the task from the demonstration image (input) and trajectory (output) pair, map the object positions to the associated tokens, and establish the correspondence between the objects in the new scene and the predicted trajectory. At test time, we sample demonstration pairs from hold-out data where the task is shared between the demonstration pair and the live image.

To enable this, we expand our training datasets by building a task-demonstration sampler. We build a look-up table of available tasks and generate a large number of random demonstration-query pairs from available scenes. Every scene can be seen only once in the pair, either as a demonstration or as a query. Due to the high number of possible demonstration pairs, we fine-tune the model for 16k iterations, keeping other hyperparameters unchanged from the original setting. Further details and exact prompt examples are provided in the appendix.

4 Dataset Generation

Our VLA sysem makes use of simulated data for training, and a combination of simulated and real data for evaluation. In the following, we describe the data collection procedure.

4.1 Simulated Training Dataset Generation

We use the ManiSkill [13] simulator to create our environments. The following outlines our data generation procedure, the 3D objects used, and our suggested augmentation strategies.

Generation Procedure: To generate a new data sample, we spawn a set of objects and an instruction, and then calculate the target object pose. We then use an analytical grasp model to generate grasps on the object and use the privileged information of known object poses to calculate a release pose. While this step can be easily performed offline to speed up generation time, we also extend our simulation to execute the task and actually verify task success. For further information, see Appendix B.

3D Model Assets: We use two different sets of object assets – a set of simple geometric shapes and a set of real-world objects, scraped from the Objaverse dataset [43]. We show example images of the objects in Appendix D.

CLEVR: Simple geometric shapes of different sizes, inspired by the CLEVR [44] dataset. Specifically, our environment consisted of three shapes: a cube, a block, and a sphere; two different sizes, with diameters of 7 cm and 3.5 cm, respectively; and eight colors.

Objaverse: To create a more diverse training and testing scenario, we construct a simulation with a large and diverse set of assets. To automatically generate text instructions for training, we require realistically scaled models combined with a concise text description. This is done similarly to Spoc [45], using the Objaverse-1M [43] dataset and described in detail in Appendix C.

Augmentations and Randomization: We apply standard image augmentations, similar to [45]. Additionally, we perform background image randomization, similar to Kubric [46], using indoor scene images from [47] to replace everything except the robot and objects, with a probability of 0.2. We perform text randomization by constructing text templates from the training split of our test data and filling in the relevant object names.

We also include randomization in the scene generation pipeline, offering easier and harder versions of the dataset. The easier version features less randomization, while the harder version includes severe scene and camera field-of-view randomization. For the harder variant of the dataset, we additionally perform a visibility test to ensure that only physically plausible environments are considered. Thus, we have four variants of the training data: CLEVR-easy, CLEVR-hard, Mix-easy, and Mix-hard. Simulation experiments are conducted using either the CLEVR variants or the Objaverse-easy and Objaverse-hard sets.

4.2 Real Evaluation Data Generation

We use sequences from DROID [12] dataset, an existing robot manipulation dataset, to evaluate our model's performance. It has a diverse mix of scenes and actions, as well as the extrinsic calibration information necessary to evaluate our system. However, the quality of the extrinsic calibration is inconsistent, thus we need to manually filter the data using the projection of the end-effector position into the image. For further information, see Appendix D.

To simplify the evaluation, and since our training data only involves move-A-to-B actions, we extract two subsets from DROID [12] in which cubes are moved in this manner. From these sequences, we take only the initial frames. The first subset, ***DROID-hard***, includes images with confounding objects. It is created to test the model's ability to predict the multi-modal distribution of trajectories. The second, ***DROID-easy***, has confounding objects blurred out, creating an easier setting in which to test generalization performance.

For offline evaluation, we calculate the L1 error for positions and rotations between predicted and ground-truth poses. For further details see Appendix A.

5 Experiments

We evaluate several aspects of our VLA system. First, the effect of various design choices on performance within the simulation domain in Sec. 5.1. Second, how our model can be used to do one-shot imitation in Sec. 5.2. Third, we investigate how inference time strategies can be used to boost performance in Sec. 5.3. Finally, we show zero-shot inference on a real robot without any real-world fine-tuning in Sec. 5.4.

5.1 Action Encoding, Depth, and Domain Randomization Ablations

As described earlier, our method leverages two different versions of the training dataset, followed by visual and textual enhancements. In Tab. 1, we evaluate the influence of each component on the final model as well as including auxiliary depth information in simulation (see Appendix B for further information about the simulation setup). All methods are trained on a harder version of the mix dataset, which includes camera and scene randomization.

Table 1: **Ablation study on simulation success rate.** We evaluate using CLEVR or Objaverse assets for differently randomized versions of the simulation (where easy uses fewer cameras and scene randomness). We observe clear patterns; adding depth to the prompt improves performance in all scenarios, and training with augmentation harms simulation performance.

CLEVR	Objaverse	Augs.	Depth	Objaverse SR (↑) easy	Objaverse SR (↑) hard	CLEVR SR (↑) easy	CLEVR SR (↑) hard
✓				8%	4%	40%	42%
✓		✓		0%	4%	46%	26%
✓	✓			18%	24%	44%	42%
✓	✓	✓		18%	20%	34%	32%
✓		✓	✓	6%	6%	**56%**	50%
✓	✓	✓	✓	**20%**	**30%**	52%	**54%**

Table 2: **One-shot imitation with demonstrations.** Our efficient pipeline can be easily extended into an imitation learning model, which achieves good success rates in robotic simulations.

