AimBot: A Simple Auxiliary Visual Cue to Enhance Spatial Awareness of Visuomotor Policies

Yinpei Dai[†*], **Jayjun Lee**[‡*], **Yichi Zhang**[†], **Ziqiao Ma**[†], **Jianing Yang**[†]
Amir Zadeh[◇], **Chuan Li**[◇], **Nima Fazeli**[†‡⋆], **Joyce Chai**[†⋆]
[†]Computer Science and Engineering Department, University of Michigan
[‡]Robotics Department, University of Michigan
[◇]Lambda Labs
[*] Equal Contribution [⋆] Equal Advising
{daiyp, jayjun, nfz, chaijy}@umich.edu

Abstract: In this paper, we propose **AimBot**, a lightweight visual augmentation technique that provides explicit spatial cues to improve visuomotor policy learning in robotic manipulation. **AimBot** overlays shooting lines and scope reticles onto multi-view RGB images, offering auxiliary visual guidance that encodes the end-effector's state. The overlays are computed from depth images, camera extrinsics, and the current end-effector pose, explicitly conveying spatial relationships between the gripper and objects in the scene. **AimBot** incurs minimal computational overhead (less than 1 ms) and requires no changes to model architectures, as it simply replaces original RGB images with augmented counterparts. Despite its simplicity, our results show that **AimBot** consistently improves the performance of various visuomotor policies in both simulation and real-world settings, highlighting the benefits of spatially grounded visual feedback. Codes and videos can be found at https://aimbot-reticle.github.io/

Keywords: Robotic Manipulation, Visuomotor Policy, Imitation Learning

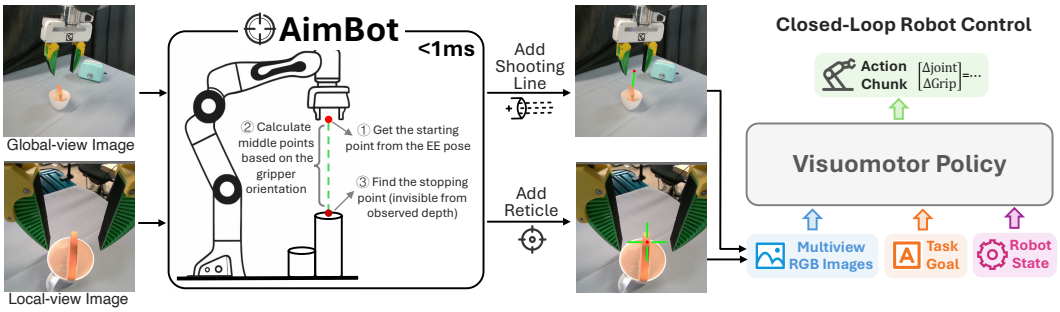

Figure 1: Overview of **AimBot**, a lightweight visual guidance method that adds spatial cues onto RGB images for visuomotor policy learning. Given the robot's end-effector pose, camera extrinsic and depth image, **AimBot** computes shooting lines and reticle overlay to highlight the spatial relationship between the gripper and objects of interest. The augmented RGB images are used as input to visuomotor policies for closed-loop control, enhancing task performance with minimal overhead.

1 Introduction

Robotic manipulation in unstructured environments demands visuomotor policies that can robustly and accurately predict continuous actions from raw RGB observations. Although recent advances, such as diffusion-based policies [1, 2] and vision-language-action (VLA) models [3, 4, 5, 6, 7], have leveraged large-scale datasets to learn complex behaviors, they often lack explicit spatial grounding

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

in visual input. As a result, these policies often exhibit limited spatial awareness of their end-effector (EE) pose and its relationship to surrounding objects [8, 9, 10, 11, 12, 13].

Motivated by the intuitive visual feedback offered by scope reticles in optical sighting systems [14, 15, 16], we present **AimBot**, a lightweight and effective visual augmentation technique that enhances spatial awareness in robotic manipulation with visual targeting cues. **AimBot** overlays auxiliary scope reticles and shooting lines onto multi-view RGB images by leveraging depth information, camera extrinsics, and the robot's EE pose. These spatial cues project the intended grasp point and the orientation of the EE onto the image plane, offering an interpretable visualization of the alignment between the gripper and objects of interest (see Figure 1). Beyond highlighting spatial relationships, these augmentations also intuitively encode the robot's proprioceptive state, making the EE position and orientation directly accessible in the visual domain.

AimBot requires no changes to model architecture and introduces negligible computational overhead (less than 1 ms), while substantially enriching the spatial information available to any visuomotor policy. Through extensive experiments in both simulation and real-world environments, we demonstrate that **AimBot** consistently improves overall task performance across diverse VLA backbones, with particularly strong gains on challenging, long-horizon tasks that demand effective spatial alignment between the EE and objects. We believe this simple yet effective approach provides a practical and scalable means to enhance learning-based robotic manipulation in 3D environments.

2 Related Works

2.1 Vision-Language-Action Models for Manipulation

Recent advances in pre-trained foundation models [17, 18, 19, 20] have catalyzed the development of vision-language-action (VLA) models, which significantly boost visuomotor policy learning in generalization, scalability, and robustness [3, 4, 21]. OpenVLA [4] and OpenVLA-OFT [22] extend the Llama model [17] to large-scale real-world robot data, achieving strong performance and effective action generation. π_0 [3], π_0-FAST [5] leverage the Gemma model [23] and compression techniques to develop generalist policies capable of handling complex manipulation tasks. GR00T [24] utilize a heterogeneous mixture of real-robot trajectories, human videos, and synthetic datasets for expressive humanoid control. FuSe [25] moves beyond data scaling by leveraging multiple sensory modalities, demonstrating that natural language can universally ground vision, touch, and sound without requiring extensive multi-modal datasets. GO-1 [26] improves long-horizon reasoning through a VQ-VAE latent action model trained on web-scale video data. All these work highlight the rapid progress in robotic manipulation, providing a foundation upon which our work can build.

2.2 Visual Augmentation/Guidance for Manipulation

With the success of visual prompting in VLMs [27, 28, 29, 30, 31, 32], recent approaches have explored visual intermediaries to enhance generalization in robotic manipulation. RT-Affordance [33] predicts affordance plans (i.e., key robot poses) conditioned on the visual input, enabling flexible learning across diverse supervision sources. RT-Trajectory [34] extends this idea by conditioning policies on coarse trajectory sketches, facilitating generalization to novel scenarios. GENIMA [35] fine-tunes a diffusion model to overlay joint-action targets onto RGB images, which are then translated into joint positions. TraceVLA [9] introduces visual trace prompting that encodes spatiotemporal trajectories directly into visual inputs, enabling VLA models to better predict actions. RoboPoint [36] leverages a VLM to predict keypoint affordances in terms of points on RGB images. HAMSTER [37] introduces a hierarchical approach that separates high-level task planning from low-level motor control using intermediate 2D path prediction. However, all those methods require online model inference during deployment, significantly limiting their practicality for real-time control. In contrast, our work presents a lightweight visual augmentation technique that overlays interpretable 2.5D spatial cues onto RGB images, offering spatial information without incurring inference costs. Moreover, by embedding EE state information directly into the pixel space, our method provides a novel and effective EE representation to enhance vision-language-action models.

3 Methodology

3.1 Method Overview

Visuomotor policies [1, 38] aim to directly map RGB camera observations and proprioceptive states to robot actions for sensorimotor control. Our goal is to enhance these policies with auxiliary visual cues, such as shooting lines and crosshair reticles, to improve spatial alignment and task success. The proposed technique, **`AimBot`**, is model-agnostic and requires no modifications to underlying policy architectures. It operates by augmenting the original multi-view RGB images with visual guidance, embedding spatial information directly into the pixel space. Fine-tuning is then performed exclusively on these new images, enabling any visuomotor policy or vision-language-action model to leverage the enhanced spatial cues.

3.2 `AimBot` Visual Guidance

Suppose we have the camera extrinsics $\mathbf{E} \in \mathbb{R}^{4\times4}$ and intrinsics $\mathbf{K} \in \mathbb{R}^{3\times3}$, along with the RGB image I and depth image D captured at the current timestep from camera c. Given a 3D point $\mathbf{p}_{\text{wld}} \in \mathbb{R}^3$ in the world frame, we can project it into camera frame with pinhole camera model [39]:

$$\begin{bmatrix}\mathbf{p}_{\text{cam}}\\ 1\end{bmatrix} = \mathbf{E}\cdot\begin{bmatrix}\mathbf{p}_{\text{wld}}\\ 1\end{bmatrix}, \quad \text{where } \mathbf{p}_{\text{cam}} = (x_c, y_c, z_c)^\top. \tag{1}$$

Then, we can continue to project $\mathbf{p}_{\text{cam}}$ to 2D image coordinates (u_c, v_c) using the intrinsic matrix:

$$\begin{bmatrix}u_c\\ v_c\\ 1\end{bmatrix} \propto \mathbf{K}\begin{bmatrix}x_c/z_c\\ y_c/z_c\\ 1\end{bmatrix}. \tag{2}$$

We define a point as *visible* if the projected pixel lies within the image bounds and the projected depth is smaller than the observed depth (i.e, not being blocked by any objects). Formally, let (u_c, v_c) be the projected pixel location and z_c the projected depth. The point is visible if:

$$0 \le u_c < W, \quad 0 \le v_c < H, \quad \text{and} \quad z_c + \epsilon < D[v_c, u_c] \tag{3}$$

where H and W are the image height and width, respectively, $D[v_c, u_c]$ is the observed depth at pixel (u_c, v_c), and $\epsilon > 0$ is a small threshold. Next, we describe how to determine the starting and stopping point of a forward line based on visible points and how to add **`AimBot`** visual guidance to RGB images. The detailed algorithm can be found in Appendix A.1.

