# SAM3D-Guided Object-Centric Representation Alignment for Vision-Language-Action Models

Zonghe Liu[*,1], ShanYuan Jie[*,2], Xiaoquan Sun[‡,3], Chen Cao[‡,1],
Zetian Xu[‡,1], Liu Zongsheng[‡,4], Jiayu Chen[†,1,5]

[1]University of Hong Kong [2]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
[3]Huazhong University of Science and Technology [4]Beijing University of Aeronautics and Astronautics
[5]Infiforce

[*]Equal Contribution [‡]Equal Second Contribution [†]Corresponding Author

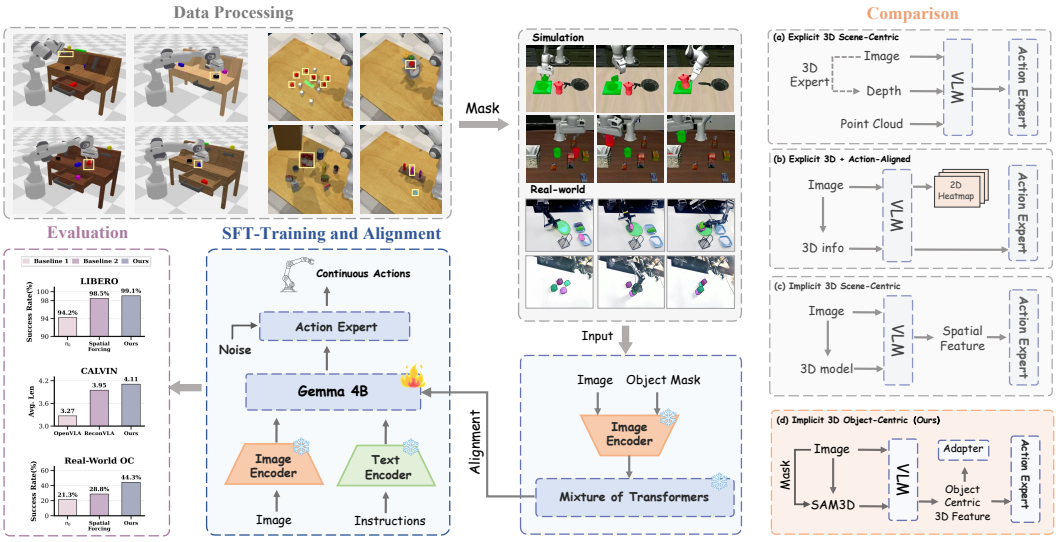


Figure 1: **Overview of SAM3D-VLA**. We propose an object-centric 3D alignment framework that uses SAM3D as a frozen teacher during training. High-level instructions are decomposed into subtasks, and task-relevant object masks are used for SAM3D feature extraction. The extracted 3D features are spatially resampled and dimensionally projected to align with intermediate VLA representations, while inference follows the original pipeline using RGB and language instructions.

**Abstract:** Vision-Language-Action (VLA) models have shown strong potential for general robot manipulation, but most existing models rely on 2D visual-language backbones and lack fine-grained 3D understanding of target objects, especially under occlusion, pose variation, scale changes, and precise spatial interaction. We propose an object-centric 3D representation alignment framework built upon $\pi_0$, using SAM3D as a frozen 3D teacher to provide target-object 3D priors during training. Specifically, we localize task-relevant objects with object recognition models, generate corresponding object masks, and use SAM3D to extract dense object-level 3D representations, which are aligned with intermediate visual features of $\pi_0$. This enables the policy to internalize target-object 3D information while preserving the original RGB-language-to-action inference pipeline without requiring depth, point clouds, masks, SAM3D, or additional 3D modules at test time. Simulation experiments show consistent improvements, achieving $99.1\%$ on LIBERO and $4.11$ average length on CALVIN. Real-world experiments

10th Conference on Robot Learning (CoRL 2026), Austin TX, USA.

further demonstrate that our method is particularly effective in long-horizon manipulation scenarios where the robot must focus on different target objects across multiple subtasks.