Train Data	CLEVR Sim. SR (↑) easy	CLEVR Sim. SR (↑) hard	Static data traj. L1 error (↓) DROID-easy	Simulation data	Objaverse-easy
CLEVR-easy	**70 %**	18%	16.37	**3.19**	15.31
CLEVR-hard	44%	**28%**	**11.56**	6.41	**14.37**

We observe that adding depth information to the model significantly improves performance in simulation success rates and results in fewer drastic failure cases. Moreover, training solely on CLEVR assets improves performance on CLEVR-based simulations, but fails to generalize to Objaverse-based simulations, demonstrating the need for diverse 3D assets.

Next, we compare different action representation schemes in Fig. 2. We compare the performance in terms of success rate on the CLEVR-easy simulation, where we observe that the camera frame performs better on average. This likely underestimates the utility of image frame coordinates, as in these simple environments, it is easier to overfit, e.g., on gripper appearance and camera intrinsics.

5.2 One-Shot Imitation Experiments

We further evaluate our system for simple one-shot trajectory-conditioned imitation from demonstrations, where the task is inferred from a single demonstration consisting of an image and a trajectory, rather than a natural language description. As described in Sec. 3.2, this set-up poses additional challenges, since the task needs to be deduced from a multi-step reasoning chain.

We train the imitation model only on CLEVR versions of the dataset, including both CLEVR-easy (with less camera and scene randomization) and CLEVR-hard (with more camera and scene randomization), and report the success rate in simulation. We further evaluate the performance on real data of the simplest variant of the DROID dataset, using hold-out validation data whose distributions are aligned with the training distributions of CLEVR-easy and CLEVR-hard. Finally, we evaluate generalization to novel objects in the scene on the Objaverse version of the data.

Results are show in Tab. 2. We report a success rate of 70% for the easy version of the dataset and 28% on the harder setup. Furthermore, we observe better results on real data and generalization data for the dataset trained with a harder version of itself, showing that camera and scene randomization are essential for achieving robustness. Visualizations of predicted trajectories and demonstration pairs are available in Appendix F.

5.3 Inference-Time Strategies

In addition to the discussed technical and dataset advances, we also evaluate recent trends in VLMs and their impact on VLAs' performance, namely image cropping and next-token decoding strategies.

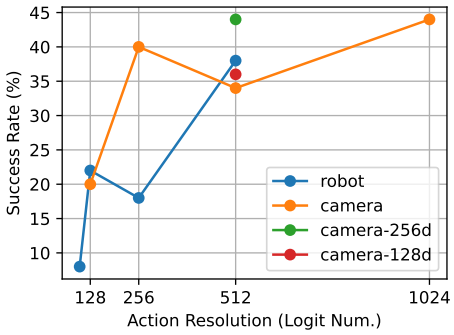


Figure 2: **Action representation ablation.** Comparing robot and image coordinate frame action predictions, success rate on CLEVR-easy simulation, with camera frame performing better on average.

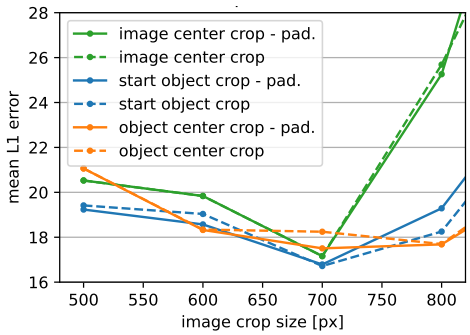
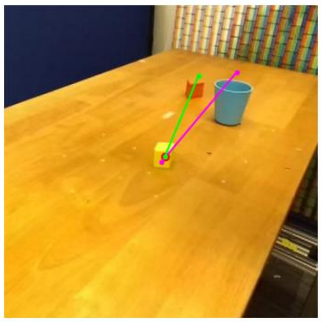
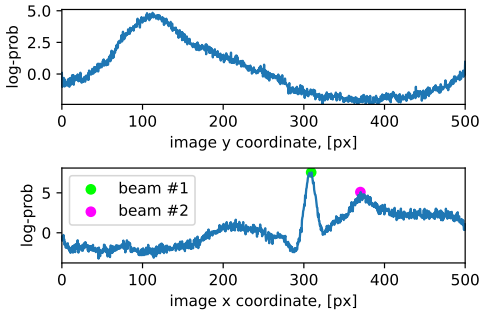


Figure 3: **Cropping strategies comparison.** Cropping can consistently improve performance, but also starts inducing failures, on DROID-hard



(a) Input image with task prompt "`put the yellow block in the cup`".



(b) Logit distribution of the x and y position tokens of the target location.

Figure 4: **Exemplary motivation for decoding.** We qualitatively visualize results on episode 81 of the DROID-hard dataset. The most probable beam corresponds to the red cube, but our proposed NMS-based beam decoding strategy also detects the correct target object location (blue cup).

Input Image Cropping. Accurate object localization is crucial for successful robotic manipulation, as even small errors in identifying object positions can result in task failure. In our system, two factors contribute to sensitivity in localization: (a) the model operates on relatively low-resolution input images (224×224 px), and (b) we predict trajectories in a single step without iterative refinement. As a result, the model's performance is susceptible to the scale of the objects within the image; smaller objects may not be resolved clearly enough to enable precise keyposes prediction. To address this, we investigate the impact of different image cropping strategies. See Fig. 3 for the results and Appendix G for details. Cropping significantly improves performance and is used in subsequent experiments.

