Is OpenVLA Truly Robust? A Systematic Evaluation of Positional Robustness

Yiran Pang[1*], Yiheng Zhao[2,*†], Zhuopu Zhou[1], Tingkai Hu[3], Ranxin Hou[3],
[1]Florida Atlantic University, Boca Raton, USA
[2]Concordia University, Montreal, Canada
[3]Independent Researcher
†**Correspondence:** yiheng.zhao@mail.concordia.ca
*Equal contribution

Abstract

Pretrained language and vision-language models have become core components in building vision-language-action models (VLAs) due to their strong spatial reasoning capabilities. Evaluating the robustness of VLAs is crucial to ensuring their reliability in practical scenarios. Although prior work has focused on background and environment robustness, positional robustness remains underexplored. In this paper, we propose a comprehensive evaluation protocol to assess the positional robustness of VLAs and apply it to OpenVLA, an open-source, high-performing, and efficient model well suited for real-world deployment. We find that OpenVLA succeeds only when the target object is placed at one of the two positions encountered during training. Even in these cases, the success rate never exceeds 50% because it exhibits a memorized behavior that it randomly executes a grasping action toward one of the two fixed positions without relying on perception to localize the target object. This reveals that OpenVLA's positional robustness is extremely weak.

1 Introduction

In recent years, pretrained language (Guo et al., 2025; Yang et al., 2024; Achiam et al., 2023; Wang et al., 2024b, 2023) and vision-language (Bai et al., 2025; Hurst et al., 2024; Team et al., 2023) models have demonstrated strong spatial reasoning capabilities, which are crucial for robotics. Consequently, leveraging these models as core components to build vision-language-action models (VLAs) can avoid training robotic policies from scratch and has emerged as a promising paradigm (Brohan et al., 2023; Team et al., 2024; Kim et al., 2024b). However, in real-world applications, robots frequently operate in complex and dynamic environments, which can significantly impact the performance of these VLAs in pick-and-place objects (Wang et al., 2024a; Guruprasad et al., 2024; Zhong et al., 2025). Therefore, evaluating their robustness is essential to ensuring their reliability in practical scenarios.

Although existing work has begun to evaluate the robustness of VLAs, these efforts have primarily focused on factors such as background variations and environmental changes (Brohan et al., 2022; Kim et al., 2024b; Zhang et al., 2024). However, there remains a lack of systematic analysis of positional robustness—that is, whether VLAs can maintain reliable pick-and-place objects performance when the position of the object changes. This aspect is particularly important for two reasons. First, in real-world scenarios, the positions of objects are rarely fixed and often subject to change (Qu et al., 2025; Li et al., 2025). Second, the ability of VLAs to adapt to such spatial shifts reflects their spatial reasoning capacity, which is crucial for executing reliable policies in dynamic environments.

In this paper, we propose a comprehensive evaluation protocol to assess the positional robustness of VLAs and apply it to OpenVLA (Kim et al., 2024a), an open-source, high-performing, and efficient model well suited for real-world deployment, in object pick-and-place tasks. Specifically, our protocol comprises three components. First, we assess the positional robustness of OpenVLA under the default configuration used in its original training and evaluation within the same simulated environment, which consists of the designated grasp-and-place target object along with multiple distractor objects. All objects are placed at canonical positions defined by the default spatial layout. By separately swapping the target object with each distractor object, we evaluate how changes in its position affect performance. Second, we remove all distractor objects and individually place the target object at each of the canonical positions defined by the default layout to evaluate how changes in its position affect performance without external interference. Finally, with all distractor objects removed, we place the target object at positions

Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 1–6

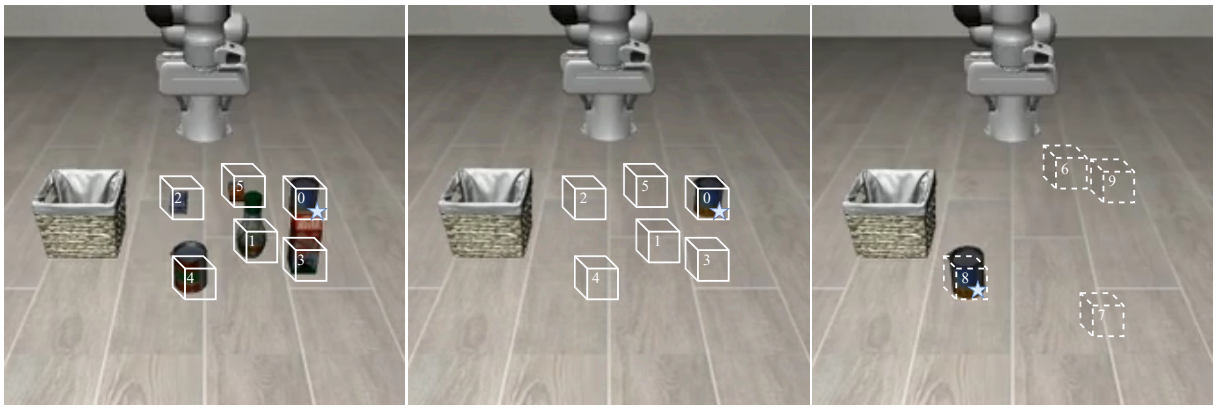


Figure 1: Positional robustness evaluation settings. **Left (Default configuration):** The target object is placed at canonical positions with distractors present, following the default setup used in OpenVLA's training and evaluation. **Middle (Only target object in canonical positions configuration):** Only the target object is retained and placed at each canonical position, without any distractors. **Right (Only target object in extend positions configuration):** The target object is placed at positions outside the default canonical layout, with all distractors removed. The four dashed white boxes indicate positions outside the default spatial layout. The solid white boxes in the left and middle figures correspond to the default canonical positions defined by the LIBERO manipulation suite, while the four dashed boxes in the right figure are additional positions selected as part of our evaluation. In each figure, the target object is marked with a blue star.

beyond those defined in the default layout to further assess OpenVLA's robustness to non-default spatial configurations.

We find that OpenVLA succeeds only when the target object is placed at one of the two positions encountered during training across all evaluation settings. Even in these cases, the success rate never exceeds 50% because it exhibits a memorized behavior that it randomly executes a grasping action toward one of the two fixed positions without relying on perception to localize the target object. This reveals that OpenVLA's positional robustness is extremely weak.

2 Evaluation Settings

Environment We conduct all our positional robustness evaluations using the LIBERO manipulation suite (Gupta et al., 2023), as OpenVLA was both trained and evaluated on this simulated environment, covering the full set of object pick-and-place tasks defined within the platform.

Protocol Our protocol comprises three components. First, we assess the positional robustness of OpenVLA under the default configuration defined by the LIBERO manipulation suite, referred to as default configuration, which was also used during its original training and evaluation. As shown on the left of Figure 1, each object pick-and-place task in this configuration contains six objects placed at canonical positions defined by the default spatial layout of the environment, one designated as the target object and the remaining five as distractors. We then swap it with each of the five distractor objects to evaluate how changes in the target's position affect performance. Second, we remove all distractor objects and retain only the target object in the environment, as shown in the middle of Figure 1. We then individually place the target object at each canonical position defined by the default layout to evaluate how changes in its position affect performance without external interference. Finally, with all distractor objects removed, we also place the target object at positions beyond those defined in the default layout, as shown on the right of Figure 1, to further assess OpenVLA's positional robustness to non-default spatial configurations.

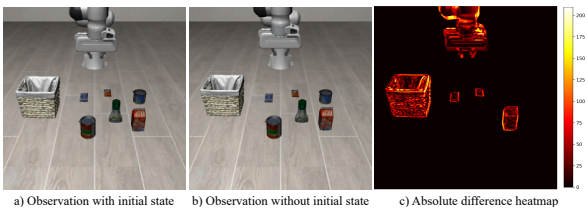


Figure 2: Effect of removing the `.init` file. a) Image rendered after loading the original initial state. b) Image after a cold reset. c) Absolute pixel-wise difference between a) and b) visualised as a heat-map. Brighter colors highlight regions with greater differences, indicating subtle pose drift of objects, the target bin, and the end-effector.

