

# GraspVLA: a Grasping Foundation Model

## Pre-trained on Billion-scale Synthetic Action Data

Anonymous Author(s)

Affiliation

Address

email

### A Details about SynGrasp-1B

We present the statistics of our synthetic dataset, SynGrasp-1B, in Table 1. The dataset consists of 10 million trajectories, each containing approximately 100 frames, resulting in a total of 1 billion frames—a substantial increase in scale compared to existing open-source datasets. Our dataset encompasses a diverse array of object categories, featuring 10,680 objects across 240 categories. While real-world datasets often encounter challenges related to scene diversity and require collaboration among laboratories across different countries, synthetic data generation enables us to easily create varied scenes by altering the textures of the table, ground, and walls. We utilize around 1,000 different textures for the table and 1,200 for the ground and walls, leading to a total of 1 million unique scenes.

Unlike existing datasets, SynGrasp-1B is the first to offer precise and fine-grained annotations for camera calibration, bounding box annotations, and the 3D poses of both the target object and the gripper. Thanks to our simulation engine, we can effortlessly obtain these annotations and incorporate additional types, such as depth maps and segmentation masks, when necessary. This flexibility is a significant advantage of synthetic datasets over their real-world counterparts.

|                       | Trajectories | Objects | Scenes | Camera Calibration | Bbox Annotation | 3D Pose Annotation |
|-----------------------|--------------|---------|--------|--------------------|-----------------|--------------------|
| RoboSet [1]           | 98k          | < 200   | 11     | ✗                  | ✗               | ✗                  |
| BridgeData V2 [2]     | 60k          | 100     | 24     | ✗                  | ✗               | ✗                  |
| RT-1 [3]              | 130k         | < 200   | 2      | ✗                  | ✗               | ✗                  |
| DROID [4]             | 76k          | < 200   | 2080   | ✓                  | ✗               | ✗                  |
| AgiBot World [5]      | 1M           | 3k      | 106    | ✓                  | ✗               | ✗                  |
| Open-X Embodiment [6] | 1.4M         | -       | 311    | ✗                  | ✗               | ✗                  |
| SynGrasp-1B           | 10M          | 10k     | 1M     | ✓                  | ✓               | ✓                  |

Table 1: **Comparison of SynGrasp-1B with Existing Datasets for Robot Manipulation.** This table highlights the advantages of SynGrasp-1B in terms of scale, annotations, and scene diversity compared to other datasets.

The generation of simulation data is also much more cost-effective than real-world data collection, considering factors such as time, financial resources, space, robotic equipment, and human labor:

- **Time:** A single human operator can collect only around 1,000 trajectories per day; scaling to 10 million would require 100 operators working for 100 days. In contrast, we can generate 10 million trajectories in 10 days using 160 NVIDIA 4090 GPUs. This efficiency accelerates the data-model feedback loop, enabling rapid model iteration and performance improvements.
- **Space:** Real-world data collection often necessitates multiple laboratories across different countries to enhance scene diversity. Additionally, it requires significant physical space to accommodate robots and objects; for example, the AgiBot World dataset [5] utilizes a 4,000  $m^2$  area for data collection. In contrast, our synthetic approach does not require any physical space.

- 28 • **Robots:** Each human operator in real-world collection needs a physical robot, resulting in  
29 high costs and maintenance overhead.
- 30 • **Money:** The total cost of generating SynGrasp-1B is around \$5,000—orders of magnitude  
31 cheaper than real-world alternatives.

32 As an initial step toward large-scale synthetic action pre-training, we focus on grasping tasks to  
33 enable detailed analysis. Our pipeline can be extended to other robotic arms (for cross-embodiment  
34 transfer) or tasks (e.g., placing, pushing, stacking), as well as large-scale camera randomization. We  
35 leave these extensions to future work.

36 **Gallery.** We randomly sample 24 trajectories from our SynGrasp-1B and visualize them in Fig. 1.  
37 Each trajectory consists of around 100 frames, and we uniformly sample 4 frames from each trajec-  
38 tory for visualization.