Multiple Prediction Generation and Evaluation. In language generation, decoding strategies approximate the most probable token sequence under the model. While greedy decoding is the default, alternative methods can improve quality by identifying high-probability sequences, but at the cost of increased computation. Generating multiple plausible predictions also enables evaluation of both solution accuracy and ability to make diverse predictions.We tested the following standard decoding approaches: Beam search, which keeps n most probable candidate sequences at each decoding step, and sampling, which diversifies the output by sampling from the token probability distribution.

Our contribution is a custom decoding strategy for VLA models, **beam-search-NMS** (Non-Maximum Suppression). We observe that the predicted distribution of VLA models for dense pose tokens behaves differently from sparse language tokens (Fig. 4), they are smooth and have multiple peaks, so

Table 3: **Results for different decoding strategies.** For all methods, we select exactly one prediction for each episode and compute the mean L1 error between the ground truth and predicted trajectories. All methods except greedy are combined with beam search with $n = 3$ beams. The top-3 row shows the lowest error among the predicted beams.

	Greedy search	Sampling	Beam search	Beam search-NMS
Top-1	34.44	34.31	34.17	**33.42**
Top-3	-	33.94	33.94	**25.00**

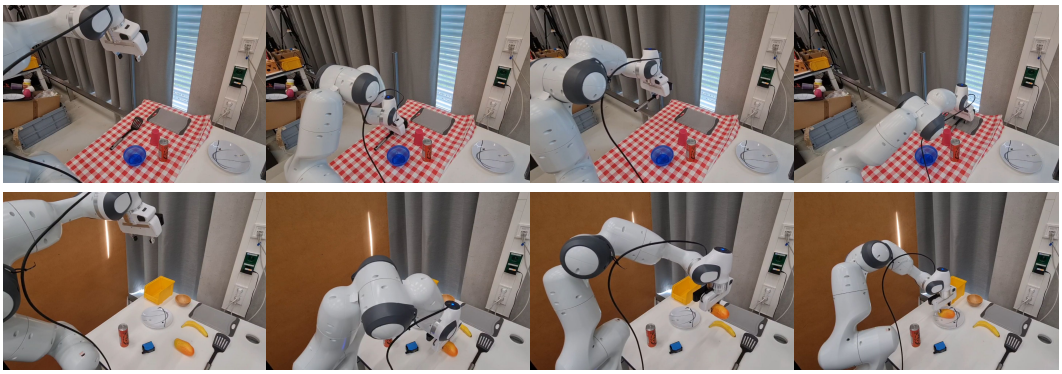


Figure 5: **Real-world demonstration of our approach.** The top row illustrates the task of placing a spatula onto a cutting board, while the bottom row depicts the robot placing a mango onto a plate.

choosing the top k tokens often results in almost the same pose. To find the peaks of the distribution, we do beam search ($n = 3$) and search for local maxima with non-maximum suppression within a window of size $w = 100$, see Appendix H for details and Tab. 3 for results.

To evaluate the distribution of predicted trajectories, we propose using mean Average Precision (mAP)—either with respect to success rate (SR) in simulation as is done in [48, 49] or thresholded Euclidean distance in offline settings. This metric offers a more informative assessment of distributional accuracy than traditional pointwise comparisons such as L1 or L2 distances. For manipulation, we suggest $\text{mAP}_{[0.5\,50]}$, meaning the mAP over L1 distances APs at thresholds of [.5, 1, 2, 5, 10, 20, 50] cm, with 1cm = 10 degrees. AP calculation and curves are shown in Appendix I.

5.4 Real Robot Setup and Experiments

We conduct our experiments using a Franka Panda robotic arm mounted on a mobile base, performing 15 distinct tabletop manipulation tasks involving everyday household items. Our setup supports both external-view cameras and wrist-mounted vision systems. For convenience, we use the wrist-mounted camera, the StereoLabs ZED2i. To ensure our method is robust to different viewpoints, we randomize the robots starting position in each trial. Fig. 5 presents two sample scenarios. Example videos are included in the supplementary.

6 Conclusion

We present an efficient VLA that is trained using image frame coordinates and makes a direct 1-step prediction of two end-effector keyposes. While this system is limited in flexibility and is not a foundation model, it is well-suited to run a wide range of VLA experiments, including ablations, 1-shot imitation, and the applicability of LLM inference strategies to VLA tasks. Finally, we show that our system is applicable in realrobot experiments without any fine-tuning. We believe that it can provide a good foundation for further research into simulation-trained VLMs.

7 Limitations

While our work demonstrates the effectiveness of a lightweight VLA system for keypose prediction in image frame coordinates, it has several limitations that constrain its applicability and generalizability. First, the model is evaluated on a single manipulation task involving small objects and top-down grasps. As such, the learned policy may not transfer well to more diverse tasks, larger objects, or more complex grasping strategies (e.g., side grasps or in-hand manipulation). Second, although we include rotation information in keypose predictions, the system exhibits poor rotation accuracy on real-world data, which limits its effectiveness in tasks that require precise orientation control. Finally, the model is trained and evaluated in simulation with limited real-world testing. This suggests a need for future work on improving robustness and generalization, potentially better data generation. Additionally, broader testing across embodiments and task types would be necessary to establish the scalability and reliability of the proposed system.

Acknowledgments

If a paper is accepted, the final camera-ready version will (and probably should) include acknowledgments. All acknowledgments go at the end of the paper, including thanks to reviewers who gave useful comments, to colleagues who contributed to the ideas, and to funding agencies and corporate sponsors that provided financial support.