In our evaluations, we disable LIBERO's default `.init` snapshots, which lock the scene to six canonical coordinates and introduce a small but systematic pose drift. The difference between using and not using the `.init` snapshots is shown in Figure 2. Instead, all evaluations starts from a cold reset, ensuring unbiased layouts and allowing fair evaluation at the additional positions.

Metrics We execute each object pick-and-place task 10 times and compute the average success rate across trials as our evaluation metric. A task is considered successful if the target object is released inside the bin within the time limit. Additionally, we use the official weights released by OpenVLA in all our evaluations.

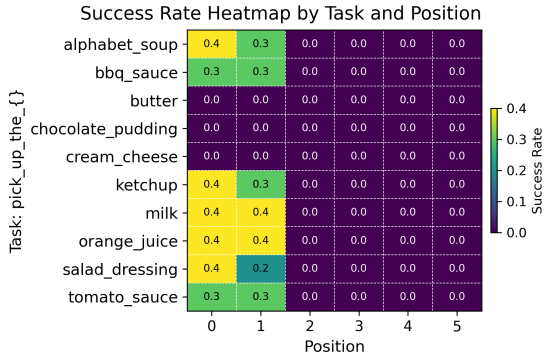


Figure 3: Success-rate heatmaps under default configuration. The y-axis represents task names, and the x-axis indicates the placement positions of the target objects. Heatmap values denote the success rate over 10 repeated trials. Successful attempts only occur when the target item is placed in position 0 or 1.

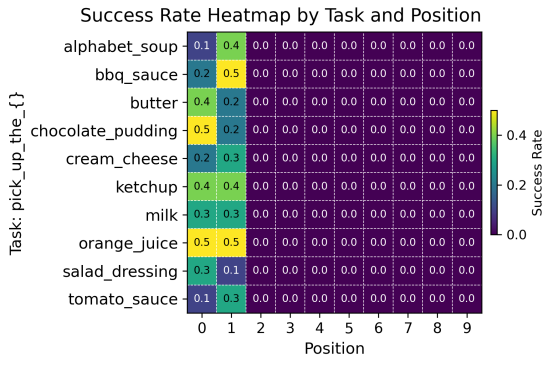


Figure 4: Success-rate heatmaps for scenarios where only the target object is present. The evaluation covers six canonical and four new positions for the target object. Even with distractors removed, successful attempts occur exclusively when the target object is placed at position 0 or 1.

3 Experiments

All of our evaluation results are shown in Figures 3 and 4. The results reveal that OpenVLA exhibits extremely weak positional robustness.

Default configuration As shown in Figure 3, every task exhibits non-zero success rates only at regions 0 and 1—the two positions encountered during pre-training. Even at these positions, the success rate never exceeds 0.5. The complete absence of successes at regions 2–5 suggests that the controller memorizes two fixed grasp trajectories rather than learning to localize the object using visual input.

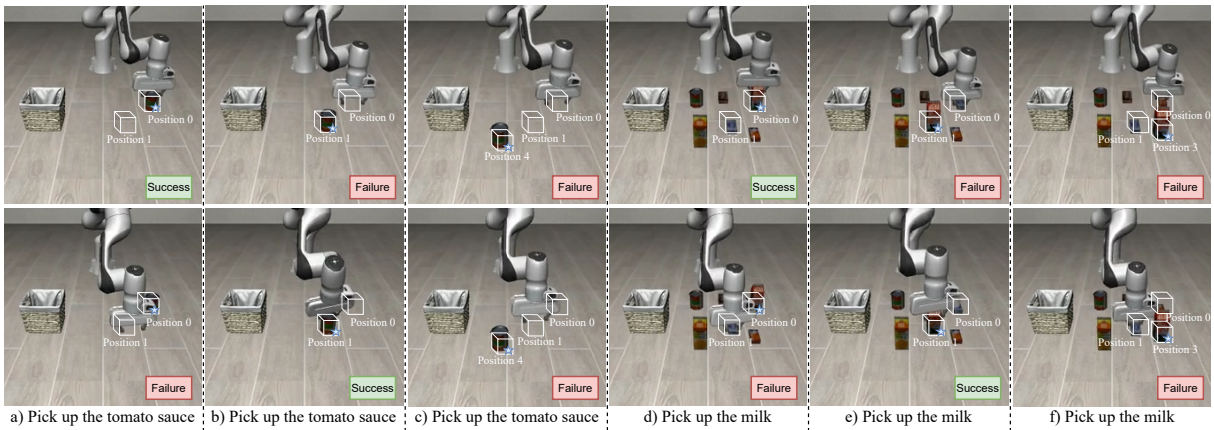
Only target object in canonical positions configuration As shown in Figures 3 and 4, the heatmaps from the default configuration and the target-only setting are nearly identical. This indicates that the choice of grasp site is not influenced by the presence or absence of distractor objects.