## 39 B Details about Data Generation

40 **Asynchronous and Grouped Data Writing.** We employ DeepMind EnvLogger with TFDS back-  
41 end [7] for the storage of our synthetic data. Despite its clean design, considerations for high-  
42 performance large-scale simulation are necessary. Firstly, to mask the time cost of image encoding  
43 and data writing, we modified the EnvLogger implementation to perform the actual data writing  
44 operation asynchronously. Second, to avoid contention on the dataset metadata across parallel pro-  
45 cesses and to minimize data loss caused by unforeseen errors (e.g., GPU failures), the processes  
46 should not write to a single shared folder. However, if each simulation instance utilizes a unique  
47 subfolder, it results in a large number of subfolders and metadata, leading to substantial overheads  
48 in data management, transfer, and loading. As a compromise, we assign each process a subfolder  
49 with random UUIDs [8], and write all the trajectories within a process to the same subfolder.

50 **Handling Data Corruption.** At billion scale, it is challenging to ensure the consistency and valid-  
51 ity of the whole dataset with reasonable overhead. As a result, we postpone the handling of data  
52 corruption and data loss to training phase. Specifically, during the dataset loading process, we care-  
53 fully handle the FailedPreconditionError, DataLossError, and NotFoundError exceptions raised by  
54 TFDS. Upon encountering a NotFoundError, we create an empty file at the expected file path, other-  
55 wise TFDS cannot continue loading the remaining records within the subfolder. On DataLossError,  
56 we count it as one missing record, and log the missing rate as a critical statistics for data validity.  
57 The missing rate was below 1%. FailedPreconditionError is raised when the successfully loaded  
58 number of records is smaller than that in the metadata of the subfolder, and can be safely converted  
59 to a StopIteration to facilitate the correct functioning of the data loader. The above handling of these  
60 exceptions ensures loading all the valid records. We hope that sharing such experiences would help  
61 future exploration.

62 **Object Processing and Layout Generation.** To ensure that the scales of synthetic objects align with  
63 real-world counterparts and are suitable for grasping, we manually define minimum and maximum  
64 size constraints for each category. Furthermore, we simplify the object meshes using the ACVD  
65 algorithm [9] to improve the simulation efficiency.

66 To enhance data diversity, we create different clutter layouts for each episode by randomly placing  
67 objects within a 0.4m by 0.5m area on a table. Objects are dropped in various poses to generate  
68 physically plausible scenes. For categories requiring specific orientations, such as cups, we manually  
69 define valid poses (e.g., upright).

70 **Details about rendering.** To facilitate the PAG mechanism, we also save the bounding boxes of  
71 objects from both viewpoints at each time step during synthetic data generation. Additionally, the  
72 camera intrinsics are set to closely match a real D435 camera. During real-world deployment, we  
73 apply a rectification process to accommodate any slight differences in the intrinsic parameters of  
74 actual cameras.



Figure 1: **Gallery of SynGrasp-1B.** 24 randomly sampled trajectories from our synthetic grasping data. For clarity, 4 frames are uniformly sampled from each trajectory for display.

## 75 C Details of Main Experiments

76 **Metrics.** In each trial, the model is allowed to attempt to grasp up to three times, with each attempt  
 77 counted by the gripper closure action. Success is strictly defined as the specified object being lifted  
 78 a minimum of 15 cm. The scene is not reset during each trial, even if the model knocks the object  
 79 off the table.

80 Additionally, we introduce Success weighted by Path Length (SPL) to further account for the number  
 81 of actions taken, which is a common metric in discrete navigation tasks to evaluate the efficiency  
 82 of the model. While for discrete navigation tasks, the shortest path is easy to define, for grasping  
 83 tasks, the shortest path is not well defined. Therefore, for each trial, if there are several methods  
 84 that can successfully grasp the object, we define the shortest path as the one with the least number  
 85 of action steps. If all methods fail to grasp the object, they all get zero SPL in this trial. Note that,  
 86 our SynGrasp-1B dataset stores actions in 10 Hz and all methods are trained on this dataset, so the  
 87 number of action steps is comparable across methods.