References

[1] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. H. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. R. Florence. Palm-e: An embodied multimodal language model. In *International Conference on Machine Learning*, 2023.

[2] D. Ghosh, H. R. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, J. Luo, Y. L. Tan, P. R. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[3] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. R. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[4] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. $\pi 0$: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[5] P. W. Mirowski, M. K. Grimes, M. Malinowski, K. M. Hermann, K. Anderson, D. Teplyashin, K. Simonyan, K. Kavukcuoglu, A. Zisserman, and R. Hadsell. Learning to navigate in cities without a map. *arXiv preprint arXiv:1804.00168*, 2018.

[6] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A platform for embodied ai research. *Conference on Computer Vision (ICCV)*, pages 9338–9346, 2019.

[7] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5, 2020.

[8] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, S. Savarese, and L. Fei-Fei. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning (CoRL)*, 2018.

[9] D. Batra, A. X. Chang, S. Chernova, A. J. Davison, J. Deng, V. Koltun, S. Levine, J. Malik, I. Mordatch, R. Mottaghi, M. Savva, and H. Su. Rearrangement: A challenge for embodied ai. *arXiv preprint arXiv:2011.01975*, 2020.

[10] L. Beyer, A. Steiner, A. S. Pinto, A. Kolesnikov, X. Wang, D. Salz, M. Neumann, I. Alabdulmohsin, M. Tschannen, E. Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

[11] X. Li, P. Li, M. Liu, D. Wang, J. Liu, B. Kang, X. Ma, T. Kong, H. Zhang, and H. Liu. Towards generalist robot policies: What matters in building vision-language-action models. *arXiv preprint arXiv:2412.14058*, 2024.

[12] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Y. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J.-P. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. B. Simpson, Q. U. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Zhao, C. Agia, R. Baijal, M. G. Castro, D. L. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, D. A. Herrera, M. Heo, K. Hsu, J. Hu, M. Z. Irshad, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. M. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Mart'in-Mart'in, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[13] S. Tao, F. Xiang, A. Shukla, Y. Qin, X. Hinrichsen, X. Yuan, C. Bao, X. Lin, Y. Liu, T. kai Chan, Y. Gao, X. Li, T. Mu, N. Xiao, A. Gurha, Z. Huang, R. Calandra, R. Chen, S. Luo, and H. Su. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *arXiv preprint arXiv:2410.00425*, 2024.

[14] R. Xu, J. Zhang, M. Guo, Y. Wen, H. Yang, M. Lin, J. Huang, Z. Li, K. Zhang, L. Wang, Y. Kuang, M. Cao, F. Zheng, and X. Liang. A0: An affordance-aware hierarchical model for general robotic manipulation. *arXiv preprint arXiv:2504.12636*, 2025.

[15] R. Zheng, Y. Liang, S. Huang, J. Gao, H. Daumé III, A. Kolobov, F. Huang, and J. Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *arXiv preprint arXiv:2412.10345*, 2024.

[16] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. J. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, J. Jang, Z. Jiang, J. Kautz, K. Kundalia, L. Lao, Z. Li, Z. Lin, K. Lin, G. Liu, E. Llontop, L. Magne, A. Mandlekar, A. Narayan, S. Nasiriany, S. Reed, Y. L. Tan, G. Wang, Z. Wang, J. Wang, Q. Wang, J. Xiang, Y. Xie, Y. Xu, Z. Xu, S. Ye, Z. Yu, A. Zhang, H. Zhang, Y. Zhao, R. Zheng, and Y. Zhu. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[17] M. Deitke, C. Clark, S. Lee, R. Tripathi, Y. Yang, J. S. Park, M. Salehi, N. Muennighoff, K. Lo, L. Soldaini, J. Lu, T. Anderson, E. Bransom, K. Ehsani, H. Ngo, Y. Chen, A. Patel, M. Yatskar, C. Callison-Burch, A. Head, R. Hendrix, F. Bastani, E. VanderBilt, N. Lambert, Y. Chou, A. Chheda, J. Sparks, S. Skjonsberg, M. Schmitz, A. Sarnat, B. Bischoff, P. Walsh, C. Newell, P. Wolters, T. Gupta, K.-H. Zeng, J. Borchardt, D. Groeneveld, J. Dumas, C. Nam, S. Lebrecht, C. M. Wittlif, C. Schoenick, O. Michel, R. Krishna, L. Weihs, N. A. Smith, H. Hajishirzi, R. Girshick, A. Farhadi, and A. Kembhavi. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models. *arXiv preprint arXiv:2409.17146*, 2024.

[18] W. Yuan, J. Duan, V. Blukis, W. Pumacay, R. Krishna, A. Murali, A. Mousavian, and D. Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics. *arXiv preprint arXiv:2406.10721*, 2024.

[19] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning (CoRL)*, pages 785–799, 2023.

[20] X. Ma, S. Patidar, I. Haughton, and S. James. Hierarchical diffusion policy for kinematics-aware multi-task robotic manipulation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18081–18090, 2024.

[21] Y. Yang, Z. Cai, Y. Tian, J. Zeng, and J. Pang. Gripper keypose and object pointflow as interfaces for bimanual robotic manipulation. *arXiv preprint arXiv:2504.17784*, 2025.

