DGA-VLA: A LIGHTWEIGHT DEPTH-GUIDED GRASP ENHANCEMENT FOR VLA MODELS WITHOUT RETRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models provide a unified policy foundation for a wide range of robotic manipulation tasks. However, their grasping performance remains constrained by imprecise gripper-object alignment at closure, leading to empty or unstable grasps and task failures. To address this limitation, we propose DGA-VLA, a lightweight depth-guided grasp refinement framework that augments pretrained VLA policies without VLA architectural changes or policy retraining. DGA-VLA couples the VLA's global motion generation with a wrist-depth feedback controller, enabling local correction without changing the underlying policy. When the policy indicates a closure intent, the controller uses wrist-depth feedback to estimate local gripper-object offsets and apply bounded corrective motions before closure. Experiments demonstrate that DGA-VLA improves average success rates on grasping tasks by 12.0 percentage points on RoboTwin, 8.4 points on SimplerEnv, and 14.6 points on real-robot pick-and-place tasks.

1 INTRODUCTION

Vision-language-action (VLA) models are emerging as a promising foundation for general-purpose robotic manipulation. By combining the semantic knowledge of large vision-language models with robot demonstrations, recent systems can map visual observations and natural-language instructions directly to low-level actions, enabling a single policy to perform diverse tasks across objects, scenes, and embodiments (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024; Physical Intelligence et al., 2025; NVIDIA et al., 2025). This end-to-end paradigm substantially reduces the need to hand-engineer a separate perception and planning stack for every task. More importantly, it gives robots a powerful semantic interface: a VLA policy can understand *what* object should be manipulated and *what* high-level behavior is required from an open-ended language command.

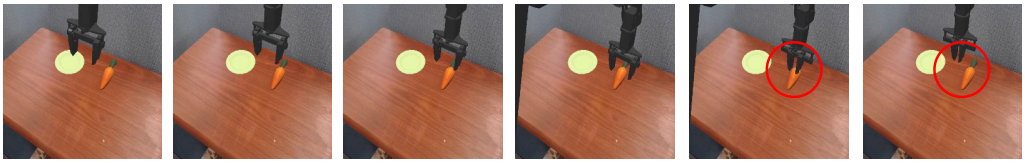
However, semantic competence does not necessarily guarantee geometric precision at contact. Fine manipulation—especially grasping—can hinge on small local errors near contact. A VLA policy may identify the correct object and generate a semantically appropriate trajectory, yet still reach the final approach with residual misalignment between the gripper and the object, as illustrated in Figure 1. Such residual errors can arise from visual ambiguity, occlusion, distribution shift, and accumulated closed-loop error. These challenges have motivated efforts to improve geometric reliability around contact, either by correcting predicted actions at inference time or by incorporating richer geometric information into the VLA policy.

One line of work attempts to correct these failures at inference time (Zhang et al., 2026b; Fan et al., 2026; Song et al., 2026; Jo et al., 2026). PhysVLA applies a phase-aware finite-state machine and an Euler–Lagrange consistency gate (Chandra et al., 2026), but its corrections rely on privileged simulator state, including object positions and task-specific distance predicates. VLA-SCT instead evaluates trajectories using geometric statistics, perturbs actions from visually matched successful experiences, and detects termination through Pearson correlation between the current image and stored successful images (Zhang et al., 2026b). Its correction quality therefore depends on the coverage of a successful-action repository, while global image correlation can be confounded by viewpoint, layout, and occlusion changes. These factors make their local corrections difficult to reuse across tasks and scenes.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) LIBERO(Pi 0.5): Put both moka pots on the stove.



(b) SimplerEnv(GR00T N1.7): Put carrot on plate.

Figure 1: **Representative grasp failures caused by local misalignment near contact.** Although the VLA policies approach the correct target, the gripper misses the object during the final grasp in (a) LIBERO with $\pi_{0.5}$ and (b) SimplerEnv with GR00T N1.7. Red circles highlight the failed grasps.

Another line of work introduces three-dimensional information into the VLA itself. Explicit approaches train point-cloud encoders, depth experts, or multi-view 3D fusion and completion modules (Li et al., 2025a; Yuan et al., 2025; Bhat et al., 2025; Rao et al., 2026; Xia et al., 2026); implicit approaches learn latent depth, stereo, or depth-attention features through additional alignment and policy fine-tuning stages (Lin et al., 2026; Han et al., 2026; Deng et al., 2025; Jia et al., 2026; Wang et al., 2026). Despite their different mechanisms, these approaches require at least part of the VLA—typically the action expert, and sometimes the vision-language backbone—to be trained or fine-tuned to absorb the new geometric representations. Their gains therefore depend on specialized 3D or stereo data and multi-stage optimization, which must be repeated for each target VLA backbone. VLM4VLA further shows that improving VLM capabilities on auxiliary embodied tasks, including depth estimation and visual pointing, does not necessarily translate into better downstream VLA control (Zhang et al., 2026a).

Classical robot control suggests a useful principle for addressing this gap. Visual servoing uses fresh geometric feedback in a closed loop and can iteratively reduce local alignment error before contact (Chaumette & Hutchinson, 2006). However, a conventional visual-servoing system typically assumes that the target and desired visual state are specified in advance, and continuously controls the robot toward that state. Integrating this principle into VLA execution therefore raises two questions: how to arbitrate control between the VLA and local geometric feedback, including when to intervene and when to return control; and how to determine the target and desired local grasp state during VLA execution. These coordination questions are not specified by conventional visual servoing alone. The relevant challenge is to exploit its closed-loop correction principle without replacing or disrupting the VLA's task-level behavior.

In this work, we propose DGA-VLA, a depth-guided precise grasping assistance framework that uses a visual-servoing-inspired feedback mechanism tailored to VLA execution. The framework uses the VLA's gripper-closure intent to trigger bounded wrist-depth corrections, extracts a local target region from the wrist observation, compares it with an object-specific calibrated grasp anchor, and returns control to the VLA for replanning after the alignment is complete. This design establishes a clear division of labor: the VLA determines the task-relevant object and generates the global approach trajectory, while the depth-guided controller identifies the nearby geometric target and temporarily refines its local alignment during the precision-critical approach. It thereby introduces explicit geometric feedback while preserving the general behavior of the underlying policy.

Our main contributions are as follows:

- We introduce DGA-VLA, a selective depth-guided assistance framework that couples the task-level semantic reasoning and global motion generation of a VLA policy with closed-loop geometric correction during object approach. The framework leaves the policy unchanged and activates geometric assistance only when local precision is required.

- We develop a lightweight depth-guided grasp alignment controller that detects a nearby target from wrist-camera depth, measures image-plane and depth errors relative to a grasp anchor, and applies bounded closed-loop translations during the approach to the target object.
- We evaluate the framework in simulation and on real-robot grasping tasks. DGA raises the average task success rate from 42.3% to 54.3% on RoboTwin, from 48.2% to 56.6% on SimplerEnv, and from 26.7% to 41.3% in the real-robot experiments.

2 Related Work

General-Purpose Vision-Language-Action Models. Vision-language-action models extend large vision-language models from semantic reasoning to closed-loop robot control. Representative systems such as RT-2, OpenVLA, π_0, and GR00T N1 demonstrate that a single policy can connect open-ended instructions and multi-view observations to general-purpose robot actions (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024; Physical Intelligence et al., 2025; NVIDIA et al., 2025). These systems span autoregressive action tokenization and continuous generative action formulations, yet they share an end-to-end objective: infer robot motion directly from visual observations and language. This design provides strong semantic generality and allows the policy to approach the correct object across diverse tasks. However, the fine-grained manipulation precision of these policies is not always reliable, particularly near contact. A trajectory that is semantically correct may therefore still fail to achieve accurate local alignment before grasping. This gap motivates contact-level feedback that can complement an existing VLA policy without requiring a new backbone.