Table 2: Hyperparameters for all the methods

| Baseline         | Hyperparameter | Value           |
|------------------|----------------|-----------------|
| GraspVLA         | batch_size     | 384             |
|                  | learning_rate  | 1.6e-4          |
| $\pi_0$          | batch_size     | 256             |
|                  | learning_rate  | cosine schedule |
|                  | warmup_steps   | 1000            |
|                  | peak_lr        | 2.5e-5          |
|                  | decay_lr       | 2.5e-6          |
|                  | decay_steps    | 30000           |
| OpenVLA          | lora_rank      | 32              |
|                  | batch_size     | 12              |
|                  | learning_rate  | 5e-4            |
|                  | image_aug      | true            |
| Octo             | batch_size     | 256             |
|                  | learning_rate  | rsqrt schedule  |
|                  | warmup_steps   | 2000            |
|                  | init_value     | 0.0             |
|                  | peak_value     | 3e-4            |
| Diffusion Policy | batch_size     | 256             |
|                  | learning_rate  | cosine schedule |
|                  | warmup_steps   | 500             |
|                  | peak_lr        | 1e-4            |
|                  | weight_decay   | 1e-6            |

88 **Baselines.** As Diffusion Policy does not support language conditioning, we train it using the subset  
 89 of SynGrasp-1B that grasps the elephant (around 40k trajectories), and replace the target object with  
 90 the elephant in the real-world experiments. We train all other models with the full SynGrasp-1B  
 91 dataset. We train  $\pi_0$  [10] with its pre-trained weights initialization and PaliGemma initialization for  
 92 comparison. Since OpenVLA [11] takes a single RGB image for visual observation, we use the front  
 93 camera view for it. For Diffusion Policy [12], we train the UNet-based version as recommended  
 94 by the original paper. For Octo [13], we finetune the pre-trained *octo-base-1.5* model. All the  
 95 models are trained with action chunks of 4 [14], except for OpenVLA, which does not support action  
 96 chunking. We provide hyperparameters of training/finetuning GraspVLA and baselines in Table 2.  
 97 We run automatic evaluation in our simulation pipeline for all the baselines, continue training until  
 98 the success rate converges, and select the best-performing checkpoint in the real world. We found  
 99 GraspVLA achieves a consistently high success rate after 120k steps.



**Real world setup and modification to robot finger.** For perception, we employ an Intel RealSense D435 as the front-facing camera and a D415i as the side-facing camera. Both cameras are positioned at the center of the randomization range used in the synthetic data generation. The workspace for test objects is confined to a 40 cm x 50 cm x 20 cm area in front of the robot.

The original Franka Panda finger is too short to firmly grasp convex-shaped objects (e.g., a bottle lying on its side). This is because the hand plank collides with the top of the object, preventing the fingers from reaching deep enough to secure a stable grip. To address this issue, we extended the fingers by 2 cm in both synthetic data generation and real-world experiments.

## D Detailed Scaling Law

**Simulation evaluation.** While the main paper analyzes the scaling law in real-world, we extend this analysis to simulation environments. Our results show that simulation is an effective proxy for predicting real-world performance. For these experiments, we use simulation environments with identical camera and table configurations to our data generation setup, but employ different object instances and materials to assess generalizability.

As shown in Fig. 2a), GraspVLA’s performance on simulation data follows a scaling trend similar to that of real-world data, confirming the simulation’s effectiveness for predicting real-world performance. However, we observe two key differences: (1) real-world performance scales more slowly (0.12B) compared to simulation, where performance saturates earlier, and (2) the sim-to-real gap decreases with more training frames, suggesting that larger datasets enable more robust representations and better transfer to real-world scenarios.

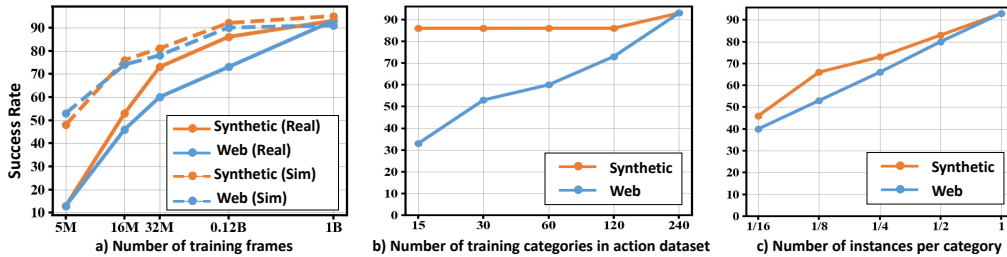


Figure 2: **Scaling laws different training regimes.** (a) Performance scaling with number of training frames in both simulation and real-world environments. (b) Impact of training category diversity while fixing instances per category. (c) Effect of varying instances per category while maintaining total category count.

We further investigate how data diversity affects GraspVLA’s performance by analyzing two additional scaling factors: (1) the number of training categories and (2) the number of instances per category. For each analysis, we hold the other factor and total training frames constant.