Starting Point. We always set the origin of the gripper frame attached to the end-effector (EE) as the starting point, which is denoted as $\mathbf{p}^{ee}_{\text{wld}}$ in the world frame. The pixel location of the starting point in the image is denoted as (u^{ee}_c, v^{ee}_c) for camera c. The camera can be either fixed or movable.

Stopping Point. Starting from $\mathbf{p}^{ee}_{\text{wld}}$, we iteratively move forward along the direction vector $\mathbf{d} \in \mathbb{R}^3$, derived from the EE's orientation (e.g., the z-axis of the gripper frame):

$$\mathbf{p}^{(i+1)}_{\text{wld}} = \mathbf{p}^{(i)}_{\text{wld}} + \delta\cdot\mathbf{d}, \quad \text{with } \mathbf{p}^{(0)}_{\text{wld}} = \mathbf{p}^{ee}_{\text{wld}}.$$

where $\delta > 0$ is a small value denoting step length. At each step, we project $\mathbf{p}^{(i)}_{\text{wld}}$ to image coordinates and check visibility using the same procedure as above. The iteration stops when a certain tolerance number of invisible $\mathbf{p}^{(i)}_{\text{wld}}$ or the maximum step is reached. Then we choose the last $\mathbf{p}^{(L)}_{\text{wld}}$ as our stopping point, where L is the total iteration number. We denote the pixel location of the stopping point in the camera image as (u^{sp}_c, v^{sp}_c). The total projection distance from $\mathbf{p}^{ee}_{\text{wld}}$ to $\mathbf{p}^{(L)}_{\text{wld}}$ is $\delta L|\mathbf{d}|$.

Augment Global-View with Shooting Lines. In robotic manipulation, global-view observations are typically captured from static external cameras, such as front-facing or shoulder-mounted cameras. For these views, we overlay a shooting line on the image, extending from the pixel location corresponding to the starting point (u^{ee}_c, v^{ee}_c) to the projected stopping point (u^{sp}_c, v^{sp}_c). This line serves as an explicit visual indicator of the EE's position and orientation. In our default implementation, we also use color to convey gripper state: a green line with a red starting point indicates that the gripper is open, while a purple line with a blue starting point indicates closed gripper.

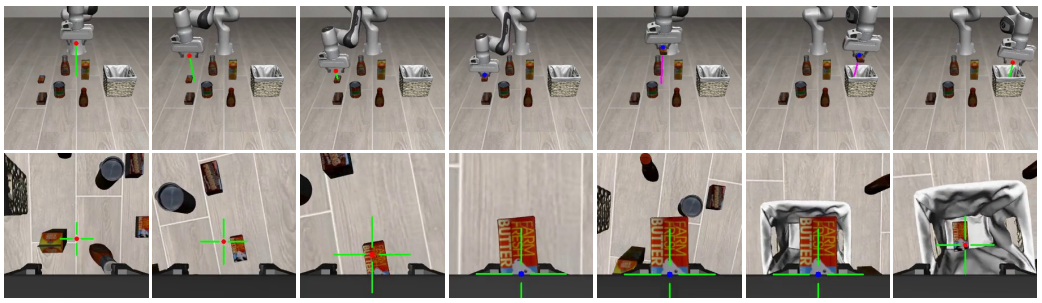

Figure 2: An example of **AimBot**-augmented LIBERO observations. We add a shooting line and a crosshair reticle for the front-view (top row) and wrist-view (bottom row) cameras, respectively.

Augment Local-View with Reticles. Wrist-mounted cameras are commonly employed to capture dynamic, close-range egocentric observations that provide detailed local views of the scene. To convey the EE's state, we overlay scope reticles onto the wrist-view image. Specifically, we render a crosshair-style reticle centered at the projected stopping point, indicating the direction in which the gripper is pointing. Note that, the pixel location $(u^{sp}_{\text{wrist}}, v^{sp}_{\text{wrist}})$ varies depending on the projection distance $\delta L|\mathbf{d}|$, which reflects the distance from the EE to the nearest surface. When the projection distance is large (e.g., the gripper is far from the table), $(u^{sp}_{\text{wrist}}, v^{sp}_{\text{wrist}})$ appears closer to the center of the wrist image due to perspective effects. Conversely, when the projection distance is small (e.g., the gripper is close to an object that obstructs the projection path), $(u^{sp}_{\text{wrist}}, v^{sp}_{\text{wrist}})$ aligns more closely with the center of the gripper pads in the image.

To encode additional information about spatial proximity, the length of the reticle lines is also modulated based on the projection distance: the reticle lines are shorter when the distance is large and longer when the distance is small, providing a visual indication of the estimated distance to the nearest orthogonal surface (see Appendix A.1 for details). This augmentation provides a visual cue that helps the policy develop a good understanding of spatial depth and distance in the local environment. In our default implementation, the crosshair lines are rendered in green, and the center point is colored red or blue to indicate whether the gripper is currently open or closed, respectively.

4 Experiments

We choose three latest vision-language-action models, π_0[3], π_0-FAST [5] and OpenVLA-OFT [22], as our visuomotor policy backbones, and conduct experiments on both simulated and real environments to evaluate our approach. For the simulation study, we choose the LIBERO [40] benchmark as the testbed. For the real-world study, we design five challenging tasks to test our approach.

4.1 Simulation Experiments

Simulation Setup: The LIBERO benchmark consists of four task suites (LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long) designed for studying lifelong learning in robotic manipulation. LIBERO-Spatial varies scene layouts with the same objects to test spatial reasoning, LIBERO-Object varies objects within fixed layouts to test object understanding, and LIBERO-Goal varies task goals while keeping objects and layouts fixed to test task-oriented behavior. LIBERO-Long includes long-horizon tasks with diverse objects, layouts, and goals, and is the most challenging suite as it demands more accurate gripper-object alignment over extended trajectories. To generate training data, we regenerate all the demonstrations following [4], and augment both multi-view RGB images with **AimBot** at a resolution of 256×256 pixels (shown in Figure 2).

Implementation Details: Following [3], we train a single multi-task policy for all tasks rather than different policies for each task. All models are optimized using AdamW [41] for 30k steps with a batch size of 32. For π_0 and π_0-FAST, we adopt a learning rate of 2e-4 with 1k warm-up steps and cosine decay. For OpenVLA-OFT, we use a fixed learning rate of 5e-4 and an L1 regression loss for action training, along with LoRA [42] optimization (rank 32, alpha 16). The action horizon is set to 50 steps for π_0, and 10 steps for the other two, following their default settings. Observations

Model	LIBERO Spatial	LIBERO Object	LIBERO Goal	LIBERO Long	Average Success Rate
OpenVLA-OFT [22]	**96.2**	97.3	93.9	87.5	93.8
OpenVLA-OFT + `AimBot`	95.2 (–1.0)	**99.1** (+1.8)	**94.2** (+0.3)	**91.2** (+3.7)	**95.0** (+1.2)
π_0-FAST [5]	96.5	**96.8**	93.6	81.6	92.1
π_0-FAST + `AimBot`	**96.9** (+0.4)	**96.8** (+0.0)	**94.0** (+0.4)	**87.1** (+5.5)	**93.7** (+1.6)
π_0 [3]	96.8	**98.8**	95.8	85.2	94.2
π_0 + `AimBot`	**96.9** (+0.1)	98.4 (–0.4)	**97.2** (+1.4)	**91.0** (+5.8)	**95.9** (+1.7)

Table 1: Performance comparison on the LIBERO simulation benchmark. Green and Red numbers indicate performance gains and losses, respectively. Each task suite averages over four runs.