[22] Y. Zhong, X. Huang, R. Li, C. Zhang, Y. Liang, Y. Yang, and Y. Chen. Dexgraspvla: A vision-language-action framework towards general dexterous grasping. *arXiv preprint arXiv:2502.20900*, 2025.

[23] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *Robotics and Automation Letters (RA-L)*, 7(3):7327–7334, 2022.

[24] S. James, Z. Ma, D. Rovick Arrojo, and A. J. Davison. Rlbench: The robot learning benchmark & learning environment. *Robotics and Automation Letters (RA-L)*, 2020.

[25] D. Niu, Y. Sharma, G. Biamby, J. Quenum, Y. Bai, B. Shi, T. Darrell, and R. Herzig. LLARVA: Vision-action instruction tuning enhances robot learning. In *Conference on Robot Learning (CoRL)*, 2024.

[26] S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, T. Armstrong, A. Balakrishna, R. Baruch, M. Bauza, M. Blokzijl, S. Bohez, K. Bousmalis, A. Brohan, T. Buschmann, A. Byravan, S. Cabi, K. Caluwaerts, F. Casarini, O. Chang, J. E. Chen, X. Chen, H.-T. L. Chiang, K. Choromanski, D. D'Ambrosio, S. Dasari, T. Davchev, C. Devin, N. D. Palo, T. Ding, A. Dostmohamed, D. Driess, Y. Du, D. Dwibedi, M. Elabd, C. Fantacci, C. Fong, E. Frey, C. Fu, M. Giustina, K. Gopalakrishnan, L. Graesser, L. Hasenclever, N. Heess, B. Hernaez, A. Herzog, R. A. Hofer, J. Humplik, A. Iscen, M. G. Jacob, D. Jain, R. Julian, D. Kalashnikov, M. E. Karagozler, S. Karp, C. Kew, J. Kirkland, S. Kirmani, Y. Kuang, T. Lampe, A. Laurens, I. Leal, A. X. Lee, T.-W. E. Lee, J. Liang, Y. Lin, S. Maddineni, A. Majumdar, A. H. Michaely, R. Moreno, M. Neunert, F. Nori, C. Parada, E. Parisotto, P. Pastor, A. Pooley, K. Rao, K. Reymann, D. Sadigh, S. Saliceti, P. Sanketi, P. Sermanet, D. Shah, M. Sharma, K. Shea, C. Shu, V. Sindhwani, S. Singh, R. Soricut, J. T. Springenberg, R. Sterneck, R. Surdulescu, J. Tan, J. Tompson, V. Vanhoucke, J. Varley, G. Vesom, G. Vezzani, O. Vinyals, A. Wahid, S. Welker, P. Wohlhart, F. Xia, T. Xiao, A. Xie, J. Xie, P. Xu, S. Xu, Y. Xu, Z. Xu, Y. Yang, R. Yao, S. Yaroshenko, W. Yu, W. Yuan, J. Zhang, T. Zhang, A. Zhou, and Y. Zhou. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

[27] H. Zhen, X. Qiu, P. Chen, J. Yang, X. Yan, Y. Du, Y. Hong, and C. Gan. 3d-vla: A 3d vision-language-action generative world model. *arXiv preprint arXiv:2403.09631*, 2024.

[28] X. Li, K. Hsu, J. Gu, K. Pertsch, O. Mees, H. R. Walke, C. Fu, I. Lunawat, I. Sieh, S. Kirmani, S. Levine, J. Wu, C. Finn, H. Su, Q. H. Vuong, and T. Xiao. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

[29] Z. Wang, Z. Zhou, J. Song, Y. Huang, Z. Shu, and L. Ma. Towards testing and evaluating vision-language-action models for robotic manipulation: An empirical study. *arXiv preprint arXiv:2409.12894*, 2024.

[30] H. Arai, K. Miwa, K. Sasaki, Y. Yamaguchi, K. Watanabe, S. Aoki, and I. Yamamoto. Covla: Comprehensive vision-language-action dataset for autonomous driving. *Winter Conference on Applications of Computer Vision (WACV)*, pages 1933–1943, 2024.

[31] C. Chen, M. Pourkeshavarz, and A. Rasouli. Criteria: a new benchmarking paradigm for evaluating trajectory prediction models for autonomous driving. *Conference on Robotics and Automation (ICRA)*, pages 8265–8271, 2023.

[32] Y. Duan, M. Andrychowicz, B. Stadie, O. Jonathan Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba. One-shot imitation learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

[33] S. James, M. Bloesch, and A. J. Davison. Task-embedded control networks for few-shot imitation learning. *Conference on Robot Learning (CoRL)*, 2018.

[34] M. Du, S. Nair, D. Sadigh, and C. Finn. Behavior retrieval: Few-shot imitation learning by querying unlabeled datasets. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[35] L.-H. Lin, Y. Cui, A. Xie, T. Hua, and D. Sadigh. Flowretrieval: Flow-guided data retrieval for few-shot imitation learning. In *Conference on Robot Learning (CoRL)*, 2024.

[36] X. Zhang and A. Boularias. One-shot imitation learning with invariance matching for robotic manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[37] N. Heppert, M. Argus, T. Welschehold, T. Brox, and A. Valada. Ditto: Demonstration imitation by trajectory transformation. In *Conference on Intelligent Robots and Systems (IROS)*, 2024.

[38] N. Di Palo and E. Johns. Keypoint action tokens enable in-context imitation learning in robotics. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[39] L. Fu, H. Huang, G. Datta, L. Y. Chen, W. C.-H. Panitch, F. Liu, H. Li, and K. Goldberg. In-context imitation learning via next-token prediction. *arXiv preprint arXiv:2408.15980*, 2024.