**Number of Training Categories (Fig. 2b).** When varying the number of categories while fixing instances per category and total frames, performance on web categories improves steadily with more training categories, whereas performance on synthetic categories saturates early. This implies that inter-category generalization (adapting to unseen categories) benefits significantly from broader categorical coverage, while intra-category generalization (recognizing diverse instances of known categories) requires less diversity.

**Number of Instances per Category (Fig. 2c).** With a fixed category count and total frames, increasing instances per category leads to consistent improvements across both synthetic and web categories. This underscores the importance of instance diversity within categories for robust generalization.

## E Details about Experiments on LIBERO Benchmark

**Setup.** We consider a trial successful if the robot successfully grasps and lifts the target object to a height of 10 cm. Since our model is trained with two camera views, we modify the original camera configurations provided by the LIBERO benchmark to match our training setup, aligning the camera poses accordingly. Additionally, we extend the gripper by 2 cm, as detailed in the robot finger modification in Section C. These adjustments are made exclusively for evaluating our model to ensure they do not affect the fine-tuned baselines, which are evaluated using the original camera configurations and gripper length.

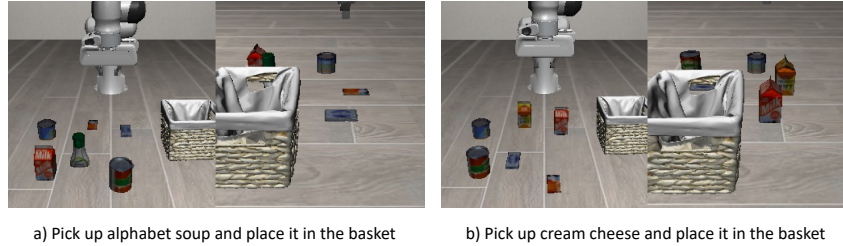


Figure 3: **Examples of LIBERO Benchmark.** We visualize both front and side views side by side.

The LIBERO-object test set presents a significant challenge for zero-shot models due to ambiguous target object descriptions. As illustrated in Figure 3, even humans may struggle to identify objects like "alphabet soup" and "cream cheese" in the given scene. To account for this, we relax the success criteria: a trial is deemed successful if the robot grasps any object belonging to the same category as the target. For instance, if the target is "alphabet soup," grasping any object in the "can" category is considered a success. Similarly, if the target is "cream cheese," grasping any object in the "box" category qualifies as success.

As noted in the main paper, we exclude non-prehensile tasks to focus solely on grasping capability. We also omit tasks requiring color-based distinctions (e.g., "pick up the yellow and white mug"), as reasoning about color falls outside our scope. The specific tasks deemed invalid in the original test set, along with the modified instructions, are detailed in Tables 3, 4, and 5.

Table 3: **Modification to LIBERO-Goal test set.**

| Original Caption                            | Valid | Modified Caption         |
|---|-------|--------------------------|
| put the wine bottle on top of the cabinet   | ✓     | pick up the wine bottle  |
| open the top drawer and put the bowl inside | ✓     | pick up the bowl         |
| turn on the stove                           | x     | -                        |
| put the bowl on top of the cabinet          | ✓     | pick up the bowl         |
| put the bowl on the plate                   | ✓     | pick up the bowl         |
| put the wine bottle on the rack             | ✓     | pick up the wine bottle  |
| put the cream cheese in the bowl            | ✓     | pick up the cream cheese |
| open the middle drawer of the cabinet       | x     | -                        |
| push the plate to the front of the stove    | x     | -                        |
| put the bowl on the stove                   | ✓     | pick up the bowl         |

**Impact of instruction format.** While the original test sets in LIBERO mainly compose of two steps, picking up an object and placing it in a container, we focus on the first step to exclusively evaluate the grasping capabilities. Therefore, to ensure a fair comparison, we also simplify the original instruction format "pick up a object and place it in a container" to "pick up a object" for the same instructions across all models. Note that, the instructions in the fine-tuning set are not simplified due to difficulties in segmenting and removing actions related to placing objects in containers.