Only target object in extended positions configuration As shown in Figure 4, when the target object is placed at the four newly introduced positions, the success rate drops to zero. This confirms that OpenVLA's policy lacks generalization to spatial configurations beyond its narrow training distribution.

4 Analysis

Figure 5 offers a microscopic view of the task named "pick up the tomato sauce" under the object only condition and "pick up the milk" under default condition. Each column fixes the target at a distinct region, while each row corresponds to the arm's first decisive motion after a cold reset. A striking regularity emerges: immediately after the scene is rendered the policy drives the end-effector to either canonical position 0 or 1, with no intermediate search or corrective behaviour. The branch chosen varies across episodes and appears random, yet once committed the controller follows a hard-coded, open-loop trajectory that was seemingly over-optimised for the training distribution.

Because the grasp site is picked before any visual feedback is used to verify the target's whereabouts, success hinges purely on coincidence. If the object happens to sit at the visited location—as in the green-framed panels of the first and second columns—the episode completes successfully; otherwise the gripper closes on empty space, producing the red-framed failures shown in all other panels. Crucially, this outcome is independent of the



a) Pick up the tomato sauce (at position 0) | b) Pick up the tomato sauce (at position 1) | c) Pick up the tomato sauce (at position 4) | d) Pick up the milk (at position 0) | e) Pick up the milk (at position 1) | f) Pick up the milk (at position 3)

Figure 5: Behaviour of the learned policy on the “pick-up-the-tomato-sauce" task (Only target object in canonical positions configuration) and the “pick-up-the-milk" task (Default configuration). Columns (a)–(c) place the tomato sauce target object at positions 0, 1, and 4, respectively. Columns (d)–(f) place the milk target object at positions 0, 1, and 3, respectively. For both tasks, the first row shows episodes in which the arm moves to canonical position 0; the second row shows episodes in which it moves to position 1. After every cold reset, the policy chooses one of the two training positions (0 or 1) and ignores the actual scene—a choice unaffected by the presence or absence of distractors, consistently demonstrating its reliance on memorized trajectories across different tasks.

presence or absence of distractors: the arm never consults the broader scene once its initial choice is made.

This behaviour explains both the 50% ceiling observed at regions 0 and 1 and the zeros elsewhere. By randomly oscillating between two memorised coordinates, the controller self-imposes an upper bound of one successful grasp every two attempts in the best case, perfectly aligning with the peak values in Figures 3 and 4. Equally importantly, refusing to explore any novel location clarifies why the four additional regions introduced in the extended object position setting record no successes. Taken together, these episode-level observations confirm that the headline performance of OpenVLA derives from positional priors baked into the training curriculum and a lack of positional robustness.

5 Conclusion

In this work, we present a comprehensive evaluation protocol designed to assess the positional robustness of VLAs, an essential yet underexplored aspect of robustness in robots, and apply it to evaluate the robustness of OpenVLA—an open-source, high-performing, and efficient model well suited for real-world deployment. Our evaluation reveals that OpenVLA’s positional robustness is extremely weak.

Limitations

Our study evaluates only the positional robustness of OpenVLA using our proposed protocol. Furthermore, all experiments are conducted in a simulated environment, which may not fully capture the challenges of real-world deployment. In future work, we plan to apply our protocol to a broader range of VLA models and assess their robustness in real-world.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, and 1 others. 2023. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, and 1 others. 2022. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Abhishek Gupta, Jacky Liang, Rahul Shah, Frederik Ebert, and 1 others. 2023. Libero: Benchmarking knowledge transfer in language-instructed robot manipulation. In *Conference on Robot Learning (CoRL)*.