Model	Fruits in Box	Tennis Ball in Drawer	Bread in Toaster	Place Coffee Cup	Egg in Carton	Total Success
OpenVLA-OFT	7/10	6/10	4/10	2/10	2/10	21/50
OpenVLA-OFT + `AimBot`	**9/10**	**7/10**	**9/10**	**8/10**	**3/10**	**36/50**
π_0-FAST	**10/10**	**10/10**	9/10	7/10	6/10	42/50
π_0-FAST + `AimBot`	**10/10**	**10/10**	**10/10**	**9/10**	**8/10**	**47/50**
π_0	7/10	7/10	4/10	5/10	4/10	27/50
π_0 + `AimBot`	**10/10**	**10/10**	**7/10**	**8/10**	**8/10**	**43/50**
Other baseline settings						
π_0 + Traces [9]	8/10	8/10	5/10	2/10	2/10	25/50
π_0 + RoboPoint [36]	8/10	9/10	4/10	6/10	0/10	27/50
π_0 + Depth Images	7/10	9/10	5/10	7/10	4/10	32/50

Table 2: Performance comparison on five real-world tasks. Each task is evaluated over ten trials.

consist of front-view and wrist-view RGB images, along with proprioceptive states provided by the simulator. All models operate in a delta Cartesian action space (6 dimensions) comprising changes in EE position and EE orientation represented in Euler angles. An additional dimension is used to represent the gripper open/close action. During evaluation, we execute 5 steps per action prediction for π_0 and π_0-FAST, and 8 steps for OpenVLA-OFT, enabling closed-loop control.

Experimental Results. For baselines trained without `AimBot`, we re-implement all models on the original LIBERO benchmarks and report the higher results between our re-implementations and their released numbers. The overall results are summarized in Table 1. Across all three backbone models, integrating and finetuning with `AimBot` consistently improves or matches baseline performance in the majority of cases, with particularly large gains on the hardest LIBERO-Long tasks. For instance, π_0-FAST improves from 81.6 to 87.1 and π_0 improves from 85.2 to 91.0 with `AimBot`. OpenVLA-OFT also achieves a notable improvement from 87.5 to 91.2 on LIBERO-Long. This indicates that the spatially grounded visual guidance provided by `AimBot` can enhance long-horizon manipulation. For other task suites, where baseline performances are already relatively high, `AimBot` leads to smaller improvements. Nevertheless, the average success rates still increase: +1.2 for OpenVLA-OFT, +1.6 for π_0-FAST, and +1.7 for π_0. Overall, `AimBot` provides the greatest benefits in more challenging, long-horizon settings while offering modest gains in easier tasks. To further study the impact of reticle design choices, we compare several `AimBot` variants in Appendix A.2 and adopt the best-performing setting (same setting as shown in Figure 2) for real-world experiments.

4.2 Real World Experiments

Real World Setup. We conduct our experiments using a 7-DoF Franka Emika Panda robot equipped with a pair of UMI fin-ray fingers [43] mounted on the default Franka hand gripper. The robot operates in a tabletop environment with three RGB-D cameras similar to the DROID setting [2]: two Intel RealSense D435 cameras each positioned on the left and right shoulder, and one Intel RealSense D405 camera mounted on the wrist. The shoulder cameras are calibrated prior to experimentation to obtain accurate camera extrinsics, while the extrinsics of the wrist-mounted camera

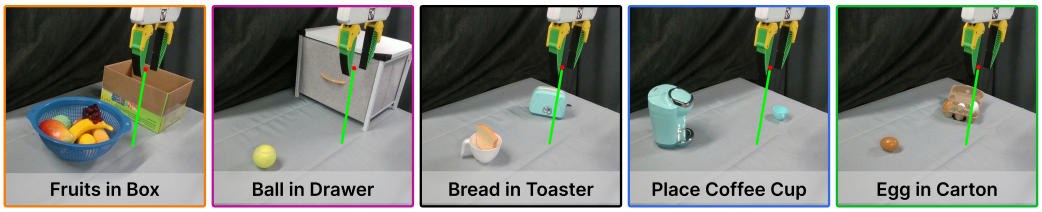

Figure 3: We design five contact-rich, long-horizon real-world tasks for policy evaluation.

are dynamically computed based on the end-effector (EE) pose. For the depth images, we use the built-in Intel RealSense SDK to preprocess and align the depth images to the RGB images.

Task Design and Data Collection. We designed five challenging tasks for the real world experiment: (1) **`Fruits in Box`**, where the robot is tasked to move all fruits inside a basket into a box; (2) **`Tennis Ball in Drawer`**, where the robot needs to open a drawer, place a tennis ball inside, and close the drawer; (3) **`Bread in Toaster`**, where the robot has to grasp a bread and insert into a narrow toaster slot; (4) **`Place Coffee Cup`**, where the robot must grab a coffee cup by its handle, reorient it, and place it onto a coffee machine; and (5) **`Egg in Carton`**, where the robot needs to pick up one or more eggs, place them into an egg carton, and firmly close the carton lid. These tasks vary in length, complexity, and contact-richness, and require a combination of prehensile (e.g., pick-and-place) and non-prehensile (e.g., pulling a drawer handle, closing a carton lid) skills. Figure 3 illustrates the table settings, and Appendix A.3 details the policy rollout as sequences of RGB images augmented by **`AimBot`** for each task. For the real-world data collection, we teleoperate the robot using an Oculus Quest 2 device, recording demonstrations at 15 Hz. We collect 80–150 demonstrations per task, resulting in a total of 548 episodes with 166k timesteps of data samples.

Implementation Details. Following the simulation study, we fine-tune π_0, π_0-FAST, and OpenVLA-OFT to evaluate the effectiveness of **`AimBot`** in real-world settings. We compare models trained on raw RGB images from multiple camera streams with models trained on RGB images augmented with **`AimBot`**. All models are optimized for 50k steps with a batch size of 32. We use a learning rate of 1e-4 for π_0 and π_0-FAST, which we found to perform better, and set the action horizon to 10 steps for all models. Other hyperparameters are kept consistent with the simulation experiments. Observations include left-shoulder, right-shoulder, and wrist RGB images, along with the robot's proprioceptive state. In terms of action space, π_0 and π_0-FAST predict delta joint angles (7 dimensions), while OpenVLA-OFT predicts delta Cartesian actions (6 dimensions); all models additionally output a binary gripper open/close command as an extra dimension. During online execution, we found that executing 8 steps per action prediction for π_0 and π_0-FAST, and 2 steps for OpenVLA-OFT, yields better performance.

Baselines. To compare **`AimBot`** with other visual guidance methods, we evaluate two open-sourced baselines: RoboPoint [36] and TraceVLA [9], by training π_0 on images augmented using their respective strategies. RoboPoint predicts spatial affordances as 2D pixel coordinates on semantically relevant regions of the image, while TraceVLA visualizes temporal motion history through colored arrows (i.e., "traces"), offering cues about past trajectories. For RoboPoint, we query the robopoint-v1-vicuna-v1.5-13b checkpoint, using the prompt *"The task is* `<task_goal>`*. Find relevant points on the image to perform the task."*, where `<task_goal>` corresponds to the language description of each real-world task. We apply their default visualization tool to overlay red crosses at the predicted affordance points onto the RGB images as the visual augmentation for policy training. For TraceVLA, we apply their co-tracker model to track objects across three camera views and overlay the resulting traces onto the images. Example comparisons of the visual augmentations are shown in Figure 4. In addition, we compare against π_0 trained with both raw RGB images and depth images to assess the effect of directly incorporating depth information. Concretely, we convert the depth image in grayscale RGB and treat it as three more images (a total of six images) to train π_0.

Experimental Results. Table 2 reports task success numbers (out of 10 trials per task) across five real-world manipulation tasks using different policy and visual guidance configurations. Overall, **`AimBot`** significantly improves performance across all models and tasks. For example, OpenVLA-

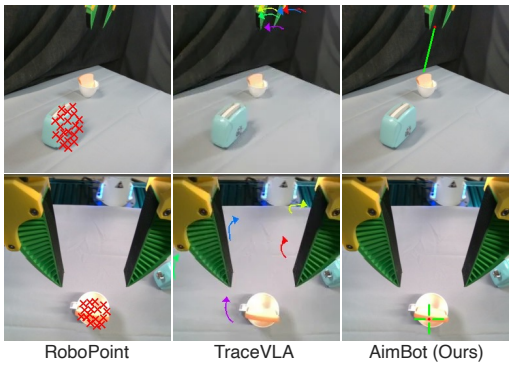

Figure 4: Comparison of different visual guidance methods.

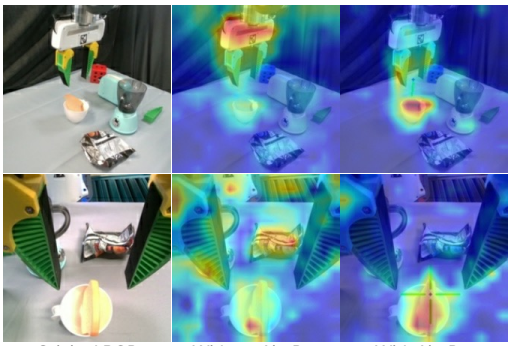

Figure 5: Visualization of attention weights trained with and without **AimBot**.