Visual Servoing for Robotic Grasping. Robotic grasping requires both selecting suitable contacts and accurately executing the end-effector motion that realizes them (Bicchi & Kumar, 2000). Visual servoing provides a classical framework for the latter by closing the feedback loop between visual observations and robot motion. Position-based visual servoing regulates an estimated relative pose, whereas image-based visual servoing maps errors in observed image features directly to control commands (Chaumette & Hutchinson, 2006; Zhu et al., 2025). In robotic grasping, it has been used specifically to refine the approach and alignment stages. Ma et al. divide precision grasping into alignment, approach, and grasp stages, using monocular visual feedback to regulate the alignment motion (Ma et al., 2020). Enyedy et al. instead map stereo image-space errors to manipulator motion for three-dimensional reaching and grasping without explicit stereo reconstruction (Enyedy et al., 2024). More recently, ViT-VS combines pretrained transformer correspondences with an image-based visual-servo controller, improving robustness to visual variation while retaining a closed-loop control structure (Scherl et al., 2025). These methods show that visual feedback can provide precise local motion regulation, but they are typically designed as standalone positioning or grasping systems. Our framework instead retains the VLA policy for task-level object selection and global approach motion, and invokes a lightweight wrist-depth feedback controller only during the precision-critical approach before gripper closure.

3 Method

3.1 Problem Formulation and Framework Overview

We consider a pretrained VLA policy π_θ that receives the current robot observation o_t and language instruction l and predicts actions

$$\mathbf{A}_t = \pi_\theta(o_t, l) = \{\mathbf{a}_{t,1}, \ldots, \mathbf{a}_{t,H}\}. \tag{1}$$

This formulation accommodates both common interfaces: autoregressive policies such as OpenVLA that predict one action at a time, and action-chunk policies such as π_0 that predict multiple temporally ordered actions. The proposed controller can wrap either interface.

For clarity, we formulate the controller for a single arm, using a standard 7-DoF Cartesian action interface as an example: $\mathbf{a} = [\Delta\mathbf{p}, \Delta\mathbf{r}, g]$, containing translational motion $\Delta\mathbf{p} \in \mathbb{R}^3$, rotational motion $\Delta\mathbf{r} \in \mathbb{R}^3$, and a gripper command g. In addition to the observations already used by the VLA, our framework reads a wrist-camera depth map D_t. It wraps, rather than modifies, π_θ and

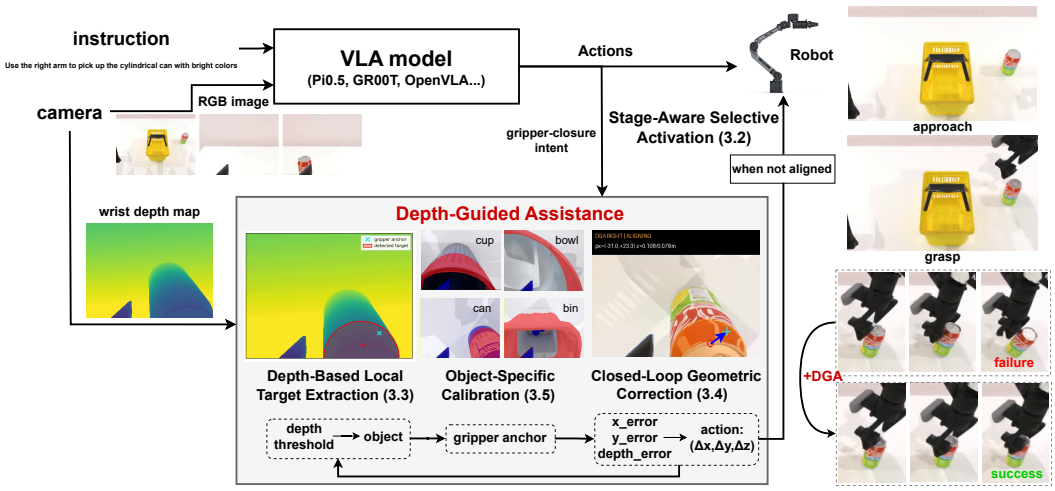


Figure 2: **Overview of the proposed DGA-VLA framework.** The VLA policy processes the language instruction and RGB observation and controls the robot by default. When its action indicates gripper-closure intent, the framework selectively activates Depth-Guided Assistance (DGA), extracts the nearby target from the wrist-depth map, retrieves the object-specific calibration, and applies bounded closed-loop geometric corrections before returning control to the VLA.

maintains a small controller state $m_t \in \{\texttt{idle}, \texttt{align}\}$. We refer to the complete wrapper as DGA-VLA and to its depth-guided local controller as DGA.

3.2 Stage-Aware Selective Activation and Control Arbitration

VLA manipulation policies exhibit clear stage-wise behavior (Li et al., 2025b; Wang et al., 2025), and their precision requirements vary across stages. A typical pick-and-place episode contains approach, grasp, transport, and placement (or release) stages (Chandra et al., 2026). The grasp stage is especially precision-critical: the appropriate grasp can depend on the object's shape and on the task-specific grasp site, and errors at this stage are a common source of task failure. Moreover, grasp stability affects the subsequent transport and placement stages. Improving grasp success therefore provides a direct route to improving overall task success.

Our activation signal exploits the structure of this execution. A predicted close command serves as a proxy for the policy's intent to initiate grasping; it does not guarantee that the end effector has reached the intended grasp site. We therefore use gripper-closure intent as the activation signal. In the normal `idle` state, VLA actions are executed unchanged. When the upcoming VLA action or action chunk expresses an intent to close the gripper, DGA extracts a local target candidate from the wrist-camera depth image and checks whether it is aligned with a predefined grasp anchor. If it is already aligned, the original close action is preserved. Otherwise, the framework enters `align`, discards the stale pending actions, and temporarily emits bounded translational corrections while holding the gripper open. Control returns to the VLA after alignment, target loss, or a maximum number of correction steps, and the VLA then replans from the updated observation. This arbitration confines intervention to the moment at which the VLA itself predicts that contact is imminent.

3.3 Depth-Based Local Target Extraction

As described in Sec. 3.2, gripper-closing intent prompts DGA to extract a local target candidate from the wrist-camera depth map D_t. This procedure assumes that the preceding VLA motion has driven the robot arm to bring the gripper into a local neighborhood of the intended grasp part. The intended grasp part is therefore closer to the wrist camera than most background geometry. A metric depth threshold can consequently remove much of the farther background and retain nearby candidate regions. Task-level affordance selection is provided by the VLA's preceding motion, whereas the depth module only localizes nearby geometry for alignment.