Pranav Guruprasad, Harshvardhan Sikka, Jaewoo Song, Yangyue Wang, and Paul Pu Liang. 2024. Benchmarking vision, language, & action models on robotic learning tasks. *arXiv preprint arXiv:2411.05821*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. 2024a. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, and 1 others. 2024b. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*.

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

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, and 1 others. 2025. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, and 1 others. 2024. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*.

Zhijie Wang, Zhehua Zhou, Jiayang Song, Yuheng Huang, Zhan Shu, and Lei Ma. 2024a. Towards testing and evaluating vision-language-action models for robotic manipulation: An empirical study. *arXiv preprint arXiv:2409.12894*.

Zhiqiang Wang, Yiran Pang, and Yanbin Lin. 2023. Large language models are zero-shot text classifiers. *arXiv preprint arXiv:2312.01044*.

Zhiqiang Wang, Yiran Pang, Yanbin Lin, and Xingquan Zhu. 2024b. Adaptable and reliable text classification using large language models. In *2024 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 67–74. IEEE.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, and 1 others. 2024. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. *arXiv preprint arXiv:2412.18194*.

Yifan Zhong, Xuchuan Huang, Ruochong Li, Ceyao Zhang, Yitao Liang, Yaodong Yang, and Yuanpei Chen. 2025. Dexgraspvla: A vision-language-action framework towards general dexterous grasping. *arXiv preprint arXiv:2502.20900*.

A Experimental Parameters

We used a consistent set of parameters for all evaluation experiments to ensure the fairness and reproducibility of our results. The detailed hyperparameters and environmental settings are listed in Table 1.

Table 1: Evaluation Hyperparameters.

Parameter	Value
Model Family	openvla
Pretrained Checkpoint	openvla/openvla-7b -finetuned-libero-object
Task Suite	libero_object
Trials per Task	10
Center Crop	True
Image Resolution	256×256 pixels
Random Seed	7
Wait Steps for Stabilization	10
Max Episode Steps	280

B Positional Coordinates

Our positional robustness evaluation used two sets of positions: Canonical Positions and Extended Positions. The canonical positions are the six default object placement regions defined in the LIBERO task suite. The extended positions are four additional regions we introduced to further test the model's generalization capabilities. All coordinates are defined on the floor of the simulation environment. The specific coordinate ranges are detailed in Table 2.

Table 2: Positional Coordinates for Target Object Placement.

Position ID	Type	Coordinate Range (xmin, ymin, xmax, ymax)
0	Canonical	(0.025, -0.125, 0.075, -0.075)
1	Canonical	(-0.145, -0.265, -0.095, -0.215)
2	Canonical	(-0.175, 0.035, -0.125, 0.085)
3	Canonical	(0.075, -0.225, 0.125, -0.175)
4	Canonical	(0.125, 0.005, 0.175, 0.055)
5	Canonical	(-0.225, -0.105, -0.175, -0.055)
6	Extended	(-0.24, -0.155, -0.19, -0.105)
7	Extended	(0.18, -0.17, 0.23, -0.12)
8	Extended	(0.14, 0.065, 0.19, 0.115)
9	Extended	(-0.16, -0.195, -0.11, -0.145)

C Task Configuration Modification

We precisely controlled the initial position of objects in each evaluation scenario by modifying the Behavior, Description, and Domain Language (BDDL) files of the LIBERO environment.

- **Default Configuration Evaluation**: In this evaluation, we kept all objects (one target and five distractors). We changed the target's position by swapping its initial region with that of a distractor object in the '(:init)' block of the BDDL file. For example, to place the target object at Position 1, we changed its '(On ...)' predicate from 'floor_target_object_region' to 'floor_other_object_region_0'.
- **Target-Only Evaluation**: For this evaluation, we first removed all distractor objects from the '(:objects)' and '(:init)' blocks of the BDDL file. We then placed the single target object at each of the 10 locations listed in Table 2 (6 canonical and 4 extended) and evaluated them independently.
- **Simulation Reset**: To avoid any positional bias from pre-saved initial states (.init files), we did not use the 'env.set_init_state()' function in our experiments. Each trial began from a "cold reset" of the simulation by calling 'env.reset()'. This step ensures an unbiased object layout and fair evaluation, especially for the extended positions.