OFT's total success increases from 21 to 36 successful trials when augmented with **AimBot**, with significant gains in more challenging tasks such as **Place Coffee Cup** and **Bread in Toaster**. Similarly, π_0 improves from 27 to 43 successful trials, while π_0-FAST achieves the highest overall performance with 47 successful trials when combined with **AimBot**, outperforming all other models.

In contrast, alternative visual guidance baselines underperform: π_0 trained with RoboPoint and TraceVLA reaches only 27 and 25 successful trials, respectively, similar to the π_0 trained without any guidance. Unlike these methods, **AimBot** provides direct spatial targeting by encoding the end-effector (EE) state as shooting lines and reticles, offering richer and clearer spatial cues without occluding objects or omitting gripper state information. Additionally, both TraceVLA and RoboPoint require online model inference, introducing significant computational overhead that limits their practicality for real-time robot control. On average, TraceVLA requires approximately 0.3 seconds to process a single image, while RoboPoint takes over 5 seconds. In contrast, **AimBot** is highly efficient, requiring less than 1 ms per image, making it suitable for real-world deployment.

While using raw depth images as additional visual input yields a modest improvement over RGB-only inputs (increasing success from 27 to 32), it still falls short of **AimBot**'s performance (43), likely due to noise and inconsistency in real-world depth sensing. Despite extensive preprocessing, real-world depth data remains highly noisy and unreliable. In contrast, **AimBot** is inherently robust and less redundant, as it only compares the projected point depth with the camera depth to determine visibility for visual cue generation, without relying on complete or very accurate depth information.

4.3 Further Analysis

To better understand how **AimBot**'s visual guidance influences the model's internal representations, spatial reasoning, and generalization to unseen scenarios, we investigate the following questions:

Does AimBot enhance the robot's spatial awareness for object alignment? To address this, we first examine how training with **AimBot** shapes the model's attention toward task-relevant objects. Specifically, we feed the same input image into the OpenVLA-OFT language model backbone and extract the attention weights from Layer 1, Head 11. We then compute the summed attention scores from the action head to the input RGB image patches and upsample them to the original image resolution to generate the heatmaps shown in Figure 5. As illustrated, models trained with **AimBot** exhibit more concentrated attention on the task-relevant objects, which facilitates better task understanding and execution. In contrast, the baseline model without **AimBot** shows dispersed attention across the entire scene, making it harder for the robot to align accurately with target objects. More visualization examples can be found in Appendix A.4.

To further investigate **AimBot**'s impact on spatial alignment, we conduct a failure and error analysis based on real-world robot experiments. We categorize all failure trials into different misalignment types: grasping position misalignment, grasping orientation misalignment, placing position misalignment, placing orientation misalignment, and other non-misalignment failures (e.g., getting stuck, failed non-prehensile skills, or performing actions in a wrong task order). Examples of fail-

Misalignment Type	w/o AimBot	w/ AimBot
Grasping Position	22	7
Grasping Orientation	6	0
Placing Position	18	7
Placing Orientation	3	3
Other Failures	11	7

Table 3: Total failure counts across different misalignment types with and without **AimBot**.

Models	LIEBRO-Long
π_0 + **AimBot**	91.0
π_0 + **AimBot** − proprio.	88.0
π_0	85.2
π_0 − proprio.	83.2
π_0 + **AimBot** (random)	77.4

Table 4: Ablation of π_0 variants with or without **AimBot** and proprioceptive states.

ures are illustrated in Appendix A.5. The aggregated results for three VLA models are summarized in Table 3. As shown, models trained with **AimBot** demonstrate a substantial reduction in misalignment-related failures, particularly in grasping. This indicates that **AimBot** effectively enhances the model's spatial understanding and improves alignment capability during manipulation.

Does AimBot provide a better representation than proprioceptive state encoding? To investigate this question, we conduct an ablation study using π_0 on the LIBERO benchmark. The original π_0 setup includes a proprioceptive state vector encoding the end-effector state. We introduce two ablated variants: (1) π_0 + **AimBot** − proprio., where **AimBot** is incorporated while the proprioceptive input is replaced with a zero vector; and (2) π_0 − proprio., where neither **AimBot** nor proprioceptive input is used, with the proprioceptive state similarly zeroed out. Table 4 reports the performance on the LIBERO-Long task suite (more results are given in Appendix A.6). As shown, removing proprioceptive input reduces task success to 83.2%, compared to 85.2% when using proprioception alone. Incorporating **AimBot** without proprioception improves performance to 88.0%, and combining both leads to the highest task success of 91.0%, suggesting that **AimBot** provides a strong alternative representation and offers complementary benefits to standard proprioceptive state encoding.

To further validate that **AimBot** effectively encodes useful spatial priors into the visual input, we introduce an additional variant, π_0 + **AimBot** (random), where the auxiliary visual cues (shooting lines and reticles) are deliberately randomized by adding noise and misaligned from the true end-effector pose during both training and testing. As shown in Table 4, this model achieves a significantly lower success rate of 77.4%, compared to 91.0% with correctly aligned **AimBot**. This performance gap indicates that the spatial visual guidance provided by **AimBot** is not only interpretable but also meaningfully enhances the model's spatial awareness and downstream policy performance.

Can AimBot improve generalization to out-of-distribution (OOD) scenarios? We evaluate the ability of **AimBot** to enhance OOD generalization in real-world tasks by introducing several distribution shifts at test time, including changes in object and receptacle heights, variations in table background colors, and lighting conditions that differ significantly from the training environment (examples can be found in Appendix A.7). For each task, we conduct 3 trials, resulting in a total of 15 evaluation episodes using the π_0-FAST model. In this experiment, our method achieves **12** successful trials, whereas the baseline model achieves only 7. This suggests that **AimBot** provides robust visual cues that remain effective under distribution shifts, allowing the model to leverage reliable spatial guidance even in unseen environments and thus improving generalization performance.

5 Conclusion and Discussions

We present **AimBot**, a lightweight visual augmentation technique that enhances visuomotor policy learning by embedding spatial cues, such as end-effector position, orientation, and grasp state, into RGB observations through scope reticles and shooting lines. This augmentation improves task performance by depicting spatial context directly in pixel space, offering grounded 2.5D visual guidance without requiring any changes to the model architecture. Experiments in both simulated and real-world environments demonstrate the effectiveness and generalizability of **AimBot** across various vision-language-action backbones. In the future, we plan to enrich **AimBot** with more semantic cues, such as object segmentation and affordance labels, and extend its applicability to diverse embodiments, including bi-manual and hand robots, to address more dexterous manipulation tasks.

Acknowledgments

This work was supported in part by NSF SES-2128623, NSF CAREER #2337870, and NSF NRI #2220876. We would like to thank Dr. Xinyi Wang for insightful discussions and Dr. Mark Van der Merwe for suggesting the cool model name. We also thank Lambda Labs for providing access to GH200 computing resources.

Limitations

While effective in our experimental setup using a Panda robot with a parallel-jaw gripper, `AimBot` has several limitations:

- **Dependence on depth sensing.** `AimBot` relies on metric depth information that is aligned with RGB images. Consequently, it requires either an RGB-D sensor or a reliable monocular depth estimation model. Our method is highly efficient ($<$1ms) when sensor depth is available, while remaining compatible with model-predicted depth in RGB-only settings, but needs to afford online depth model inference overhead.
- **Assumption of nearby surfaces.** The visual guidance provided by `AimBot`, such as shooting lines and reticles, assumes the end-effector is oriented toward a nearby or enclosed surface (e.g., tabletop tasks). In open-space scenarios, such as manipulation in mid-air or above empty space, these visual cues may project onto distant regions, reducing their utility as effective spatial indicators.
- **Limited generalization to high-DOF effectors.** `AimBot` is most suitable for simple end-effectors with limited degrees of freedom, such as parallel-jaw grippers. Extending the method to more dexterous end-effectors (e.g., anthropomorphic hands) would require complex designs to visually represent finger states and contact points, which may be nontrivial and reduce the method's generalizability to high-DOF manipulation tasks.
- **Reduced effectiveness in constrained motion settings.** In tasks where the end-effector operates within a small spatial region and constantly holds an object (e.g., in-hand manipulation using tools), the projection length of the guidance cues becomes minimal. As a result, the encoded spatial information is limited and may provide little additional benefit to the visuomotor policy.

References

[1] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024.

[2] 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.

[3] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. Pi0: A vision-language-action flow model for general robot control. Physical Intelligence, 2024. URL https://www.physicalintelligence.company/download/pi0.pdf. Accessed: 2025-04-06.

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

[5] K. Pertsch, K. Stachowicz, B. Ichter, D. Driess, S. Nair, Q. Vuong, O. Mees, C. Finn, and S. Levine. Fast: Efficient action tokenization for vision-language-action models, 2025. URL https://arxiv.org/abs/2501.09747.