The wrist camera also observes parts of the gripper. To prevent them from being mistaken for the target, we construct a gripper mask M^g from the initial depth frame. Target extraction is then restricted to a grasp-centered region of interest Ω. Let (u, v) denote the image column and row coordinates, respectively. For a metric detection threshold τ_d, the binary candidate mask is

$$M_t(u, v) = \mathbb{I}[(u, v) \in \Omega]\, \mathbb{I}[D_t(u, v) \le \tau_d]\, \mathbb{I}[M^g(u, v) = 0]. \tag{2}$$

We divide M_t into connected components, each representing a separate candidate region. Components with relatively few pixels are discarded to suppress isolated depth noise and fragments. For each remaining component $\mathcal{C}_j$, we assign larger weights to closer pixels, $w_i = \tau_d - D_t(u_i, v_i) + \epsilon$, and estimate its depth-weighted image centroid and mean depth:

$$\mathbf{c}_j = \frac{\sum_{i\in\mathcal{C}_j} w_i [u_i, v_i]^\top}{\sum_{i\in\mathcal{C}_j} w_i}, \qquad z_j = \frac{\sum_{i\in\mathcal{C}_j} w_i D_t(u_i, v_i)}{\sum_{i\in\mathcal{C}_j} w_i}. \tag{3}$$

The selected target is the valid component whose centroid is closest to the reference grasp anchor $\mathbf{q}^* = [u^*, v^*]^\top$. This rule favors candidate regions near the expected grasp position. The anchor and its desired approach depth z^* describe where a successfully graspable object should appear between the fingers.

3.4 Closed-Loop Geometric Correction

At each control step t, the controller acquires a new wrist-depth image, repeats the target extraction and selection in Section 3.3, recomputes the geometric errors, and applies one bounded correction. The next step observes the target after that motion and repeats this process, forming a closed feedback loop. For the selected component, let (c_u, c_v) denote its centroid coordinates and z_t its current mean depth. We define image-plane errors $e_u = c_u - u^*$ and $e_v = c_v - v^*$. The one-sided depth error is

$$e_z = \max(z_t - z^*, 0), \tag{4}$$

so the controller advances toward an object that is farther than the reference depth but does not command an additional retreat once the object is sufficiently close. To obtain a metric correction, the pixel errors are back-projected using the current target depth and the camera focal lengths f_x and f_y. The complete camera-frame correction is

$$\widetilde{\Delta\mathbf{p}}_t^{\,c} = \left[\frac{e_u z_t}{f_x}, \frac{e_v z_t}{f_y}, e_z\right]^\top, \tag{5}$$

We then define

$$\mathrm{sat}_{s_{\max}}(\mathbf{v}) = \begin{cases} \mathbf{v}, & \|\mathbf{v}\|_2 \le s_{\max}, \\ s_{\max}\mathbf{v}/\|\mathbf{v}\|_2, & \|\mathbf{v}\|_2 > s_{\max}. \end{cases} \tag{6}$$

After norm limiting, the complete correction is rotated into the robot base/world frame using the current wrist-camera extrinsics:

$$\Delta\mathbf{p}_t^b = \mathbf{R}_c^b\, \mathrm{sat}_{s_{\max}}(\widetilde{\Delta\mathbf{p}}_t^{\,c}). \tag{7}$$

Here, $\mathbf{R}_c^b$ is the camera-to-base/world rotation and $s_{\max}$ bounds the total translation magnitude in one control step. The resulting auxiliary action contains only translation: $\mathbf{a}_t^{\text{align}} = [\Delta\mathbf{p}_t^b, \mathbf{0}_3, g_{\text{open}}]$, where $\mathbf{0}_3$ holds the three rotational action dimensions fixed and g_{open} keeps the gripper open.

The target is considered aligned when

$$|e_u| \le \varepsilon_u, \qquad |e_v| \le \varepsilon_v, \qquad e_z \le \varepsilon_z \tag{8}$$

at the current control step, where ε_u, ε_v, and ε_z are tolerances along the two image axes and depth axis, respectively. Once this condition is met, the auxiliary controller terminates and control returns to the VLA; the policy then receives a fresh observation and issues the next action. If no valid component can be found, the target leaves the region of interest, or the correction reaches its step budget, the controller likewise aborts and returns to the VLA.

3.5 Object-Specific Calibration and Training-Free Integration

Object geometry and task-specific contact locations vary across scenes and tasks, so a single fixed reference center cannot represent the desired alignment for every object. We therefore estimate object-specific reference coordinates from a small set of expert demonstrations and organize the resulting parameters in a reusable library indexed by task and object category.

For each expert trajectory, we locate the last frame in which the gripper remains open before a close transition. We then apply the same gripper-mask construction, grasp ROI, depth threshold, and connected-component filtering used at runtime, and select the valid component corresponding to the demonstrated object. Across these demonstrations, the depth-weighted centroid and mean depth determine the reference target coordinates (u^*, v^*) and desired approach depth z^*. For the same object category and task instruction, the corresponding library entry can be retrieved and reused across later scenes. New object categories can be incorporated by adding corresponding anchor entries to the library.

The resulting calibration library is used by a runtime controller that operates at inference time without updating the VLA policy parameters. The controller takes as input the upcoming VLA action chunk, metric wrist depth, the current end-effector pose, and the wrist-camera intrinsics and extrinsics. With this interface, the same controller can be attached to another VLA backbone with a compatible action interface.

4 Experiments

4.1 Setup

Simulation. We evaluate DGA-VLA on three manipulation benchmarks: RoboTwin (Chen et al., 2025), LIBERO (Liu et al., 2023), and SimplerEnv (Li et al., 2024). RoboTwin provides a multi-task evaluation platform across diverse robot embodiments and supports extensive scene and object randomization. We use the $\pi_{0.5}$ checkpoint provided in the Motus code repository (Bi et al., 2025) and select eight RoboTwin tasks that involve grasping objects of varied shapes, evaluating each task in clean scenes over 100 episodes with unseen instructions. LIBERO comprises the Spatial, Object, Goal, and Long suites; each suite contains 10 tasks, and each task is evaluated with 50 trials. We test DGA on LIBERO with three VLA backbones—$\pi_{0.5}$, π_0 (Black et al., 2024), and OpenVLA (Kim et al., 2024). For SimplerEnv, we use the WidowX robot and evaluate five grasping tasks with GR00T N1.7 (NVIDIA et al., 2025), using 100 trials per task. All simulation experiments are conducted on NVIDIA GeForce RTX 3090 Ti.

Real-Robot. We evaluate DGA on a Galbot G1 using RhinoVLA (Huixi Intelligence et al., 2026), a VLA designed for efficient deployment with a π_0-style architecture that couples a pretrained vision–language backbone with a continuous action expert. We evaluate RhinoVLA with and without DGA on pick-and-place tasks involving three distinct objects, using 25 position configurations per object. The metric depth is obtained from the right-wrist-mounted Intel RealSense D405C stereo depth camera. Policy inference runs on an NVIDIA GeForce RTX 5090.

DGA calibration. We use demonstration trajectories with wrist-depth observations to estimate the grasp anchor and target depth. Each RoboTwin task uses 10 such trajectories, whereas we use one trajectory per task in LIBERO, SimplerEnv, and the real-robot experiments because their current calibration setup has a single fixed grasp configuration. The larger RoboTwin set is needed because one task may involve different grasped objects or either arm, which changes the relevant wrist view and target anchor.

4.2 Simulation Results

RoboTwin. Table 1 evaluates $\pi_{0.5}$ with DGA on eight tasks involving grasps of objects with varied shapes. DGA raises average grasp success from 68.3% to 79.5%, making grasps more accurate and stable. It also improves task success on every task, raising the average from 42.3% to 54.3%. Among successful episodes, the average number of action steps to task completion remains similar. DGA contributes only 5.4 control steps per successful episode on average, approximately 2.9% of the

Table 1: **Performance of $\pi_{0.5}$ with and without DGA in RoboTwin. SR:** task success rate. **ΔSR:** success-rate improvement. **GSR:** grasp success rate. **ΔGSR:** grasp-success-rate improvement. **Steps:** average number of action steps executed to task completion. **DGA steps:** average number of DGA control steps. Steps and DGA steps are counted only for successful episodes.