As shown in Table 6, the performance of both fine-tuned baselines drops significantly when the instruction format is simplified. This indicates that the models are not robust to instruction variations and are sensitive to the specific instruction format. Additionally, our zero-shot model, GraspVLA, outperforms the fine-tuned models in the simplified instruction format and achieves comparable

Table 4: **Modification to LIBERO-Object test set.**

| Original Caption   | Valid | Modified Caption              |
|--|-------|-------------------------------|
| pick up the alphabet soup and place it in the basket     | ✓     | pick up the alphabet soup     |
| pick up the cream cheese and place it in the basket      | ✓     | pick up the cream cheese      |
| pick up the milk and place it in the basket              | ✓     | pick up the milk              |
| pick up the tomato sauce and place it in the basket      | ✓     | pick up the tomato sauce      |
| pick up the butter and place it in the basket            | ✓     | pick up the butter            |
| pick up the orange juice and place it in the basket      | ✓     | pick up the orange juice      |
| pick up the chocolate pudding and place it in the basket | ✓     | pick up the chocolate pudding |
| pick up the bbq sauce and place it in the basket         | ✓     | pick up the bbq sauce         |
| pick up the ketchup and place it in the basket           | ✓     | pick up the ketchup           |
| pick up the salad dressing and place it in the basket    | ✓     | pick up the salad dressing    |

Table 5: **Modification to LIBERO-Long test set.**

| Original Caption  | Valid | Modified Caption             |
|---|-------|------------------------------|
| turn on the stove and put the moka pot on it  | ✓     | pick up the moka pot         |
| put the black bowl in the bottom drawer of the cabinet and close it                     | ✓     | pick up the black bowl       |
| put the yellow and white mug in the microwave and close it                              | x     | -                            |
| put both moka pots on the stove   | ✓     | pick up the moka pot         |
| put both the alphabet soup and the cream cheese box in the basket                       | ✓     | pick up the cream cheese box |
| put both the alphabet soup and the tomato sauce in the basket                           | ✓     | pick up the alphabet soup    |
| put both the cream cheese box and the butter in the basket                              | ✓     | pick up the cream cheese box |
| put the white mug on the left plate and put the yellow and white mug on the right plate | x     | -                            |
| put the white mug on the plate and put the chocolate pudding to the right of the plate  | x     | -                            |
| pick up the book and place it in the back compartment of the caddy                      | ✓     | pick up the book             |

162 performance in the original instruction format. This demonstrates the robustness of our model to  
 163 generalize to unseen environments, even in the absence of fine-tuning.

|   | Long        | Goal        | Object      |
|---|-------------|-------------|-------------|
| <i>Format: pick up {object} and place it in {container}</i> |             |             |             |
| OpenVLA (fine-tuned)  | 70.9        | 78.6        | 91.2        |
| $\pi_0$ (fine-tuned)  | <b>88.7</b> | <b>95.4</b> | <b>98.4</b> |
| <i>Format: pick up {object}</i>                             |             |             |             |
| OpenVLA (fine-tuned)  | 33.7        | 56.6        | 65.4        |
| $\pi_0$ (fine-tuned)  | 62.7        | 79.4        | 93.8        |
| Ours (zero-shot)  | <b>82.0</b> | <b>91.2</b> | <b>94.1</b> |

Table 6: **Impact of instruction format.** Fine-tuned baselines exhibit performance drops when the original instructions are simplified.

## 164 F Details about Comparison with AnyGrasp

165 **Setup.** To ensure a fair comparison, we run the AnyGrasp baseline with up to three attempts per  
 166 trial, counting it as a success if the object is grasped in any attempt. The baseline is implemented  
 167 using the authors’ official SDK. For perception, we use the same Franka Emika Panda robot and a  
 168 RealSense D435i camera mounted on the end-effector, with the camera calibrated for accurate depth  
 169 perception. Inference speed is evaluated on an NVIDIA RTX 3090 GPU.

170 For the language-conditioned test set, we integrate Grounding DINO [15] to parse language in-  
 171 structions into bounding boxes. Grasp candidates whose 2D projections fall outside these boxes are  
 172 filtered out. Given the sparse layout, this simple approach effectively eliminates irrelevant grasps.  
 173 Motion planning is then used to generate trajectories for execution.