[6] C.-Y. Hung, Q. Sun, P. Hong, A. Zadeh, C. Li, U. Tan, N. Majumder, S. Poria, et al. Nora: A small open-sourced generalist vision language action model for embodied tasks. *arXiv preprint arXiv:2504.19854*, 2025.

[7] F. Lin, R. Nai, Y. Hu, J. You, J. Zhao, and Y. Gao. Onetwovla: A unified vision-language-action model with adaptive reasoning. *arXiv preprint arXiv:2505.11917*, 2025.

[8] D. Qu, H. Song, Q. Chen, Y. Yao, X. Ye, Y. Ding, Z. Wang, J. Gu, B. Zhao, D. Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

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

[10] C. Li, J. Wen, Y. Peng, Y. Peng, F. Feng, and Y. Zhu. Pointvla: Injecting the 3d world into vision-language-action models. *arXiv preprint arXiv:2503.07511*, 2025.

[11] C. H. Song, V. Blukis, J. Tremblay, S. Tyree, Y. Su, and S. Birchfield. Robospatial: Teaching spatial understanding to 2d and 3d vision-language models for robotics. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 15768–15780, 2025.

[12] E. Zhou, J. An, C. Chi, Y. Han, S. Rong, C. Zhang, P. Wang, Z. Wang, T. Huang, L. Sheng, et al. Roborefer: Towards spatial referring with reasoning in vision-language models for robotics. *arXiv preprint arXiv:2506.04308*, 2025.

[13] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman, et al. Foundation models in robotics: Applications, challenges, and the future. *The International Journal of Robotics Research*, page 02783649241281508, 2023.

[14] B.-S. Jeung, S.-S. Lim, and D.-H. Lee. Development of optical sighting system for moving target tracking. *Current Optics and Photonics*, 3(2):154–163, 2019.

[15] C. Zhang, J. Sun, Q. Hou, B. Chen, H. Yao, Z. Zhou, and J. Cong. Augmented reality enabled galilean sighting system with an adjustable reticle. *Applied Optics*, 60(18):5387–5391, 2021.

[16] S. Ledin. Advanced riflescope reticle guide, 2021. URL https://www.opticsplanet.com/howto/how-to-advanced-reticle-guide.html.

[17] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[18] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[19] W. He, Y. Dai, Y. Zheng, Y. Wu, Z. Cao, D. Liu, P. Jiang, M. Yang, F. Huang, L. Si, et al. Galaxy: A generative pre-trained model for task-oriented dialog with semi-supervised learning and explicit policy injection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 10749–10757, 2022.

[20] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[21] J. Xi, Y. He, J. Yang, Y. Dai, and J. Chai. Teaching embodied reinforcement learning agents: Informativeness and diversity of language use. In Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4097–4114, Miami, Florida, USA, Nov. 2024. Association for Computational Linguistics. doi:10.18653/v1/2024.emnlp-main.237. URL https://aclanthology.org/2024.emnlp-main.237/.

[22] M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success, 2025. URL https://arxiv.org/abs/2502.19645.

[23] G. Team. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

[24] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[25] J. Jones, O. Mees, C. Sferrazza, K. Stachowicz, P. Abbeel, and S. Levine. Beyond sight: Finetuning generalist robot policies with heterogeneous sensors via language grounding. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[26] Q. Bu, J. Cai, L. Chen, X. Cui, Y. Ding, S. Feng, S. Gao, X. He, X. Huang, S. Jiang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

[27] J. Yang, H. Zhang, F. Li, X. Zou, C. Li, and J. Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*, 2023.

[28] A. Shtedritski, C. Rupprecht, and A. Vedaldi. What does clip know about a red circle? visual prompt engineering for vlms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11987–11997, 2023.

[29] Y. Dai, J. Lee, N. Fazeli, and J. Chai. Racer: Rich language-guided failure recovery policies for imitation learning. *arXiv preprint arXiv:2409.14674*, 2024.

[30] Y. Dai, R. Peng, S. Li, and J. Chai. Think, act, and ask: Open-world interactive personalized robot navigation. In *2024 IEEE international conference on robotics and automation (ICRA)*, pages 3296–3303. IEEE, 2024.

[31] S. Nasiriany, F. Xia, W. Yu, T. Xiao, J. Liang, I. Dasgupta, A. Xie, D. Driess, A. Wahid, Z. Xu, et al. Pivot: Iterative visual prompting elicits actionable knowledge for vlms. *arXiv preprint arXiv:2402.07872*, 2024.

[32] F. Li, Q. Jiang, H. Zhang, T. Ren, S. Liu, X. Zou, H. Xu, H. Li, J. Yang, C. Li, et al. Visual in-context prompting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12861–12871, 2024.

[33] S. Nasiriany, S. Kirmani, T. Ding, L. Smith, Y. Zhu, D. Driess, D. Sadigh, and T. Xiao. Rt-affordance: Affordances are versatile intermediate representations for robot manipulation, 2024. URL https://arxiv.org/abs/2411.02704.

[34] J. Gu, S. Kirmani, P. Wohlhart, Y. Lu, M. G. Arenas, K. Rao, W. Yu, C. Fu, K. Gopalakrishnan, Z. Xu, P. Sundaresan, P. Xu, H. Su, K. Hausman, C. Finn, Q. Vuong, and T. Xiao. Rt-trajectory: Robotic task generalization via hindsight trajectory sketches, 2023.

[35] M. Shridhar, Y. L. Lo, and S. James. Generative image as action models. In *Proceedings of the 8th Conference on Robot Learning (CoRL)*, 2024.

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

[37] Y. Li, Y. Deng, J. Zhang, J. Jang, M. Memmel, R. Yu, C. R. Garrett, F. Ramos, D. Fox, A. Li, A. Gupta, and A. Goyal. Hamster: Hierarchical action models for open-world robot manipulation, 2025. URL https://arxiv.org/abs/2502.05485.

[38] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[39] D. A. Forsyth and J. Ponce. *Computer vision: a modern approach*. prentice hall professional technical reference, 2002.

[40] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

[41] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[42] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[43] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[44] J. Gildenblat and contributors. Pytorch library for cam methods. https://github.com/jacobgil/pytorch-grad-cam, 2021.

A Appendix

A.1 `AimBot` Algorithm

Algorithm 1 WLD2IMG

Require: 3D point $\mathbf{p}_{\text{wld}}$, extrinsic matrix $\mathbf{E} \in \mathbb{R}^{4\times4}$ and intrinsic matrix $\mathbf{K} \in \mathbb{R}^{3\times3}$ of camera c
Ensure: 2D image coordinates (u_c, v_c) and depth z_c

Extract rotation $\mathbf{R} \leftarrow \mathbf{E}[0{:}3,\ 0{:}3]$, translation $\mathbf{T} \leftarrow \mathbf{E}[0{:}3,\ 3]$
$\mathbf{p}_{\text{cam}} \leftarrow \mathbf{R} \cdot \mathbf{p}_{\text{wld}} + \mathbf{T}$ ▷ Transform to camera frame
$(x_c,\ y_c,\ z_c) \leftarrow \mathbf{p}_{\text{cam}}$
Extract intrinsics: $f_x \leftarrow \mathbf{K}_{00}$, $f_y \leftarrow \mathbf{K}_{11}$, $c_x \leftarrow \mathbf{K}_{02}$, $c_y \leftarrow \mathbf{K}_{12}$
$u_c \leftarrow \left\lfloor \frac{f_x x_c}{z_c} + c_x \right\rfloor$
$v_c \leftarrow \left\lfloor \frac{f_y y_c}{z_c} + c_y \right\rfloor$
return (u_c, v_c, z_c)

Algorithm 2 CHECKVISIBILITY

Require: Pixel coordinates (u_c, v_c), depth z_c, depth map D, image size (H, W), threshold ϵ

if $z_c \leq 0$ **then**
    return False ▷ Behind the camera
else if $u \notin [0, W)$ or $v \notin [0, H)$ **then**
    return False ▷ Out of image bounds
end if
if $D[v_c, u_c] \leq z_c + \epsilon$ **then**
    return False ▷ Occluded by closer object
else
    return True ▷ Visible
end if

Algorithm 3 FINDSTOPPOINT

Require: Starting position $\mathbf{p}$, orientation quaternion q, camera extrinsics $\mathbf{E}$, intrinsics $\mathbf{K}$, depth map D, image size (H, W), step size δ, tolerance number N

Convert q to direction vector $\mathbf{d}$
Initialize empty lists points3D, points2D, visibility; Initialize tolerance count $n = 0$
while step along $\mathbf{d}$ within 2 meter **do**
    $\mathbf{p} \leftarrow \mathbf{p} + \delta \cdot \mathbf{d}$
    $(u, v, z) \leftarrow$ WLD2IMG$(\mathbf{p}, \mathbf{E}, \mathbf{K})$
    $vis \leftarrow$ CHECKVISIBILITY(u, v, x, D, H, W)
    if $n < N$ and vis is False **then**
        $n \leftarrow n + 1$
    else if $n >= N$ **then**
        break
    end if
    Append $\mathbf{p}$ to points3D, (u, v) to points2D, vis to visibility
end while
return (points3D, points2D, visibility)