Task	$\pi_{0.5}$			$\pi_{0.5}$ + DGA (ours)					
	GSR	SR	Steps	GSR	Δ GSR	SR	Δ SR	Steps	DGA steps
dump bin bigbin	73.0	64.0	217.3	**85.0**	**+12.0**	**74.0**	**+10.0**	208.1	5.9
move can pot	38.0	20.0	151.9	**46.0**	**+8.0**	**26.0**	**+6.0**	141.5	2.5
place can basket	61.0	29.0	315.2	**89.0**	**+28.0**	**54.0**	**+25.0**	299.2	9.7
place container plate	74.0	70.0	144.7	**89.0**	**+15.0**	**83.0**	**+13.0**	147.5	5.6
place empty cup	63.0	60.0	174.6	**79.0**	**+16.0**	**74.0**	**+14.0**	174.2	8.8
place fan	75.0	32.0	165.1	**82.0**	**+7.0**	**48.0**	**+16.0**	163.6	3.3
place mouse pad	69.0	13.0	156.0	**68.0**	-1.0	**18.0**	**+5.0**	168.2	3.1
place shoe	93.0	50.0	153.8	**98.0**	**+5.0**	**57.0**	**+7.0**	166.9	4.0
Average	68.3	42.3	184.8	**79.5**	**+11.2**	**54.3**	**+12.0**	183.7	5.4

Table 2: **Comparison on LIBERO.** LIBERO comprises the Spatial, Object, Goal, and Long suites. Bold denotes the highest success rate within each VLA backbone.

Method	Spatial	Object	Goal	Long	Average
$\pi_{0.5}$	98.2	**99.0**	98.0	91.4	96.65
$\pi_{0.5}$ + PhysVLA	**98.4**	**99.0**	98.0	91.2	96.65
$\pi_{0.5}$ + DGA (ours)	98.2	98.6	**98.6**	**94.2**	**97.40**
π_0	95.6	97.4	93.0	**81.2**	91.80
π_0 + PhysVLA	95.8	**98.2**	91.6	78.8	91.10
π_0 + DGA (ours)	**96.4**	**98.2**	**93.2**	**81.2**	**92.25**
OpenVLA	86.6	86.6	77.2	52.6	75.75
OpenVLA + PhysVLA	86.2	**88.8**	**79.0**	54.4	**77.10**
OpenVLA + DGA (ours)	**86.8**	87.2	78.2	**55.4**	76.90

183.7 total steps. This small share reflects the stage-aware activation: DGA intervenes only during the precision-critical grasp approach to correct local misalignment, leaving the rest of the VLA trajectory largely unchanged.

LIBERO. Across the four suites in Table 2, DGA raises the average task success rate from 96.65% to 97.40% for $\pi_{0.5}$, from 91.80% to 92.25% for π_0, and from 75.75% to 76.90% for OpenVLA. The gains vary across suites, with limited headroom for the already strong $\pi_{0.5}$ and π_0 baselines. For $\pi_{0.5}$ and OpenVLA, the largest suite-level gains occur on LIBERO-Long, both at 2.8 percentage points. On its challenging Task 8, success rises from 50% to 68% for $\pi_{0.5}$ and from 18% to 28% for OpenVLA. These task-level results illustrate improvements that are less apparent in the overall averages.

We also compare with PhysVLA (Chandra et al., 2026), a training-free method that uses privileged simulator state for physically grounded action correction. We reproduce it from the published method description and evaluate it under the same protocol. Using wrist-depth feedback, DGA-VLA achieves comparable or greater gains in success rate than PhysVLA, without requiring privileged simulator state.

SimplerEnv. As shown in Table 3, DGA improves the task success rate of GR00T N1.7 on five grasping tasks with the WidowX robot, raising the average from 48.2% to 56.6%.

Ablation analysis. We first compare DGA with two variants that retain its triggering conditions but replace geometric correction. Depth-Gated Close Suppression (DGCS) keeps the gripper open without changing the VLA's motion commands, whereas Depth-Gated Close Replanning (DGCR) discards the pending action chunk and immediately queries the VLA again using the current observa-

Table 3: **Ablation Study with GR00T N1.7 on SimplerEnv. DGCS:** Depth-Gated Close Suppression. **DGCR:** Depth-Gated Close Replanning. **DGA-1Step:** one depth-guided alignment followed by an immediate return to the VLA.

Task	GR00T N1.7	+ DGCS	+ DGCR	+ DGA-1Step	+ DGA (ours)
put spoon on towel	89.0	87.0	49.0	87.0	**90.0**
put carrot on plate	62.0	84.0	66.0	59.0	**82.0**
stack cube	40.0	40.0	12.0	41.0	**46.0**
put eggplant in basket	50.0	12.0	21.0	57.0	**61.0**
put eggplant in sink	0.0	0.0	5.0	1.0	**4.0**
Average	48.2	44.6	30.6	49.0	**56.6**

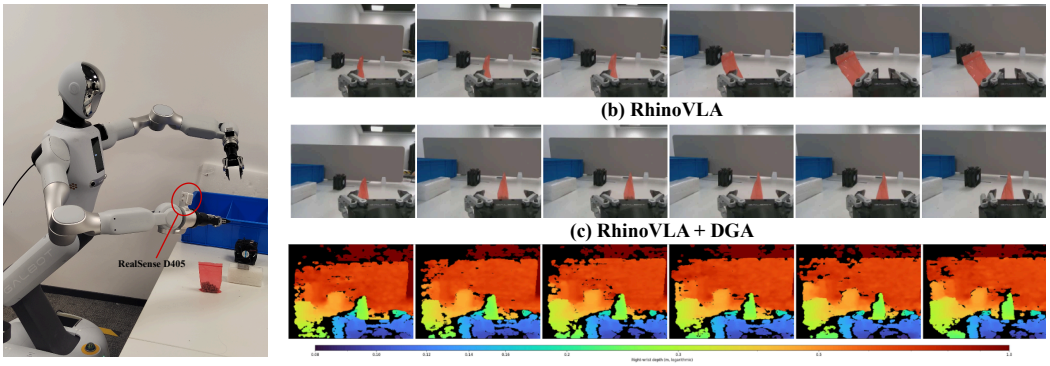


Figure 3: **Real-robot setup and qualitative comparison on "Place red bag in bin."** (a) Galbot G1 and the experimental scene. (b) RhinoVLA fails to grasp the red bag because it is not centered between the gripper fingers. (c) With DGA, the gripper is aligned with the red bag before closure, enabling a successful grasp. (d) Right-wrist depth observations used by DGA for local grasp alignment.

tion. Both variants defer recovery from grasp misalignment to the VLA. DGCS and DGCR achieve average success rates of 44.6% and 30.6%, respectively, both below the vanilla policy, showing that relying on VLA replanning to recover from grasp misalignment does not provide consistent gains. We then evaluate DGA-1Step, which applies one depth-guided correction at activation and immediately returns control to the VLA. Its average success rate is 49.0%, only 0.8 percentage points above the vanilla policy and 7.6 points below DGA. Together, these results indicate that DGA's consistent gains require iterative closed-loop geometric correction; relying on VLA replanning or applying only a single correction is insufficient. Additional ablations on LIBERO are reported in Appendix D.

4.3 Real-Robot Results

In the real-robot experiments, the wrist camera estimates depth using stereo vision and provides a metric depth map directly to DGA. Although these maps contain sparse invalid returns and local measurement noise, the target remains distinguishable in the evaluated workspace, and the noise does not prevent depth-based target-region extraction. Manual inspection of failed rollouts shows that over 90% fail during grasping because of inaccurate approach or alignment, rather than during subsequent transport or placement. Figure 3 shows the experimental setup and representative wrist observations during execution.