**Keywords:** Vision-Language-Action Models, Object-centric 3D Priors, Representation Alignment

## 1 Introduction

Vision-Language-Action (VLA) models have become an important direction for general robot manipulation, as they adapt pretrained vision-language models to predict robot actions [1, 2, 3]. With large-scale visual-language pretraining, these models can understand language instructions and handle a wide range of manipulation tasks. However, most existing VLAs still rely mainly on 2D RGB images and learn from action prediction alone. As a result, they may not capture the 3D properties of the target object, such as its shape, pose, scale, and spatial layout. This becomes a clear limitation in tasks with occlusion, pose changes, clutter, or precise object placement.

Recent works have attempted to enhance VLA models with spatial or 3D information. Some methods introduce explicit 3D observations, such as depth maps, point clouds, or RGB-D inputs, into the policy [4, 5, 6, 7]. Others inject estimated 3D positions, spatial action grids, or representation-level spatial supervision into VLA models [8, 9]. These methods demonstrate the importance of 3D structure for manipulation, but they often require additional 3D inputs, modify the original input-output interface, or focus mainly on global scene-level spatial representations. Meanwhile, visual grounding methods improve target-region attention through cropped inputs, bounding-box prediction, or gaze-region reconstruction [10, 11, 12, 13], but their supervision remains primarily in the 2D image space and does not explicitly provide target-object 3D shape or layout priors.

This raises a key question: can we inject object-centric 3D knowledge into an RGB-based VLA policy while preserving its original inference pipeline? To this end, we propose a SAM3D-guided object-centric 3D representation alignment framework built upon $\pi_0$ [3]. During training, we automatically ground the task-relevant object with open-vocabulary detection [14, 15] and SAM2 [16] segmentation, and use a frozen SAM3D [17] teacher to extract dense object-centric 3D features from the masked target. These teacher features are spatially resampled to match the visual token grid of $\pi_0$, and the intermediate VLA features are projected into the SAM3D feature space for masked normalized representation alignment. For long-horizon tasks, we further decompose high-level instructions into subtasks and associate each subtask with its corresponding target object, providing stage-specific 3D supervision. Importantly, all teacher-side modules are used only during training.

During inference, our policy follows the original $\pi_0$ RGB-language-to-action pipeline and requires no depth maps, point clouds, object masks, SAM3D, or additional 3D modules. Experiments on LIBERO and CALVIN show that our method consistently improves over strong VLA baselines, achieving **99.1%** on LIBERO and average length of **4.11** on CALVIN. We further verify the mechanism with a frozen-representation probing experiment, showing that SAM3D object-centric 3D priors can be more accurately predicted from the learned VLA features after training.

Our contributions are summarized as follows:

- We identify Object-Centric 3D understanding as a key bottleneck of RGB-based VLA policies, complementing prior works that mainly focus on global spatial awareness, action-space spatialization, or 2D visual grounding.
- We propose **SAM3D-VLA** that distills shape and layout priors of target object into the intermediate visual features of $\pi_0$ [3], while preserving the original inference pipeline without depth maps, point clouds, masks, or additional 3D modules at test time.
- We demonstrate consistent improvements on simulation and real-world tasks, and further verify the mechanism through a frozen-representation probing experiment, showing that object-centric 3D priors can be more accurately recovered from the learned VLA features after training.

## 2 Related Works

**Vision-Language-Action Models**. Vision-Language-Action (VLA) models have become a promising paradigm for general robot manipulation by adapting pretrained vision-language backbones to robot control [1, 2, 3]. These models can follow language instructions and perform diverse manipulation tasks, but most of them rely mainly on 2D RGB observations, limiting their ability to capture object-level 3D structure. Visual grounding methods improve target attention through cropped regions, bounding boxes, or gaze reconstruction [10, 11, 18, 12, 13]. However, their supervision remains mostly in the 2D image space and does not explicitly provide target-object 3D shape or layout priors.