Algorithm 4 RENDERONFIXEDCAMERA

Require: RGB image I, depth D, extrinsics $\mathbf{E}$, intrinsics $\mathbf{K}$, gripper state $(\mathbf{p}, q, open)$, image size (H, W)

$(u, v, z) \leftarrow$ WLD2IMG$(\mathbf{p}, \mathbf{E}, \mathbf{K})$
if CHECKVISIBILITY(u, v, z, D, H, W) is not visible **then**
    return I
end if
$(\_, \text{points2D}, \text{visibility}) \leftarrow$ FINDSTOPPOINT$(\mathbf{p}, q, \mathbf{E}, \mathbf{K}, D, H, W)$
if $open$ **then**
    Draw green line on I over longest visible span using points2D
else
    Draw purple line on I through all visible spans using points2D
end if
Draw center dot at (u, v) with color based on $open$
return I

Algorithm 5 RENDERONWRISTCAMERA

Require: Wrist RGB I_w, depth D_w, extrinsics $\mathbf{E}_w$, intrinsics $\mathbf{K}_w$, gripper state $(\mathbf{p}, q, open)$, image size (H, W), MaxEEtoSurfaceDistance, MinReticleLength, MaxReticleLength

$(u_w, v_w, z_w) \leftarrow$ WLD2IMG$(\mathbf{p}, \mathbf{E}_w, \mathbf{K}_w)$
if CHECKVISIBILITY$(u_w, v_w, z_w, D_w, H, W)$ is not visible **then**
    return I_w
end if
$(\_, \text{points2D}, \text{visibility}) \leftarrow$ FINDSTOPPOINT$(\mathbf{p}, q, \mathbf{E}_w, \mathbf{K}_w, D_w, H, W)$
Compute *center* $\leftarrow$ last visible point in points2D or (u_w, v_w)
Compute *dynamic line length* with:

$$\textit{scaling} \leftarrow \max\left(\frac{\text{MaxEEtoSurfaceDistance} - z_w}{\text{MaxEEtoSurfaceDistance}},\ 0\right)$$

$$\textit{line\_length} \leftarrow \text{MinReticleLength} + \textit{scaling} \times (\text{MaxReticleLength} - \text{MinReticleLength})$$

Add crosshair centered at *center* with *line_length*, adapting color based on $open$
return I_w

A.2 Ablation Settings

To further investigate how different designs of **AimBot** reticles affect task performance, we conduct a comprehensive comparison of several visual cue variants. Specifically, we evaluate π_0 and π_0-FAST under the following settings:

1. W/ PLAIN COLOR: using a uniform gray color for the augmented guidance;
2. W/ GRASP SENSE: detecting whether an object is in between the gripper fingers and changing the reticle color to indicate a successful grasp;
3. W/ FIXED LENGTH: rendering crosshair reticles with a fixed line length instead of adapting to the estimated depth;
4. W/ SMALL SCALE: reducing the size and thickness of the visual cues;
5. W/ BULLSEYE STYLE: replacing the crosshair with a classical bullseye (concentric circle) reticle design;

Figure 6 illustrates different ablated settings of our **AimBot** augmentation. Results of π_0-FAST and π_0 are given in Table 5 and Table 6, respectively. As we can see, the default setting achieve the best performance in average. Interestingly, even with plain color, the model can still achieve adequate performance, indicating the importance of visual cues for spatial information.

AimBot Setting	LIBERO Spatial	LIBERO Object	LIBERO Goal	LIBERO Long	Avg. Success
Default	97.2	96.5	**94.0**	**87.1**	**93.7**
W/ PLAIN COLOR	**97.4**	97.6	93.2	84.0	93.1
W/ GRASP SENSE	95.8	97.0	93.6	81.6	92.0
W/ FIXED LENGTH	95.8	**97.8**	90.8	84.6	92.3
W/ SMALL SCALE	94.6	**97.8**	89.6	85.2	91.8
W/ BULLSEYE STYLE	94.8	96.8	91.0	82.4	91.3

Table 5: Ablation Study for π_0-FAST for different **AimBot** settings.

AimBot Setting	LIBERO Spatial	LIBERO Object	LIBERO Goal	LIBERO Long	Avg. Success
Default	**96.9**	**98.4**	**97.2**	**91.0**	**95.9**
W/ PLAIN COLOR	96.8	97.2	96.8	89.0	95.0
W/ GRASP SENSE	95.0	98.0	95.4	86.8	93.8
W/ FIXED LENGTH	96.0	96.2	94.6	87.2	93.5
W/ SMALL SCALE	95.0	98.0	95.4	86.8	93.8
W/ BULLSEYE STYLE	97.2	96.2	93.2	86.4	93.3

Table 6: Ablation Study for π_0 for different **AimBot** settings.

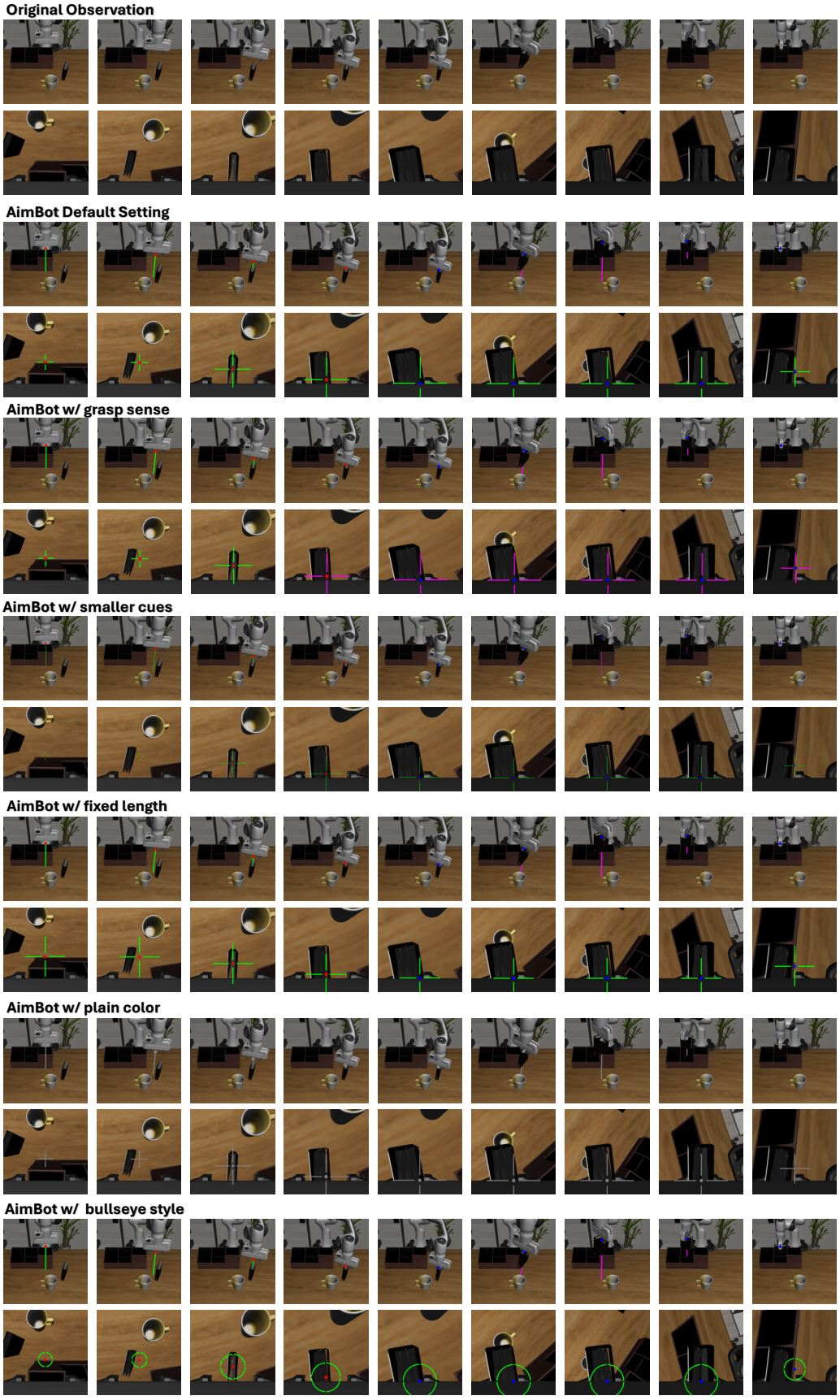

Figure 6: Comparison of different `AimBot` variants.