Across the three evaluated pick-and-place tasks, augmenting RhinoVLA with DGA improves task success on two tasks and maintains the baseline performance on the remaining task. The paired analysis in Table 4 shows that, across 75 trials, DGA converts 15 failures into successes while 4 successes regress to failures. For the red bag and black fan, DGA corrects residual wrist-view misalignment after RhinoVLA has brought the gripper into the local grasping region, yielding gains

Table 4: **Real-robot task success rates on the Galbot G1. F→S:** fraction of matched trials that change from failure to success. **S→F:** fraction that change from success to failure. Each method is evaluated on the same 25 position configurations per task.

Task	RhinoVLA	RhinoVLA + DGA	F→S	S→F
Place red bag in bin	56% (14/25)	**76% (19/25)**	**28% (7/25)**	8% (2/25)
Place black fan in bin	16% (4/25)	**40% (10/25)**	**28% (7/25)**	4% (1/25)
Place white foam in bin	8% (2/25)	**8% (2/25)**	4% (1/25)	4% (1/25)
Overall	26.7% (20/75)	**41.3% (31/75)**	**20.0% (15/75)**	5.3% (4/75)

Table 5: **VLA inference and DGA processing latency.**

Setting	Hardware	VLA (ms)	DGA (ms)
RoboTwin	RTX 3090 Ti	149	3
Galbot G1	RTX 5090	110	12

of 20 and 24 percentage points, respectively. On the white-foam task, RhinoVLA frequently fails to move the right arm close to the foam, so the proximity-gated DGA controller cannot intervene and therefore provides no improvement. This result is consistent with the intended scope of the method: DGA refines a local grasp after the VLA has approached the target, but does not replace the VLA's coarse approach behavior.

4.4 Runtime Overhead

Table 5 reports average processing latency in simulation and on the real robot. In RoboTwin, VLA inference and DGA take 149 and 3 ms, respectively; on the Galbot G1, they take 110 and 12 ms. The DGA latency is within the control-period budget in both settings, satisfying the required real-time control frequency. In real-robot deployment, DGA latency does not introduce perceptible stuttering in robot motion.

5 Conclusion

We introduced DGA-VLA, a lightweight depth-guided grasp refinement framework that augments pretrained VLA policies without model retraining. Experiments across RoboTwin, SimplerEnv, LIBERO, and real-robot pick-and-place tasks demonstrate improvements in task success across multiple VLA backbones. These results support an effective division of labor: the VLA provides global approach motion, while DGA extracts the nearby geometric target and refines the gripper–object alignment before closure through wrist-depth feedback. Overall, DGA-VLA provides a practical way to improve grasp reliability without retraining the underlying VLA policy.

AI use statement

In this work, we used generative AI tools in a limited supporting role to search for relevant literature, discuss possible methodological and experimental options, and assist with code implementation. We have not used generative AI tools to conduct experiments, process or analyze experimental data, generate synthetic datasets, formulate or prove mathematical claims, or interpret the reported results. Additionally, we used generative AI tools to improve the readability of the manuscript. We reviewed all AI-assisted work: literature suggestions were checked by the authors, AI-assisted code was inspected and tested before use, and manuscript edits were revised against the authors' intended claims and evidence. We take responsibility for the final content of this work, including text, claims, code, data, and artifacts produced with the aid of generative AI.

Reproducibility statement

We support reproducibility by providing the complete DGA algorithm in Appendix A, benchmark-specific implementation details in Appendix B, and real-robot deployment settings in Appendix C. Section 4 reports the evaluation protocols, baselines, numbers of trials, and hardware-specific latency measurements. Code will be released.

References

Vineet Bhat, Yu-Hsiang Lan, Prashanth Krishnamurthy, Ramesh Karri, and Farshad Khorrami. 3D CAVLA: Leveraging depth and 3d context to generalize vision language action models for unseen tasks. *arXiv preprint arXiv:2505.05800*, 2025.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, Hongyan Zhao, Hanyu Liu, Zhizhong Su, Lei Ma, Hang Su, and Jun Zhu. Motus: A unified latent action world model. *arXiv preprint arXiv:2512.13030*, 2025.

Antonio Bicchi and Vijay Kumar. Robotic grasping and contact: A review. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, volume 1, pp. 348–353, 2000. doi: 10.1109/ROBOT.2000.844081.

Kevin Black, Noah Brown, Danny Driess, Adnan Esber, Michael Suber, Arjun Srivastava, Pete Florence, Sergey Levine, and Karol Hausman. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of the 7th Conference on Robot Learning*, 2023.

Namai Chandra, Shriram Damodaran, and Lin Wang. PhysVLA: Towards physically-grounded VLA for embodied robotic manipulation. *arXiv preprint arXiv:2606.13886*, 2026.

François Chaumette and Seth Hutchinson. Visual servo control. I. basic approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90, 2006.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huanang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

Shengliang Deng, Mi Yan, Yixin Zheng, Jiayi Su, Wenhao Zhang, Xiaoguang Zhao, Heming Cui, Zhizheng Zhang, and He Wang. StereoVLA: Enhancing vision-language-action models with stereo vision. *arXiv preprint arXiv:2512.21970*, 2025.

Albert Enyedy, Ashay Aswale, Berk Calli, and Michael Gennert. Stereo image-based visual servoing towards feature-based grasping. In *2024 IEEE International Conference on Robotics and Automation*, pp. 7325–7331, 2024. doi: 10.1109/ICRA57147.2024.10611604.

Xianzhe Fan, Yuxiang Lu, Shenyuan Gao, Xiaoyang Wu, Ruihua Han, Manling Li, and Hengshuang Zhao. DreamAvoid: Critical-phase test-time dreaming to avoid failures in VLA policies. *arXiv preprint arXiv:2605.11750*, 2026.

Evans Han, Yunfan Jiang, Yingke Wang, Haoyue Xiao, Huang Huang, Jianwen Xie, Jiajun Wu, Li Fei-Fei, and Ruohan Zhang. StereoPolicy: Improving robotic manipulation policies via stereo perception. *arXiv preprint arXiv:2605.09989*, 2026.

Huixi Intelligence, Chen Zhang, Chenyang Zhou, Guanglei Ding, Guanghui He, Haibin Gao, Jiajia Chen, Jianyong Zhang, Lianyi Yu, Ningyi Xu, Ping Xu, Qingchen Li, Yingjun Hu, Yijia Zhang, and Yuxi Liu. RhinoVLA technical report. *arXiv preprint arXiv:2606.07383*, 2026. doi: 10.48550/arXiv.2606.07383.

Xiaosong Jia, Bowen Yang, Zuhao Ge, Xian Nie, Yuchen Zhou, Cunxin Fan, Yufeng Li, Yilin Chai, Chao Jing, Zijian Liang, et al. GuidedVLA: Specifying task-relevant factors via plug-and-play action attention specialization. *arXiv preprint arXiv:2605.12369*, 2026.

Minsoo Jo, Taeju Kwon, Junha Chun, Youngjoon Jeong, and Taesup Kim. Uncovering vulnerability of vision-language-action models under joint-level physical faults. *arXiv preprint arXiv:2606.10501*, 2026.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Chengmeng Li, Junjie Wen, Yan Peng, Yaxin Peng, Feifei Feng, and Yichen Zhu. PointVLA: Injecting the 3d world into vision-language-action models. *arXiv preprint arXiv:2503.07511*, 2025a.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Oier Mees, Karl Pertsch, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

Ye Li, Yuan Meng, Zewen Sun, Kangye Ji, Chen Tang, Jiajun Fan, Xinzhu Ma, Shutao Xia, Zhi Wang, and Wenwu Zhu. SP-VLA: A joint model scheduling and token pruning approach for VLA model acceleration. *arXiv preprint arXiv:2506.12723*, 2025b.