**Spatial and 3D-aware VLA Models**. Another line of work introduces spatial or 3D information into VLA models to improve manipulation performance. Some methods incorporate explicit 3D observations such as depth maps, point clouds, or RGB-D inputs [7, 4, 19, 5]. For example, BridgeVLA projects point clouds into orthographic 2D views and predicts heatmaps for action localization, but it still relies on 3D observations and changes the policy input-output formulation. Other methods inject spatial information through architectural designs, action representations, or representation-level supervision. SpatialVLA [8] uses Ego3D Position Encoding and Adaptive Action Grids, while Spatial Forcing [9] aligns intermediate VLA features with representations from pretrained 3D foundation models. These works show the value of spatial structure for manipulation, but often focus on explicit 3D inputs, action-space redesign, or scene-level spatial supervision. In contrast, our method uses subtask-aware object-centric 3D priors only during training, while preserving the original RGB-language-to-action inference pipeline of $\pi_0$.

## 3 Method

### 3.1 SAM3D-guided VLA Training Framework

**Vision-Language-Action Models.** SAM3D-VLA is built upon the $\pi_0$ [3] architecture, which combines a pretrained vision-language backbone with a continuous action expert. Following $\pi_0$, the VLM backbone is composed of a SigLIP [20] vision encoder and a Gemma [21] language model. Formally, at robot timestep $t$, we model the conditional action distribution $p_\theta(\mathbf{A}_t \mid \mathbf{o}_t)$, where $\mathbf{A}_t = [\mathbf{a}_t, \mathbf{a}_{t+1}, \ldots, \mathbf{a}_{t+H-1}]$ denotes an action chunk of horizon $H$, and $\mathbf{o}_t$ is the robot observation. The observation consists of multi-view RGB images, a language instruction, and the robot proprioceptive state: $\mathbf{o}_t = [\mathbf{I}_t^1, \ldots, \mathbf{I}_t^n, \ell, \mathbf{q}_t]$, where $\mathbf{I}_t^i$ is the $i$-th camera image, $\ell$ is the language command, and $\mathbf{q}_t$ denotes the robot state such as joint angles or gripper state.

The RGB images are encoded by the SigLIP vision encoder into visual tokens, while the language command is embedded into text tokens in the Gemma token space. The robot state is projected into the same embedding dimension through a linear projection layer. These tokens are then processed by the Gemma-based multimodal Transformer:

$$\mathbf{H}_t = f_{\mathrm{VLM}}(\mathbf{I}_t^1, \ldots, \mathbf{I}_t^n, \ell, \mathbf{q}_t). \tag{1}$$

where $\mathbf{H}_t = \{\mathbf{h}_t^{(1)}, \ldots, \mathbf{h}_t^{(L)}\}$ denotes hidden representations from different Transformer layers. In the standard $\pi_0$ pipeline, these features condition the action expert for continuous action generation.

For action prediction, each action in the chunk is represented as a continuous action token and processed by the action expert. We train the action expert using a conditional flow matching objective. Given a ground-truth action chunk $\mathbf{A}_t$, we sample Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and a flow matching timestep $\tau \in [0, 1]$. The noisy action chunk is constructed as $\mathbf{A}_t^\tau = \tau \mathbf{A}_t + (1-\tau)\boldsymbol{\epsilon}$, the action expert predicts a velocity field conditioned on the VLM features: $\hat{\mathbf{v}}_\theta = \mathbf{v}_\theta(\mathbf{A}_t^\tau, \tau, \mathbf{o}_t)$, and is optimized to match the target denoising direction:

$$\mathcal{L}_{\mathrm{action}} = E_{\tau, \boldsymbol{\epsilon}} \left[ \left\| \mathbf{v}_\theta(\mathbf{A}_t^\tau, \tau, \mathbf{o}_t) - (\mathbf{A}_t - \boldsymbol{\epsilon}) \right\|_2^2 \right]. \tag{2}$$

During inference, actions are generated by starting from Gaussian noise and integrating the learned velocity field from $\tau = 0$ to $\tau = 1$, producing the final action chunk for robot execution.