174 **Test Sets.** The language-driven task uses the same test set as in the main experiment (Table 1 in  
 175 the main paper), comprising both synthetic and web categories for a total of 60 trials. For arbitrary  
 176 grasping of common objects, we randomly select 30 objects (15 synthetic, 15 web), ensuring diffuse,  
 177 non-reflective materials (e.g., rubber, wood). The transparent object test set consists of 5 objects,  
 178 including 3 bottles, 1 cup, and 1 bowl. To focus on grasping transparent objects, we remove distrac-

179 tors from the scene and place the transparent objects at 6 different poses on the table, resulting in 6  
 180 trials per object. We visualize transparent objects in Figure 4.



Figure 4: **Transparent objects used for evaluation.**

181 **Analysis.** In the language-conditioned test set, the baseline fails in 5 trials. Three failures stem from  
 182 incorrect bounding box predictions by Grounding DINO, largely due to ambiguities in the top-down  
 183 monocular view—for example, a toy ambulance being misidentified as a charger. The remaining two  
 184 failures involve flat objects (a metal fork and a plastic spoon), where the point clouds merge with the  
 185 table surface, rendering the objects indistinguishable even to human observers. Transparent objects  
 186 pose a similar challenge, as missing depth information leads to point-cloud-based grasping failures.  
 187 However, since RGB images reliably capture these objects, our RGB-based model overcomes these  
 188 limitations and succeeds where the baseline fails.

189 Overall, AnyGrasp and our method provide complementary solutions. AnyGrasp excels at fast,  
 190 robust grasping when clear point clouds are available, while our approach addresses its limitations  
 191 by incorporating RGB observation and supporting intuitive human interaction through language  
 192 instructions and few-shot learning. This flexibility enables our model to adapt to diverse objects  
 193 and specialized tasks (e.g., grasping in a specific pose and order), making it a versatile solution for  
 194 real-world robotic applications.

## 195 G Ablation of Camera Views

196 **Impact of the Number of Input Views.** To ensure a fair comparison, we use only front-view  
 197 images as input for our method, consistent with the single-view baseline OpenVLA. As shown in  
 198 Table 7, this constraint results in approximately 30% lower performance compared to our multi-  
 199 view approach. Nevertheless, our model still achieves 40% higher performance than OpenVLA,  
 200 demonstrating the effectiveness of our design.

| Model                 | Synthetic   | Web         |
|-----------------------|-------------|-------------|
| OpenVLA (single-view) | 20.0        | 3.3         |
| Ours (single-view)    | 60.0        | 56.6        |
| Ours (multi-view)     | <b>93.3</b> | <b>93.3</b> |

Table 7: **Impact of number of input views.** Comparison of GraspVLA with different numbers of input views. The results demonstrate that while multiple views significantly improve performance, our single-view implementation still outperforms the OpenVLA baseline by 40%.

## 201 H Mitigation of Sim-to-Real Gap

202 In this section, we examine the sim-to-real gap in the context of training a VLA model for grasping  
 203 using imitation learning. The sim-to-real gap primarily appears in two key areas: visual appearance  
 204 and physical dynamics.

205 **Visual appearance.** Thanks to advances in pre-trained vision encoders and ray-traced rendering,  
 206 the visual discrepancy between synthetic and real-world RGB images has significantly narrowed.  
 207 By leveraging diverse material and texture datasets, we can generate realistic scenes that cover a  
 208 wide range of robotic grasping scenarios—far more efficiently than collecting equivalent real-world



data across varied environments (as discussed in A). Even when certain material or texture combinations appear unrealistic (e.g., a red table against a green wall), the model still learns generalizable representations from such diversity, consistent with findings in [16]. Additionally, co-training with large-scale Internet vision-language datasets further enhances the model’s robustness to visual discrepancies [17].

**Physical dynamics.** The sim-to-real gap in physical dynamics arises mainly from inaccuracies in modeling material properties (e.g., surface friction), contact dynamics (e.g., forces, friction, deformations), and actuator/sensor behavior. In this work, we mitigate this gap through three key design choices:

- **Simplified control.** We use positional control and treat gripper actions as discrete open/close commands, avoiding complex dynamics modeling.
- **Stability filtering.** We reject grasps that are overly sensitive to physical dynamics, ensuring the model prioritizes robust strategies.
- **Geometry-driven planning.** We focus on mesh-based grasp poses rather than dynamics-dependent policies, enhancing robustness to physical variations.

While these strategies effectively reduce the sim-to-real gap for grasping, they may not generalize to tasks requiring fine-grained dynamics understanding, such as non-prehensile manipulation. We leave the investigation of such scenarios to future work.