Tao Lin, Yuxin Du, Jiting Liu, Nuobei Zhu, Yunhe Li, Yuqian Fu, Yinxinyu Chen, Hongyi Cai, Zewei Ye, Bing Cheng, et al. Evo-Depth: A lightweight depth-enhanced vision-language-action model. *arXiv preprint arXiv:2605.14950*, 2026.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

Yanqin Ma, Xilong Liu, Juan Zhang, De Xu, Dapeng Zhang, and Wenrong Wu. Robotic grasping and alignment for small size components assembly based on visual servoing. *The International Journal of Advanced Manufacturing Technology*, 106(11):4827–4843, 2020. doi: 10.1007/s00170-019-04800-0.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Jim Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Zhifeng Rao, Wenlong Chen, Lei Xie, Xia Hua, Dongfu Yin, Zhen Tian, and F. Richard Yu. AugVLA-3D: Depth-driven feature augmentation for vision-language-action models. *arXiv preprint arXiv:2602.10698*, 2026.

Alessandro Scherl, Stefan Thalhammer, Bernhard Neuberger, Wilfried Wöber, and José García-Rodríguez. ViT-VS: On the applicability of pretrained vision transformer features for generalizable visual servoing. *arXiv preprint arXiv:2503.04545*, 2025. doi: 10.48550/arXiv.2503.04545.

Pinhao Song, Ze Fu, Yutong Hu, and Renaud Detry. Assistron: Bayesian shared autonomy with off-the-shelf vision-language-action models. *arXiv preprint arXiv:2606.23147*, 2026.

Hanzhen Wang, Jiaming Xu, Yushun Xiang, Jiayi Pan, Yongkang Zhou, Yong-Lu Li, and Guohao Dai. SpecPrune-VLA: Accelerating vision-language-action models via action-aware self-speculative pruning. *arXiv preprint arXiv:2509.05614*, 2025.

Runze Wang, Yuqian Fu, Yu Li, Tao Lin, Tianwen Qian, Mohamed Elhoseiny, Bo Zhao, Yanwei Fu, Yu-Gang Jiang, and Xiangyang Xue. Afford-VLA: Action-aligned visual planning via internalized affordance. *arXiv preprint arXiv:2605.24203*, 2026.

Zhongyu Xia, Yousen Tang, Bingqing Wei, and Yongtao Wang. 3DVLA: Enhancing vision-language-action models via 3d spatial and instance understanding. *arXiv preprint arXiv:2605.29416*, 2026.

Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Zhuoguang Chen, Tao Jiang, and Hang Zhao. DepthVLA: Enhancing vision-language-action models with depth-aware spatial reasoning. *arXiv preprint arXiv:2510.13375*, 2025.

Jianke Zhang, Xiaoyu Chen, Yanjiang Guo, Yucheng Hu, and Jianyu Chen. VLM4VLA: Revisiting vision-language-models in vision-language-action models. ICLR 2026 poster, 2026a.

Wentao Zhang, Aolan Sun, Wentao Mo, Xiaoyang Qu, Yuxin Zheng, and Jianzong Wang. From knowing to doing precisely: A general self-correction and termination framework for VLA models. *arXiv preprint arXiv:2602.01811*, 2026b.

Junjie Zhu, Huayu Liu, Jin Wang, Bangrong Wen, Kaixiang Huang, Xiaofei Li, Haiyun Zhan, and Guodong Lu. Bridging VLM and KMP: Enabling fine-grained robotic manipulation via semantic keypoints representation. *arXiv preprint arXiv:2503.02748*, 2025.

A DGA ALGORITHM AND COMPLEXITY ANALYSIS

Algorithm 1 summarizes the DGA-VLA framework.

Algorithm 1 DGA-VLA

Require: Observation o_t with depth D_t, action chunk $a_{t:t+L}$, mask M^g, ROI Ω, depth threshold τ_d, anchor $(\mathbf{q}^*, z^*)$, look-ahead horizon K, correction budget B, step bound $s_{\max}$, camera intrinsics (f_x, f_y), camera-to-base rotation $\mathbf{R}_c^b$, cooldown τ_c, open-history length W_o, and last issued gripper command g_{t-1}

persistent (m, b, r, h), initialized once per episode as $(\texttt{idle}, 0, 0, 0)$
$h = \min(h+1, W_o)$ if g_{t-1} is open; otherwise $h = 0$
if $m = \texttt{idle}$ **then**
  if $r > 0$ **then**
    $r = r - 1$; $\mathrm{VLA}(a_t)$; **return**
  end if
  if $h < W_o$ **then**
    $\mathrm{VLA}(a_t)$; **return**
  end if
  $t_c = \min\{j : a_j^g \text{ closes the gripper}\}$
  if t_c does not exist or $t_c - t > K$ **then**
    $\mathrm{VLA}(a_t)$; **return**
  end if
end if
$M_t = \mathbb{I}_\Omega \mathbb{I}[D_t \le \tau_d] \mathbb{I}[M^g = 0]$
$\{\mathcal{C}_j\} = \textsc{ConnectedComponents}(M_t)$
if no valid retained component **then**
  if $m = \texttt{align}$ **then**
    $m = \texttt{idle}, b = 0, r = \tau_c, h = 0$; $\textsc{ReplanVLA}(o_t)$; **return**
  else
    $\mathrm{VLA}(a_t)$; **return**
  end if
end if
$j^* = \arg\min_j \|\mathbf{c}_j - \mathbf{q}^*\|_2, \quad (c_u, c_v, z_t) = (\mathbf{c}_{j^*}, z_{j^*})$
if $m = \texttt{idle}$ and the target is aligned **then**
  $\mathrm{VLA}(a_t)$; **return**
end if
if $m = \texttt{idle}$ **then**
  $m = \texttt{align}, b = 0$; $\textsc{Discard}(a_{t:t+L})$
end if
if the target is aligned or $b \ge B$ **then**
  $m = \texttt{idle}, b = 0, r = \tau_c, h = 0$; $\textsc{ReplanVLA}(o_t)$; **return**
end if
$e_u = c_u - u^*, \quad e_v = c_v - v^*, \quad e_z = \max(z_t - z^*, 0)$
$\widetilde{\Delta\mathbf{p}}_t^c = [e_u z_t / f_x, e_v z_t / f_y, e_z]^\top$
$\Delta\mathbf{p}_t^b = \mathbf{R}_c^b \,\mathrm{sat}_{s_{\max}}(\widetilde{\Delta\mathbf{p}}_t^c)$
$\mathrm{DGA}(\Delta\mathbf{p}_t^b, g_{\text{open}})$; $b = b + 1$; **return**

Complexity analysis. Let the depth image have size $H \times W$, let $R = |\Omega|$, let J be the number of retained connected components, and let $C = \sum_{j=1}^{J} |\mathcal{C}_j|$. Constructing the candidate mask and finding connected components costs $O(HW + R)$, computing component statistics costs $O(C)$, and selecting the component nearest to the grasp anchor costs $O(J)$. Thus, the depth analysis has complexity

$$O(HW + R + C + J)\,,$$

and the remaining fixed-dimensional operations are $O(1)$. Thus, an intervention with at most B correction steps has total complexity

$$O(B(HW + R + C + J))\,.$$

B SIMULATION IMPLEMENTATION DETAILS

The three simulation benchmarks expose different observation, camera, and action conventions. We therefore adapt each environment separately.