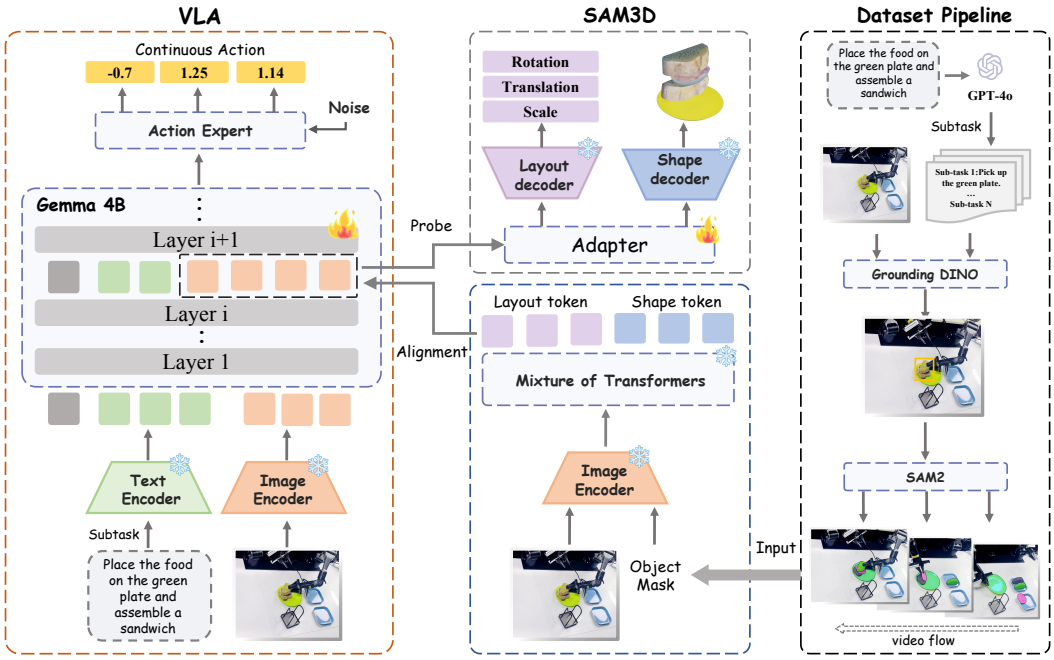


Figure 2: **The framework of SAM3D-VLA**. Task-relevant object masks are generated from subtask instructions and fed to a frozen SAM3D teacher to provide object-centric 3D supervision for intermediate VLA features. We further use a frozen-representation probing adapter to verify whether SAM3D priors are encoded in the learned VLA representations.

In our framework, we preserve the original $\pi_0$ formulation, where the policy maps RGB observations and language instructions to continuous actions. The key difference is that, during training, we additionally extract the intermediate visual features $\mathbf{h}_t^{(m)}$ from a selected Gemma layer and project them into the SAM3D representation space. These projected VLA features are supervised by object-centric 3D shape and layout priors from SAM3D. This design injects 3D object knowledge into the VLA representation while keeping the original $\pi_0$ inference pipeline unchanged.

**SAM3D-guided Object-centric Feature Alignment.** We use SAM3D as a frozen 3D teacher to provide object-centric geometric supervision for the intermediate visual features of the $\pi_0$ backbone. Since SAM3D operates on single images, we flatten a batch of multi-view observations $\mathbf{I}_t \in R^{B\times V\times 3\times H\times W}$ into $\tilde{\mathbf{I}}_t \in R^{(BV)\times 3\times H\times W}$, where $B$ is the batch size, $V$ is the number of camera views, and $H, W$ denote the image resolution. We flatten the corresponding subtask-specific object masks in the same way. The image-mask pairs are fed into the frozen SAM3D teacher to extract dense object-centric 3D features: $\mathbf{T}_t = f_{\text{SAM3D}}(\tilde{\mathbf{I}}_t, \tilde{\mathbf{M}}_t)$, $\mathbf{T}_t \in R^{(BV)\times L_T\times D_T}$, where $L_T$ is the number of SAM3D teacher tokens and $D_T$ is the teacher feature dimension. These features are extracted from the last transformer block of SAM3D and contain object-level geometric priors such as shape, surface structure, and spatial layout.