## I Inference Delay

The combination of autoregression and flow matching in GraspVLA introduces additional inference delay. Based on Section 5.6 and Table 8, while PAG is critical for a high grasp success rate, it contributes to  $\sim 63\%$  inference delay due to 14 additional tokens to generate. We leave the further improvement of the inference efficiency with PAG as future work. We additionally found that the prefill stage has a similar delay as the decode stage, which could be due to a low GPU utilization with single-sample inference.

| component                 | inference time (ms) |
|---------------------------|---------------------|
| vision encoder            | 9                   |
| bounding boxes (8 tokens) | 72                  |
| grasp pose (6 tokens)     | 50                  |
| flow matching             | 64                  |

Table 8: Breakdown of inference time on NVIDIA L40s GPU.

## J Non-Blocking Control

We explore the implementation of non-blocking controller for smooth action. We implement a Cartesian-space impedance controller adapted from Franka ROS [18] and SERL Franka Controllers [19]. The architecture converts Cartesian impedance commands into joint-space control via real-time Jacobian-based transformation with singularity handling, while optimizing impedance parameters through system identification.

To mitigate abrupt target transitions, we evaluated multiple filter implementations and selected a cascaded filter design for its superior smoothing performance (Figure 5). It achieves fast convergence without overshoot while avoiding excessive initial acceleration, which is suitable for the output characteristics of the GraspVLA model. Additionally, positional interpolation was adopted instead of temporal interpolation to address synchronization mismatches between model computation latency and control pipelines.

To achieve more fluent and coherent motions, GraspVLA generates multi-step predictions at each inference cycle. These predictions are incorporated via a receding-horizon optimization scheme

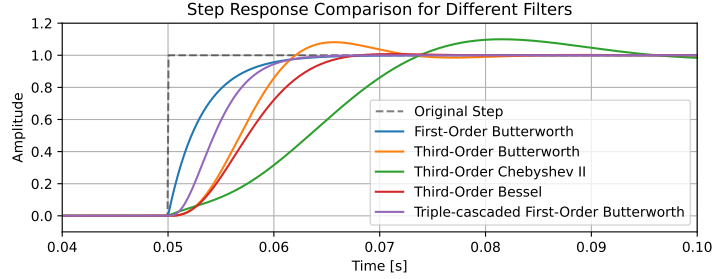


Figure 5: **Step response comparison for different filters.** In the comparison, these filters set the same sampling frequency and cutoff frequency. The first-order Butterworth filter exhibits a large initial acceleration in its step response. The third-order Butterworth filter, third-order Chebyshev II filter, and third-order Bessel filter all exhibit overshoot to varying degrees. The triple-cascaded first-order Butterworth filter can avoid excessive initial acceleration and eliminate overshoot while maintaining convergence speed, making it an ideal filter choice.

248 within the asynchronous control architecture, where filter and interpolation strategies are systemat-  
 249 ically applied to the predicted trajectory. This non-blocking control architecture proactively com-  
 250 pensates for computational latency variations while ensuring smooth interleaving of control actions,  
 251 which significantly mitigates oscillatory patterns in dynamic manipulation scenarios.

252 While we employ non-blocking control for demonstration recording to achieve natural trajectories,  
 253 all experimental evaluations use blocking control to ensure rigorous performance measurement.

## 254 K Failure Analysis

255 To thoroughly assess the limitations of our approach, we conduct a detailed failure analysis. Since  
 256 the test set in the main paper reveals only two failure cases—which may not be representative—we  
 257 design a significantly more challenging test set featuring cluttered scenes. Specifically, we randomly  
 258 place objects across the table to cover the entire workspace and stack some objects (e.g., placing a  
 259 strawberry on top of a bulldozer) to create complex, occluded scenarios. We then evaluate the  
 260 model’s performance under these conditions and identify the primary failure modes.

261 The most frequent failure case (31%) occurs when the model hesitates due to ambiguous language  
 262 instructions, such as when multiple objects match the description (e.g., two target bottles). This  
 263 could be mitigated by incorporating longer contextual history. The second most common issue  
 264 (27%) arises in highly cluttered scenes, where the model misidentifies objects, likely due to in-  
 265 sufficient training data for such scenarios. Future work could utilize advanced data augmentation  
 266 methods and generative modeling techniques to create more diverse and complex training samples.  
 267 Another notable failure mode (21%) involves objects with smooth surfaces (e.g., plastic balls) slip-  
 268 ping during grasping, which tactile feedback might help resolve. Additionally, when the target object  
 269 is occluded (14%), the model struggles to grasp it precisely, suggesting a need for active perception  
 270 techniques. Finally, the remaining failures (7%) include minor errors such as early gripper closure or  
 271 collisions with the environment, which reinforcement learning could potentially address. We leave  
 272 these potential improvements for future work.