A.3 `AimBot` Real World Set-up and Dataset Statistics

Figure 7 shows the Franka Emika Panda robot setup used in our real-world experiments. Table 7 provides detailed statistics of the collected dataset, including language goals, number of demonstrations, required skills, and average episode length in environment timesteps. Additionally, Figure 8 presents sample rollouts for each real-world task, showcasing the use of **`AimBot`** visual guidance.

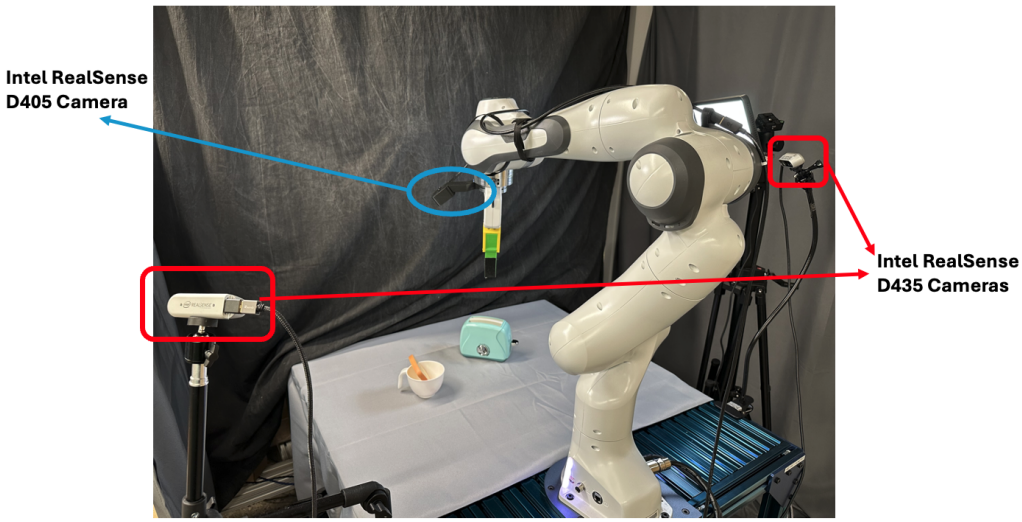

Figure 7: Our real robot platform setup.

Tasks	Language Goal	Demos	Skills	Mean Ep. Length
`Fruits in Box`	Put all the fruits into the box.	77	Pick. Place.	366
`Ball in Drawer`	Put the tennis ball inside the drawer.	80	Open. Grasp. Place. Close.	390
`Bread in Toaster`	Put the bread inside the toaster.	120	Grasp. Reorient. Insert.	251
`Place Coffee Cup`	Put the cup on the coffee machine.	120	Grasp. Reorient. Place.	270
`Egg in Carton`	Put the eggs inside the egg carton.	151	Grasp. Reorient. Place. Close.	319

Table 7: Real-world dataset statistics for training multi-task policies. Mean episode lengths are given in environment timesteps.

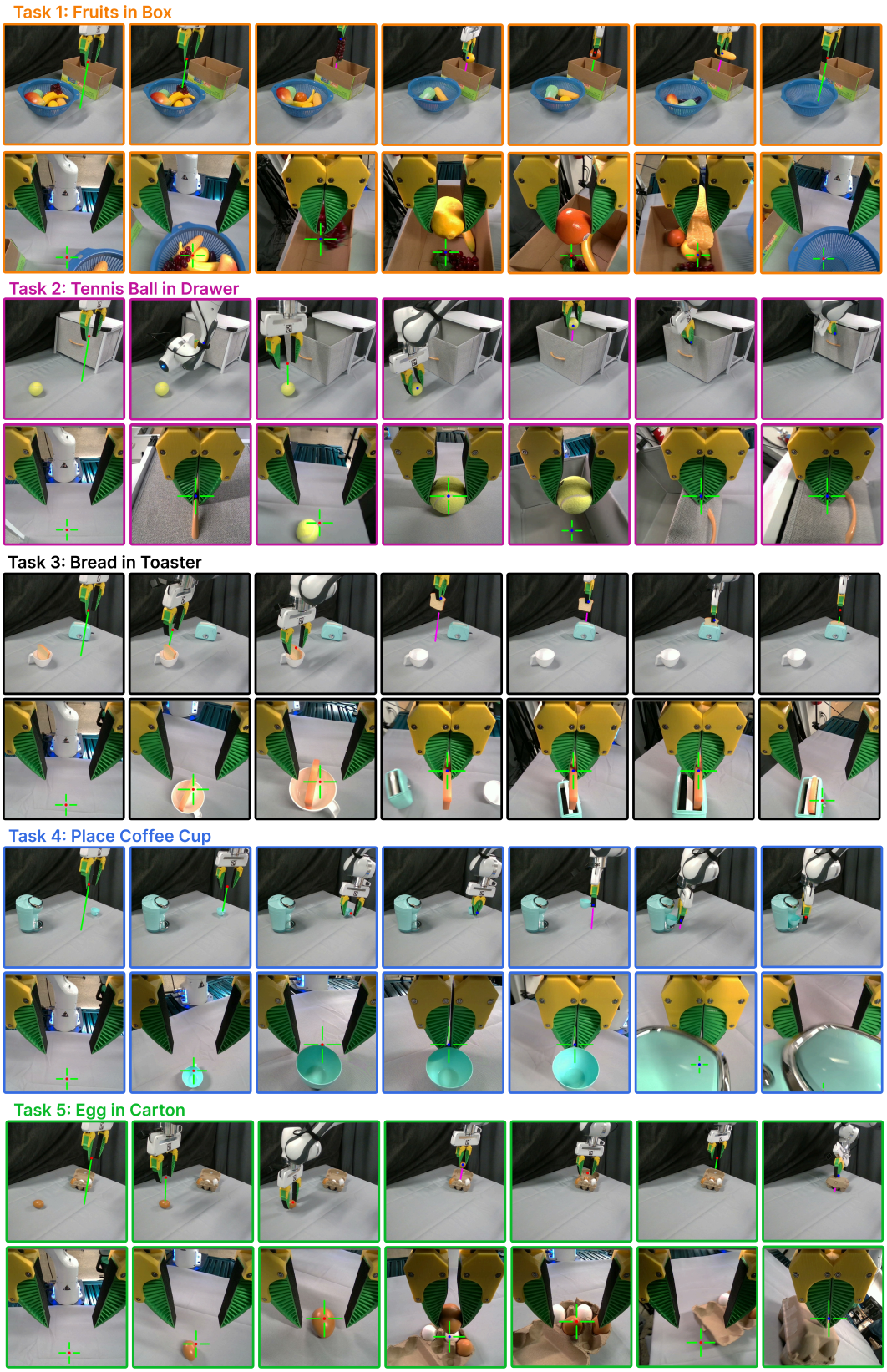

Figure 8: **Task Examples and Visualization of `AimBot`'s in Multi-View Observations.** We test our method on five challenging tasks: **`Fruits in Box`**, **`Tennis Ball in drawer`**, **`Bread in Toaster`**, **`Place Coffee Cup`** and **`Egg in Carton`**. Here we show how our **`AimBot`** augment the shooting line and scope reticle in the left shoulder and wrist camera views, respectively.

A.4 Visualization of attention maps

Figure 9 shows more examples of attention visualizations from the OpenVLA-OFT model, trained with and without our proposed **`AimBot`**. The visualized attention maps correspond to the 1st layer 11th head in the LLM backbone, summing up all attention probabilities from the action prediction tokens to the image patches. To generate the heatmaps, we project the patch-wise attention scores onto the original image using an upsampling method as in GradCAM[44]. We observe that incorporating **`AimBot`** leads to more focused attention on task-relevant objects, thereby enhancing the model's spatial understanding and improving manipulation performance.