Because SAM3D and $\pi_0$ have different token resolutions, we reshape the teacher sequence into a 2D feature grid, $\mathbf{T}_t \to \mathbf{T}_t^{2D} \in R^{(BV)\times D_T\times H_T\times W_T}$, remove the global token if it exists, and resize the grid to the student visual-token resolution by bilinear interpolation: $\bar{\mathbf{T}}_t^{2D} = \text{Interp}(\mathbf{T}_t^{2D}, H_S, W_S)$, $H_S W_S = L_S$, where $L_S$ is the number of student visual tokens per camera view. The resized features are flattened and reassembled across views as $\bar{\mathbf{T}}_t \in R^{B\times(VL_S)\times D_T}$, making the teacher tokens spatially aligned with the corresponding $\pi_0$ visual tokens. On the student side, we extract intermediate visual features from a selected Gemma layer: $\mathbf{S}_t = \mathbf{h}_t^{(m)} \in R^{B\times(VL_S)\times D_S}$, where $D_S$ is the student feature dimension. Since the feature dimensions differ, we project the student features into the SAM3D feature space: $\hat{\mathbf{T}}_t = P_\phi(\mathbf{S}_t)$, $\hat{\mathbf{T}}_t \in R^{B\times(VL_S)\times D_T}$.

To focus supervision on task-relevant regions, we construct a token-level mask $\mathbf{m}_t \in \{0,1\}^{B\times(VL_S)}$ from the subtask-specific object masks and apply alignment only to target-object tokens. We further denote by $\mathbf{M}_t \in \{0,1\}^{B\times(VL_S)\times D_T}$ the mask expanded along the feature dimension. Both projected student features and SAM3D teacher features are $\ell_2$-normalized, and the object-centric alignment loss is defined as a masked normalized MSE:

$$\mathcal{L}_{\text{align}} = \text{MSE}\left(\text{Norm}(\hat{\mathbf{T}}_t)[\mathbf{M}_t], \text{Norm}(\bar{\mathbf{T}}_t)[\mathbf{M}_t]\right). \tag{3}$$

This normalized objective encourages directional feature alignment and is robust to scale differences between SAM3D and $\pi_0$. The final training objective combines the standard $\pi_0$ flow-matching action loss with the SAM3D-guided alignment loss:

$$\mathcal{L} = \mathcal{L}_{\text{action}} + \alpha\mathcal{L}_{\text{align}}. \tag{4}$$

where $\alpha$ controls the strength of the 3D feature supervision. This objective encourages the VLA backbone to encode object-centric 3D priors while preserving its original action prediction ability.

**Frozen-representation Probing Adapter.** To verify whether SAM3D-guided alignment makes object-centric 3D information more accessible in the learned VLA representation, we conduct a frozen-representation probing experiment. After policy training, we freeze the entire VLA backbone and train only a lightweight two-layer MLP probe $P_\omega$ on top of the same intermediate visual features $\mathbf{S}_t$ used for alignment. The probe predicts the spatially resampled SAM3D target $\bar{\mathbf{T}}_t$: $\tilde{\mathbf{T}}_t = P_\omega(\mathbf{S}_t)$, $\tilde{\mathbf{T}}_t \in R^{B\times(VL_S)\times D_T}$. Using the same expanded object mask $\mathbf{M}_t$, the probe is trained with

$$\mathcal{L}_{\text{probe}} = \text{MSE}\left(\text{Norm}(\tilde{\mathbf{T}}_t)[\mathbf{M}_t], \text{Norm}(\bar{\mathbf{T}}_t)[\mathbf{M}_t]\right). \tag{5}$$

Only $P_\omega$ is optimized, while the VLA model remains frozen. Therefore, better probing performance indicates that SAM3D-style object-centric 3D priors are more recoverable from the learned VLA features. We compare the original $\pi_0$ baseline with our SAM3D-aligned model to quantify this effect.

### 3.2 Subtask-aware Data Processing

For each training frame and camera view, we localize the subtask-relevant object using an object detection model and then use SAM2 to generate its binary mask. The resulting image-mask pairs are fed into the frozen SAM3D teacher to extract object-centric 3D features. Since SAM3D processes single images, multi-view observations are flattened along the camera dimension and each view is processed independently. These SAM3D features are used as training-time targets for representation alignment. During inference, this entire data processing pipeline is removed, and the policy follows the original $\pi_0$ pipeline using only RGB observations and language instructions.