## References

- [1] H. Bharadhwaj, J. Vakil, M. Sharma, A. Gupta, S. Tulsiani, and V. Kumar. Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking, 2023.
- [2] H. Walke, K. Black, A. Lee, M. J. Kim, M. Du, C. Zheng, T. Zhao, P. Hansen-Estruch, Q. Vuong, A. He, V. Myers, K. Fang, C. Finn, and S. Levine. Bridgedata v2: A dataset for robot learning at scale, 2024. URL <https://arxiv.org/abs/2308.12952>.
- [3] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.
- [4] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.
- [5] AgiBot-World-Contributors, Q. Bu, J. Cai, L. Chen, X. Cui, Y. Ding, S. Feng, S. Gao, X. He, X. Hu, X. Huang, S. Jiang, Y. Jiang, C. Jing, H. Li, J. Li, C. Liu, Y. Liu, Y. Lu, J. Luo, P. Luo, Y. Mu, Y. Niu, Y. Pan, J. Pang, Y. Qiao, G. Ren, C. Ruan, J. Shan, Y. Shen, C. Shi, M. Shi, M. Shi, C. Sima, J. Song, H. Wang, W. Wang, D. Wei, C. Xie, G. Xu, J. Yan, C. Yang, L. Yang, S. Yang, M. Yao, J. Zeng, C. Zhang, Q. Zhang, B. Zhao, C. Zhao, J. Zhao, and J. Zhu. Agibot world colosseum: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.
- [6] A. O’Neill, A. Rehman, A. Gupta, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.
- [7] google-deepmind/envlogger, Jan. 2025. URL <https://github.com/google-deepmind/envlogger>. original-date: 2021-07-28T15:35:08Z.
- [8] Universally unique identifier, Jan. 2025. URL [https://en.wikipedia.org/w/index.php?title=Universally\\_unique\\_identifier&oldid=1272340425](https://en.wikipedia.org/w/index.php?title=Universally_unique_identifier&oldid=1272340425). Page Version ID: 1272340425.
- [9] S. Valette and J.-M. Chassery. Approximated centroidal voronoi diagrams for uniform polygonal mesh coarsening. In *Computer Graphics Forum*, volume 23, pages 381–389. Wiley Online Library, 2004.
- [10] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. pi0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.
- [11] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.
- [12] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.
- [13] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.
- [14] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi:10.15607/RSS.2023.XIX.016.

- 319 [15] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su, J. Zhu,  
320 and L. Zhang. Grounding dino: Marrying dino with grounded pre-training for open-set object  
321 detection, 2024. URL <https://arxiv.org/abs/2303.05499>.
- 322 [16] A. Maddukuri, Z. Jiang, L. Y. Chen, S. Nasiriany, Y. Xie, Y. Fang, W. Huang, Z. Wang, Z. Xu,  
323 N. Chernyadev, S. Reed, K. Goldberg, A. Mandlekar, L. Fan, and Y. Zhu. Sim-and-real co-  
324 training: A simple recipe for vision-based robotic manipulation, 2025. URL <https://arxiv.org/abs/2503.24361>.
- 326 [17] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi,  
327 C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak,  
328 T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z.  
329 Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke,  
330 A. Walling, H. Wang, L. Yu, and U. Zhilinsky.  $\pi_{0.5}$ : a vision-language-action model with  
331 open-world generalization, 2025. URL <https://arxiv.org/abs/2504.16054>.
- 332 [18] F. R. GmbH. Ros integration for franka robotics research robots. [https://github.com/frankaemika/franka\\_ros](https://github.com/frankaemika/franka_ros), 2023.  
333
- 334 [19] J. Luo, Z. Hu, C. Xu, Y. L. Tan, J. Berg, A. Sharma, S. Schaal, C. Finn, A. Gupta, and S. Levine.  
335 Serl: A software suite for sample-efficient robotic reinforcement learning, 2024.