[40] A. Steiner, A. S. Pinto, M. Tschannen, D. Keysers, X. Wang, Y. Bitton, A. Gritsenko, M. Minderer, A. Sherbondy, S. Long, S. Qin, R. R. Ingle, E. Bugliarello, S. Kazemzadeh, T. Mesnard, I. M. Alabdulmohsin, L. Beyer, and X.-Q. Zhai. Paligemma 2: A family of versatile vlms for transfer. *arXiv preprint arXiv:2412.03555*, 2024.

[41] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. C. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. S. Ryoo, G. Salazar, P. R. Sanketi, K. Sayed, J. Singh, S. A. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. H. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[42] J. Ning, C. Li, Z. Zhang, Z. Geng, Q. Dai, K. He, and H. Hu. All in tokens: Unifying output space of visual tasks via soft token. *Conference on Computer Vision (ICCV)*, pages 19843–19853, 2023.

[43] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi. Objaverse: A universe of annotated 3d objects. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13142–13153, 2023.

[44] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. B. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1988–1997, 2016.

[45] K. Ehsani, T. Gupta, R. Hendrix, J. Salvador, L. Weihs, K.-H. Zeng, K. P. Singh, Y. Kim, W. Han, A. Herrasti, et al. Spoc: Imitating shortest paths in simulation enables effective navigation and manipulation in the real world. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16238–16250, 2024.

[46] K. Greff, F. Belletti, L. Beyer, C. Doersch, Y. Du, D. Duckworth, D. J. Fleet, D. Gnanapragasam, F. Golemo, C. Herrmann, T. Kipf, A. Kundu, D. Lagun, I. H. Laradji, H.-T. Liu, H. Meyer, Y. Miao, D. Nowrouzezahrai, C. Oztireli, E. Pot, N. Radwan, D. Rebain, S. Sabour, M. S. M. Sajjadi, M. Sela, V. Sitzmann, A. Stone, D. Sun, S. Vora, Z. Wang, T. Wu, K. M. Yi, F. Zhong, and A. Tagliasacchi. Kubric: A scalable dataset generator. *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3739–3751, 2022.

[47] A. Quattoni and A. Torralba. Recognizing indoor scenes. *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 413–420, 2009.

[48] H. Fang, C. Wang, H. Fang, M. Gou, J. Liu, H. Yan, W. Liu, Y. Xie, and C. Lu. Anygrasp: Robust and efficient grasp perception in spatial and temporal domains. *Transactions on Robotics (T-RO)*, 39:3929–3945, 2022.

[49] H. Zhou, D. Blessing, G. Li, O. Celik, X. Jia, G. Neumann, and R. Lioutikov. Variational distillation of diffusion policies into mixture of experts. *arXiv preprint arXiv:2406.12538*, 2024.

[50] M. Tschannen, A. Gritsenko, X. Wang, M. F. Naeem, I. M. Alabdulmohsin, N. Parthasarathy, T. Evans, L. Beyer, Y. Xia, B. Mustafa, O. H'enaff, J. Harmsen, A. Steiner, and X.-Q. Zhai. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

[51] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*, pages 740–755. Springer, 2014.

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 cm = 1**°. The overall pose L1 error is given by:

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

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.
- $R_i, \hat{R}_i \in \mathrm{SO}(3)$ 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 - this quantity. The resulting rewards are clipped to the unit range. Success rates are computed using a reward threshold of 0.75.