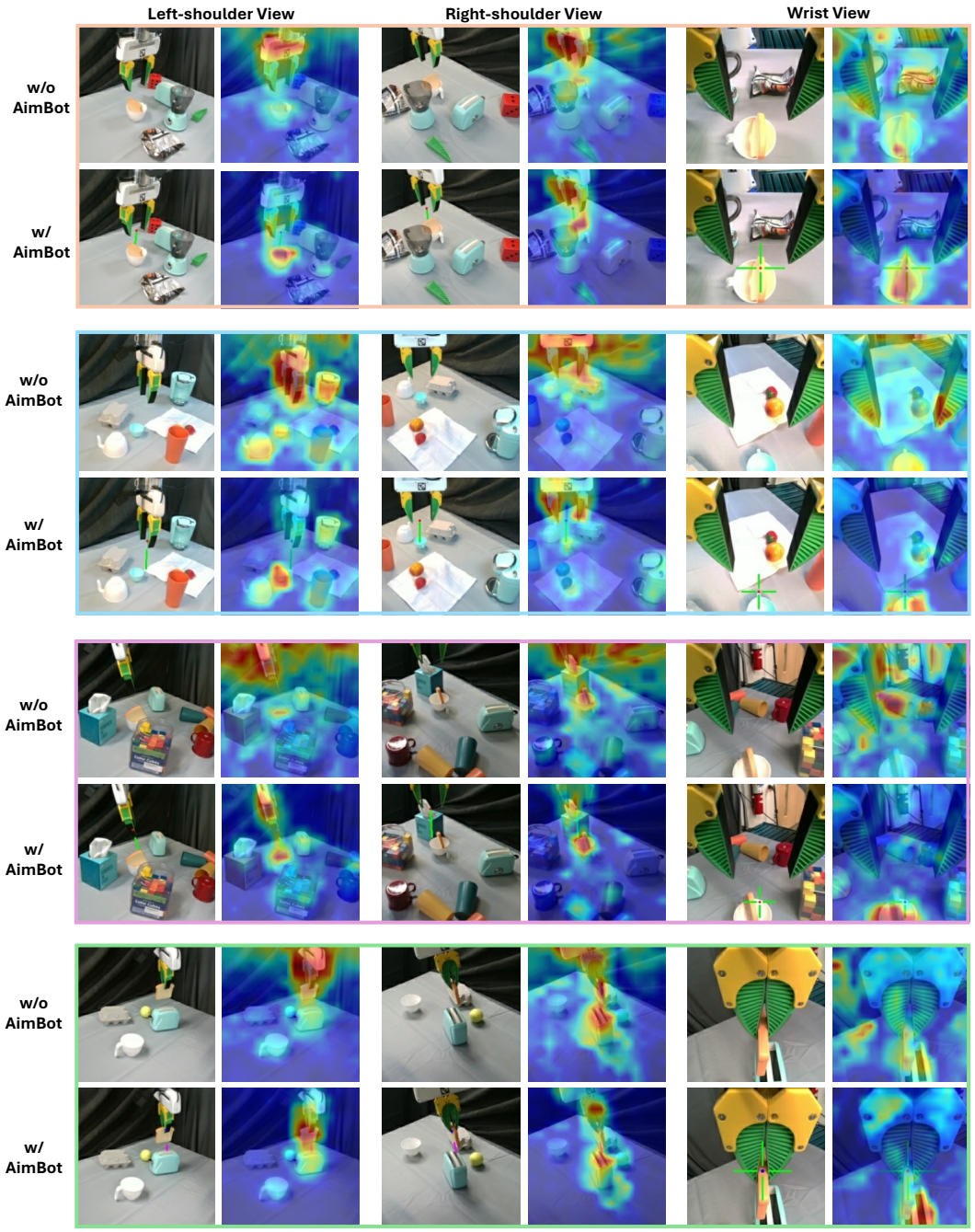

Figure 9: Additional attention heatmap examples comparing w/ and w/o **`AimBot`**.

A.5 Misalignment Failure Examples

We categorize all failure trials into different types: (1) grasping position misalignment, (2) grasping orientation misalignment, (3) placing position misalignment, (4) placing orientation misalignment, and (5) other non-misalignment failures (e.g., getting stuck, failed non-prehensile skills, or performing actions in the wrong task order). If a failure trial involves multiple misalignment cases, we attribute the failure to the last observed misalignment. Examples of such misalignments that led to failures in our real-world experiments are illustrated in Figure 10.

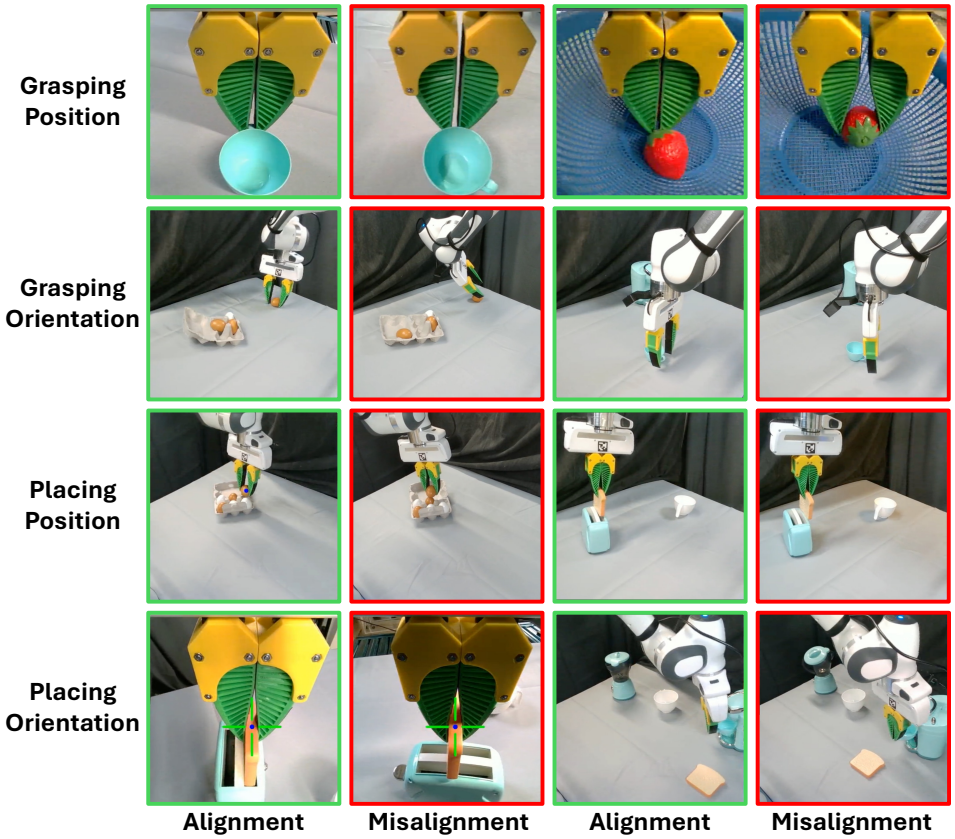

Figure 10: Examples of four misalignment types in real-world robot trials: grasping position, grasping orientation, placing position, and placing orientation. Each row shows representative wrist-view or shoulder-view images. For each misalignment type, we include aligned (green borders) and misaligned (red borders) examples to contrast correct behavior with failure cases.

A.6 Comparison of Proprioceptive State Representation

We compare different ablation of π_0 variants training with or without **AimBot** and using or not using proprioceptive state vector encoding. The results on the LIBERO benchmark are given in Table 8.

Model	LIBERO Spatial	LIBERO Object	LIBERO Goal	LIBERO Long	Average Success Rate
π_0 + **AimBot**	**96.9**	98.4	**97.2**	**91.0**	**95.9**
π_0 + **AimBot** − proprio.	96.6	96.8	94.8	88.0	94.1
π_0	96.8	**98.8**	95.8	85.2	94.2
π_0 − proprio.	96.6	96.4	94.8	83.2	92.8
π_0 + **AimBot** (random)	93.4	92.0	89.8	77.4	88.1

Table 8: Comparison of different proprioceptive state representations.

A.7 Out-of-distribution Scenes

Out-of-distribution (OOD) examples are illustrated in Figure 11. All OOD evaluation rollouts can be found at https://aimbot-reticle.github.io/.

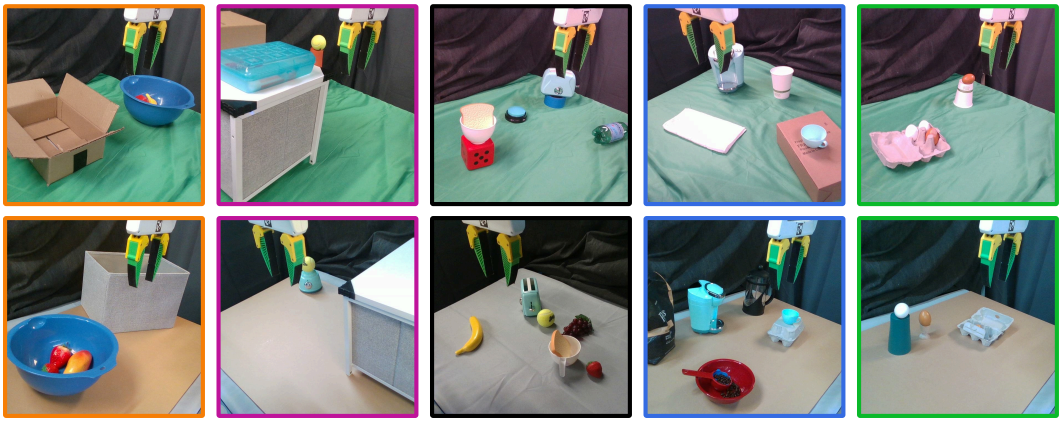

Figure 11: Sample out-of-distribution evaluation scenes to test generalization of policies to different backgrounds, varying lighting conditions (e.g. flashing lights, cool/warm lights), unseen distractors with real-time human perturbations, and varying camera poses, where **AimBot** bridges the distribution gap by grounding to useful depth information instead of plain visual features.