B.1 ROBOTWIN

RoboTwin exposes separate `left_camera` and `right_camera` observations. For each active arm, the runtime camera configuration supplies the OpenCV intrinsic matrix `intrinsic_cv` and the OpenGL camera-to-world matrix `cam2world_gl`. DGA back-projects the image-plane error with the intrinsics, appends the approach correction along camera Z, norm-limits the complete 3D camera-frame vector, and then transforms it with the camera-to-world rotation before rebuilding the dual-arm 16-dimensional end-effector action.

RoboTwin provides a task-level `check_success()` signal but does not provide an independent grasp-success signal. We therefore define a separate episode-level grasp metric. The target object is specified per task, and after the scene settles we record the static bodies initially supporting that object. A grasp window begins when a normalized gripper command crosses from $q_g \geq 0.5$ to $q_g < 0.5$ and ends when the gripper reopens. The episode is counted as a grasp success only if, within this window, two distinct finger links of the same arm contact the target and the target remains free of all recorded static supports for 10 consecutive physics steps. Task success remains the simulator's independent task-level result.

B.2 SIMPLERENV

The original GR00T N1.7 input in SimplerEnv does not include a wrist view, so we add a wrist camera to the WidowX (Bridge) setup. It is attached to `ee_gripper_link` with offset $[-0.08, 0, 0]$ m and identity quaternion $[1, 0, 0, 0]$. The camera renders 256×256 images with $f_x = f_y = 256$ and principal point $(128, 128)$, and provides wrist-view metric depth observations to DGA. The policy continues to receive the original external view.

B.3 LIBERO

For LIBERO, the policy uses the standard `agentview` and `robot0_eye_in_hand` camera observations at 256×256 resolution. DGA obtains wrist RGB and depth from the latter. MuJoCo depth values are converted to metric camera depth using the simulator's near and far planes. For pose-guided correction, the runtime camera field of view and camera-to-base rotation are read from the MuJoCo model.

B.4 PHYSVLA REPRODUCTION IN LIBERO

The PhysVLA paper illustrates its design with a representative LIBERO task. We extend the same mechanism to all LIBERO tasks and use a safe no-op fallback when an object–target relation cannot be parsed reliably. The adapter parses the BDDL goal state to obtain an object–target pair, then uses the phase-aware finite-state machine (approach, grasp, transport, and place) with the published 0.06 m distance threshold. A selective Euler–Lagrange gate uses MuJoCo state to attenuate the phase correction when the dynamics residual is large, and the final action retains the default 5% PhysVLA blend.

Because LIBERO exposes normalized OSC actions, our adapter converts them to physical translation and rotation scales before computing the residual; its corresponding gate threshold is 100.0 rather than the paper's raw 0.05 value. Thus the comparison is a reproduction based on the published method description and a LIBERO interface adaptation, not a claim of a line-by-line official implementation.

Although PhysVLA also reports a real-robot experiment, the paper does not specify how the object and target positions and task-dependent state predicates required by its correction mechanism are obtained on physical hardware. We therefore restrict our reproduction and comparison to LIBERO, where these states are explicitly available from the simulator.

C Real-Robot Implementation Details

C.1 Deployment Setup

We deploy the Galbot G1 with RhinoVLA (Huixi Intelligence et al., 2026) and use the right-wrist-mounted Intel RealSense D405C to obtain metric RGB-D observations. Only the right arm is actuated; the left arm remains stationary. The deployed policy uses action chunks through the 72-dimensional Galbot interface: right-arm joints D7–D13 use delta-from-current-state actions, while the right gripper D15 uses an absolute closed-ratio command.

The real-robot policy outputs joint-angle actions rather than Cartesian end-effector displacements, so a camera-space alignment vector cannot be sent directly as a policy action. At an intervention, DGA acquires a new right-wrist depth frame, removes invalid returns, and extracts the target component while excluding the gripper mask. The measured pixel error is back-projected with the runtime camera intrinsics to obtain a small camera-frame translation, which is transformed to the robot base frame using the timestamped camera-to-base transform. We then construct a target pose for the right end effector and call the collision-checked Galbot inverse kinematics (IK) service, initialized from the current full-joint snapshot. Only the right-arm chain is allowed to change; the other measured chains are held fixed.

Each correction is bounded to 5 mm and the cumulative translation budget is 30 mm. Before execution, the IK result is checked for a valid status, a maximum joint change of 0.05 rad, and forward-kinematics residuals below 1 mm and 0.02 rad.

C.2 Experimental Configuration and Success Criterion

The evaluation uses 25 fixed position configurations. Each configuration places the red bag, black fan, and white foam at cells in the tabletop grid. RhinoVLA and RhinoVLA+DGA use the same position configurations. For each object-level pick-and-place rollout, we count success only when the robot grasps the specified target object and places it at the specified target location. Grasping an incorrect object, dropping the object, or exceeding the predefined rollout budget is counted as a failure.

D Additional Ablations on LIBERO

Table 6 reports additional ablations on LIBERO. OpenVLA with DGCR is identical to vanilla OpenVLA because it predicts one action at a time, so it is not reported separately. DGCS and DGCR remain below their corresponding vanilla policies. Full DGA achieves higher average success than DGA-1Step for both $\pi_{0.5}$ (97.40% vs. 97.05%) and π_0 (92.25% vs. 91.55%), consistent with the SimplerEnv results.

Table 6: **Ablation results on LIBERO.** Bold denotes the highest success rate within each VLA backbone and column.

Method	Spatial	Object	Goal	Long	Average
$\pi_{0.5}$	**98.2**	99.0	98.0	91.4	96.65
$\pi_{0.5}$ + DGCS	97.4	96.6	97.0	90.4	95.35
$\pi_{0.5}$ + DGCR	97.8	97.4	97.8	91.0	96.00
$\pi_{0.5}$ + DGA-1Step	97.0	**99.2**	98.0	94.0	97.05
$\pi_{0.5}$ + DGA (ours)	**98.2**	98.6	**98.6**	**94.2**	**97.40**
π_0	95.6	97.4	93.0	**81.2**	91.80
π_0 + DGCS	94.2	97.8	88.8	78.4	89.80
π_0 + DGCR	94.4	97.0	92.8	80.8	91.25
π_0 + DGA-1Step	96.0	96.4	**93.8**	80.0	91.55
π_0 + DGA (ours)	**96.4**	**98.2**	93.2	**81.2**	**92.25**
OpenVLA	86.6	86.6	77.2	52.6	75.75
OpenVLA + DGCS	83.8	84.6	75.0	50.4	73.45
OpenVLA + DGA (ours)	**86.8**	**87.2**	**78.2**	**55.4**	**76.90**

E Additional Rollout Visualizations

Figure 4 shows representative RoboTwin rollouts. Across these examples, DGA makes the grasp more accurate by correcting local gripper–object misalignment before closure.