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

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

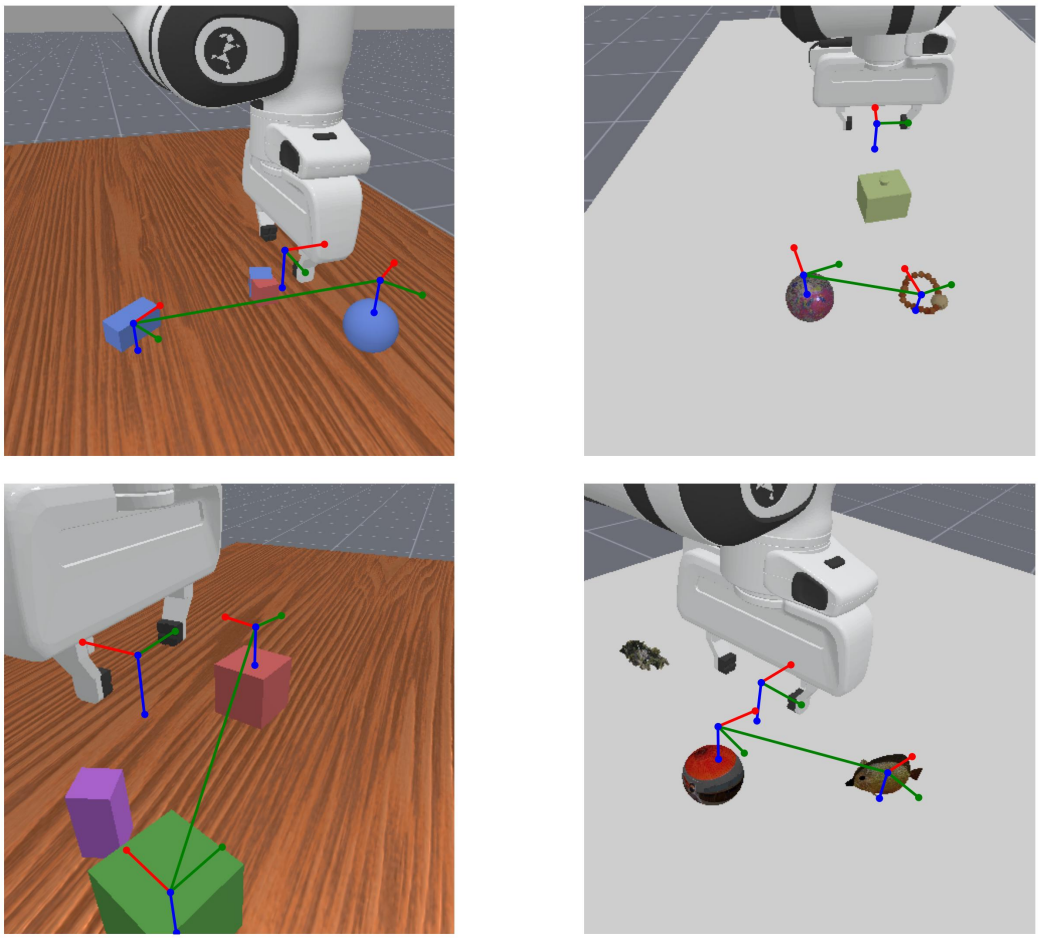
C Objaverse Asset Curation

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.

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

D Simulation and Real Experiments

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



(a) CLEVR assets (b) 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.

We evaluated our models on four different application domains - simulated data, real data, simulations, and real robot setups. Evaluations on simulated data and simulations were performed in the same 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>` $\rightarrow$ `<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.

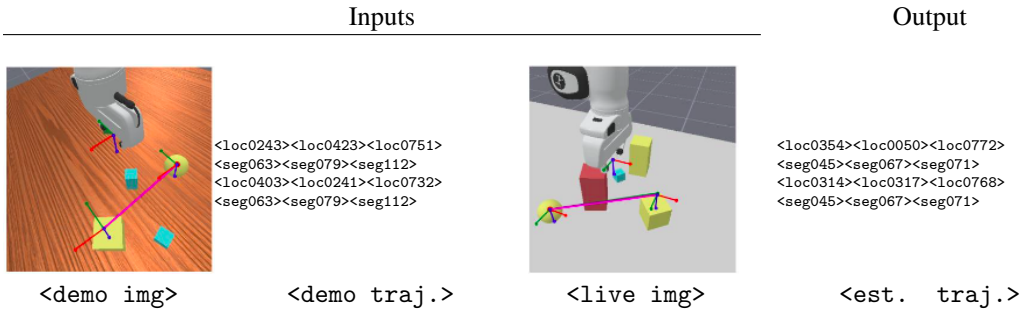


Figure 7: **One-shot prompt example.** Prompts consist of a demonstration image-trajectory path and a live image for which the model should predict the trajectory of the task represented with demonstration data. No language description is provided, yet the model is able to imitate the task. The given 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.