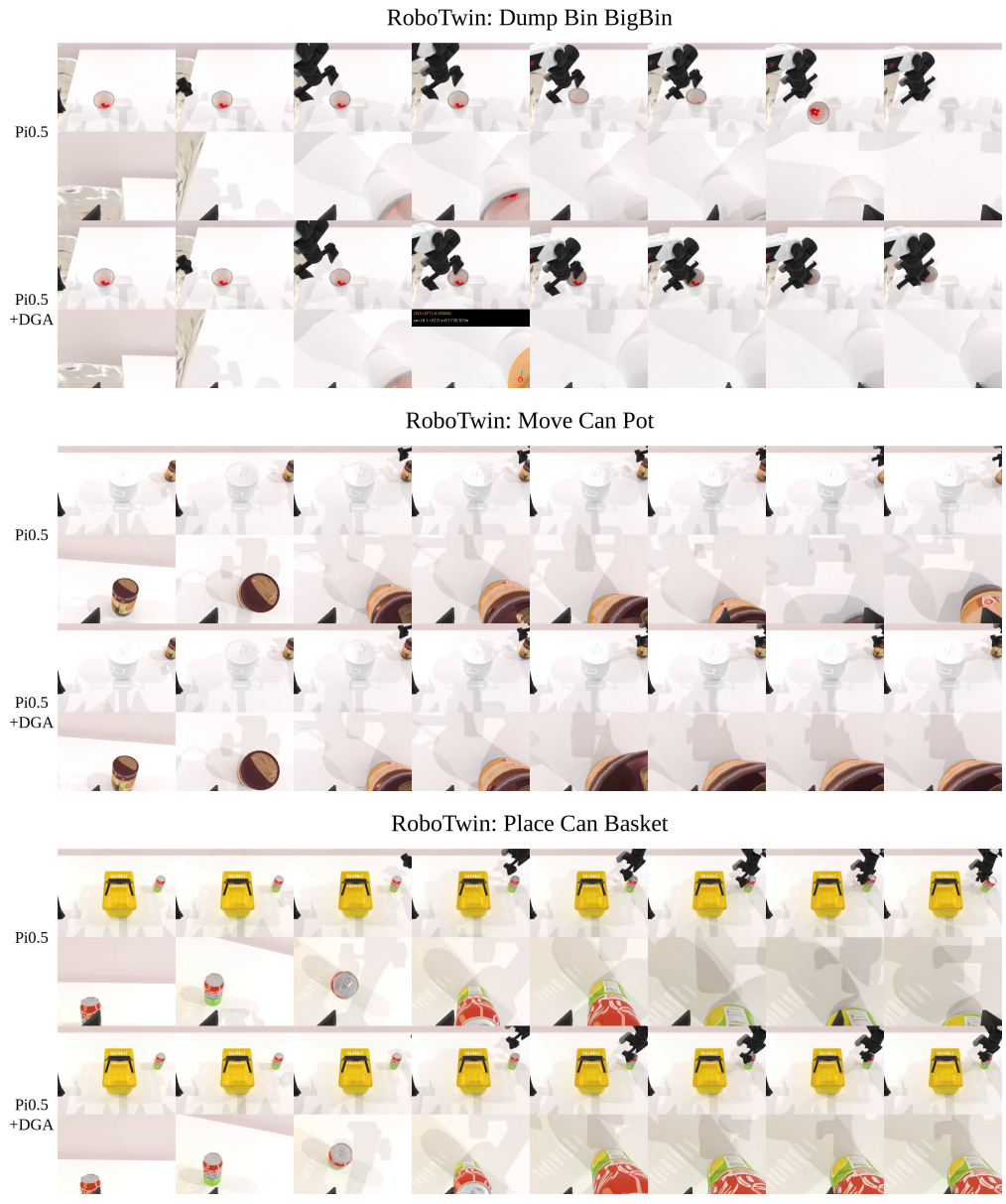


Figure 4: **RoboTwin rollout visualizations.** In each panel, the upper sequence shows the $\pi_{0.5}$ baseline and the lower sequence shows $\pi_{0.5}$ with DGA. Each sequence includes the head-camera view above the wrist-camera view. DGA improves grasp accuracy by correcting local gripper–object misalignment before closure.

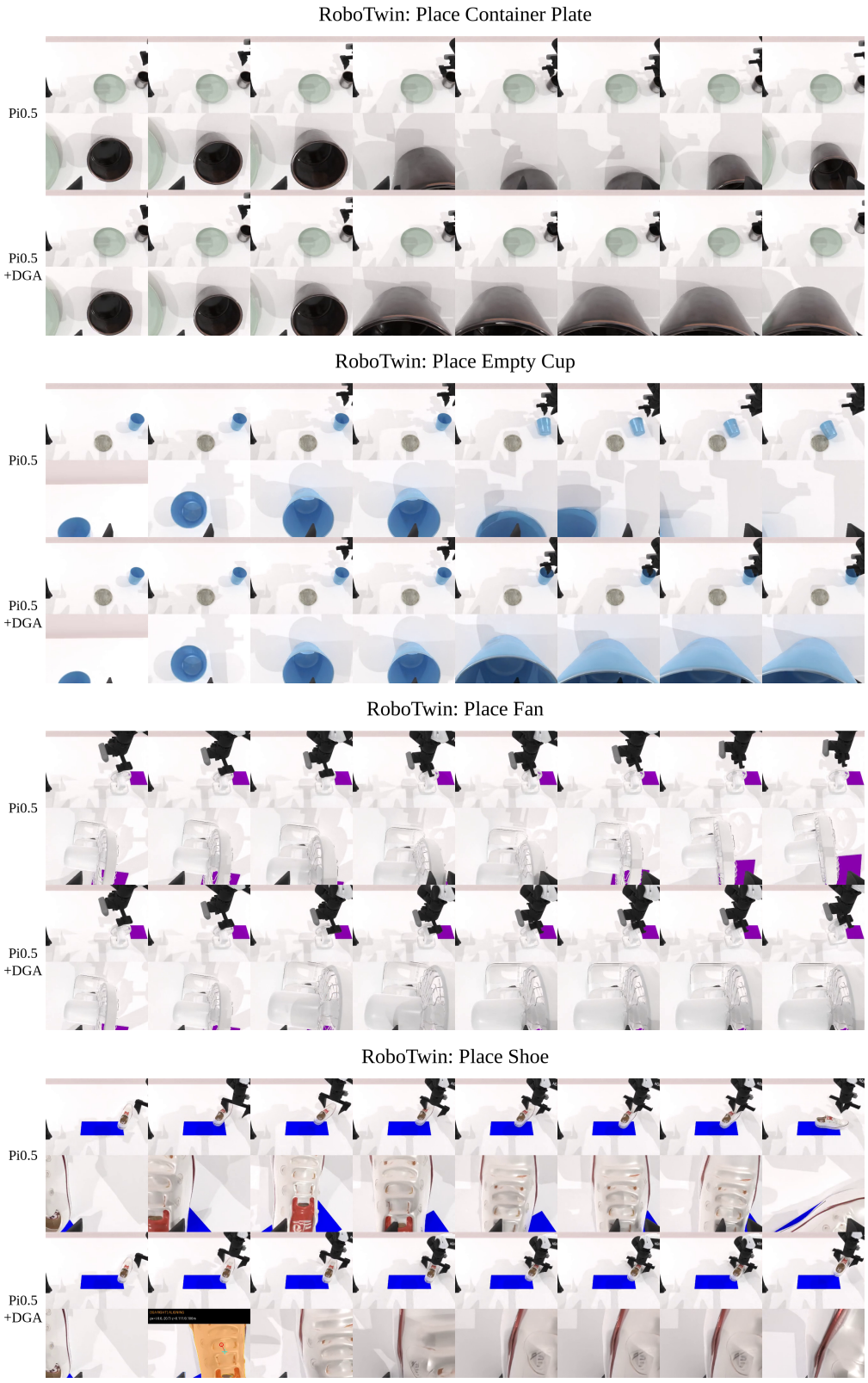


Figure 4: **RoboTwin rollout visualizations (continued).**