G Input Image Cropping

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

H Beam-Search-NMS Implementation

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

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

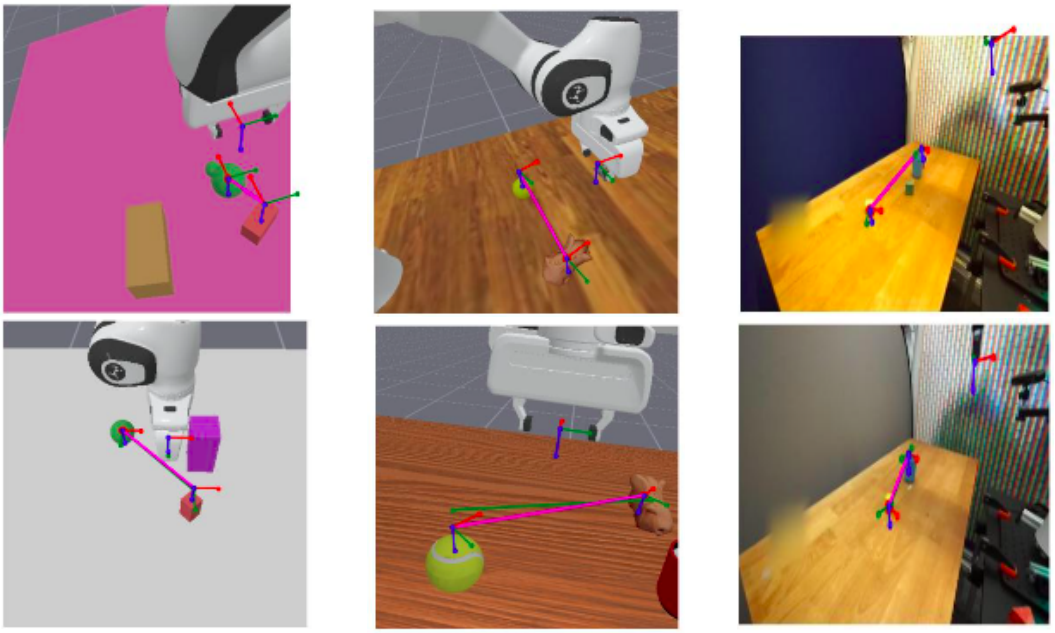
I mAP Calculation

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

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

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

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

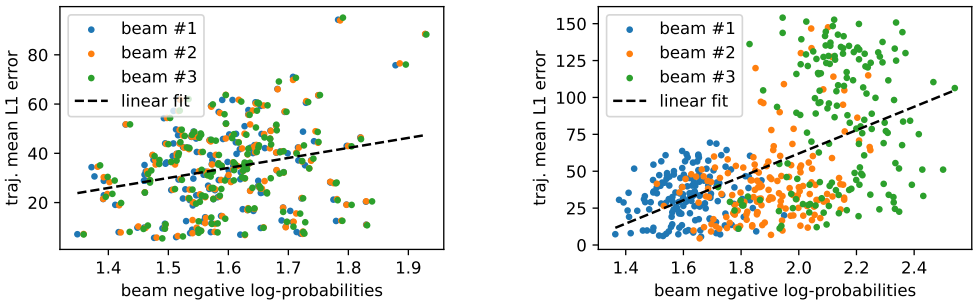


(a) **CLEVR-hard:** move large green sphere onto small red box

(b) **Objaverse-easy:** move terra-cotta animal onto yellow ball

(c) **DROID-easy:** put the yellow block inside the light blue cup

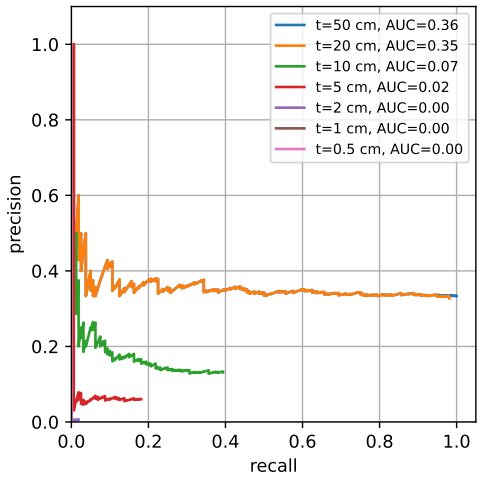
Figure 8: **Demonstration and live image with predictions.** Examples of demonstration image in the first row with visualized ground truth trajectory (pink line) and live image with both ground truth (green line) and predicted trajectory (pink line) in the second row for three different datasets - simulation 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.



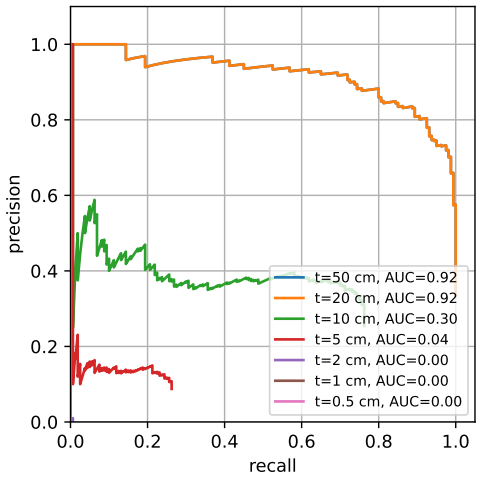
a) Standard beam search

b) Beam Search-NMS (ours)

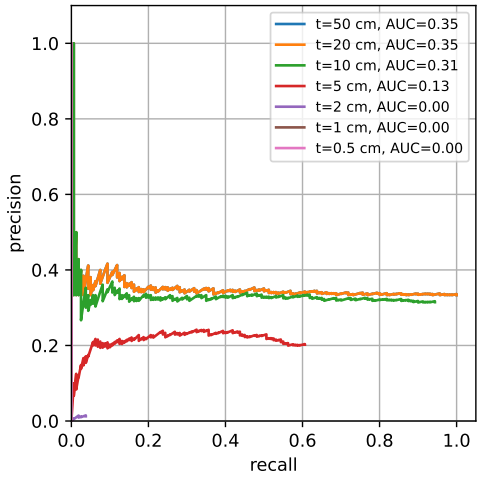
Figure 9: **Beam log-pobs correlate with L1 error.** Higher negative log-prob means lower confidence. Existence of correlation allows us to use log-probs as confidences. Results show an exploration-exploitation tradeoff. Standard beam search generates low error samples, these predictions are not very 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.



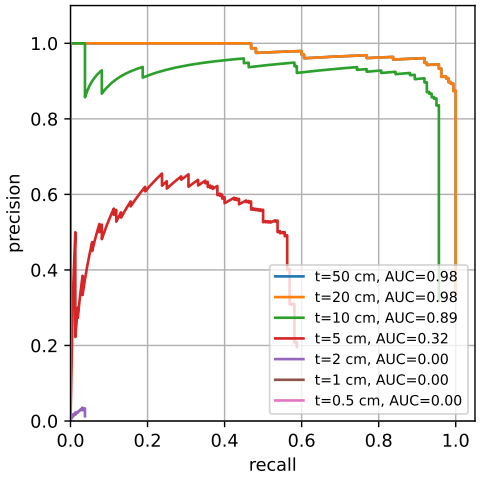
(a) Standard beam search, no crop, mAP=0.11



(b) Beam search-NMS (ours), no crop, mAP=0.31



(c) Standard beam search, crop size 700, mAP=0.16



(d) Beam search-NMS (ours), crop size 700, mAP=0.45

Figure 10: Precision-recall curves at different error thresholds. Upper row – no crop, lower row – crop size 700 with padding. Object sizes are around 10 cm, thus we suggest $\text{mAP}_{[0.5\ 50]}$, meaning the mAP at thresholds of [.5, 1, 2, 5, 10, 20, 50] cm, with 1cm = 10 degrees.