## 4 Experiments

**Simulation Experiments on LIBERO.** We evaluate our method on LIBERO benchmarks [35]. LIBERO contains LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long. Each task suite contains 500 expert demonstrations across 10 tasks, designed to probe generalization to different spatial layouts, objects, goals, and long-horizon behaviors. SAM3D-VLA is compared with representative 2D VLA models [22, 36, 23, 2, 24, 25, 3, 37], recent 3D or spatially enhanced VLA methods [8, 6, 7] and Spatial Forcing [9]. All methods are evaluated using the standard success-rate metrics of each benchmark.

**Experiments results.** As shown in Table 1, SAM3D-VLA achieves the best average success rate on LIBERO, reaching **99.1%**. Compared with strong 2D VLA baselines such as $\pi_0$, UniVLA, and OpenVLA-OFT, our method obtains clear gains across object, goal, and long-horizon task categories. The improvement is especially evident on LIBERO-Long, where our method reaches **98.4%**. Since each LIBERO-Long task contains two sequential subtasks, the task-relevant object

Table 1: Comparisons with state-of-the-art methods on LIBERO benchmark. **Bold** denotes the best performance among all methods.

| Method | Spatial SR (%) | Object SR (%) | Goal SR (%) | Long SR (%) | Average SR (%) |
|---|---|---|---|---|---|
| 2D VLA | | | | | |
| Diffusion Policy [22] | 78.3 | 92.5 | 68.3 | 50.5 | 72.4 |
| Octo [23] | 78.9 | 85.7 | 84.6 | 51.1 | 75.1 |
| OpenVLA [2] | 84.7 | 88.4 | 79.2 | 53.7 | 76.5 |
| Dita [24] | 84.2 | 96.3 | 85.4 | 63.8 | 82.4 |
| CoT-VLA [25] | 87.5 | 91.6 | 87.6 | 69.0 | 83.9 |
| $\pi_0$ [3] | 96.8 | 98.8 | 95.8 | 85.2 | 94.2 |
| UniVLA [26] | 96.5 | 96.8 | 95.6 | 92.0 | 95.2 |
| OpenVLA-OFT [27] | 97.6 | 98.4 | 97.9 | 94.5 | 97.1 |
| Explicit 3D VLA | | | | | |
| SpatialVLA [8] | 88.2 | 89.9 | 78.6 | 55.5 | 78.1 |
| GeoVLA [6] | 98.4 | 99.0 | 96.6 | 96.6 | 97.7 |
| 3D-CAVLA [7] | 98.2 | **99.8** | 98.2 | 96.1 | 98.1 |
| Implicit 3D VLA | | | | | |
| Spatial Forcing [9] | **99.4** | 99.6 | 98.8 | 96.0 | 98.5 |
| SAM3D-VLA (Ours) | 99.2 | 99.7 | **99.1** | **98.4** | **99.1** |

Table 2: Comparisons with state-of-the-art methods on CALVIN benchmark. **Bold** denotes the best performance among all methods.

| Method | Splits | Success Rate (%) | | | | | Avg. Len |
|---|---|---|---|---|---|---|---|
| | | 1/5 | 2/5 | 3/5 | 4/5 | 5/5 | |
| Generative Methods | | | | | | | |
| UniPi [28] | ABC→D | 56.0 | 16.0 | 8.0 | 8.0 | 4.0 | 0.92 |
| SuSIE [29] | ABC→D | 87.0 | 69.0 | 49.0 | 38.0 | 26.0 | 2.69 |
| GR-1 [30] | ABC→D | 85.4 | 71.2 | 59.6 | 49.7 | 40.1 | 3.06 |
| VidMan [31] | ABC→D | 91.5 | 76.4 | 68.2 | 59.2 | 46.7 | 3.42 |
| CLOVER [32] | ABC→D | 96.0 | 83.5 | 70.8 | 57.5 | 45.4 | 3.53 |
| Large VLA Models | | | | | | | |
| VLAS [33] | ABC→D | 87.2 | 64.2 | 40.9 | 28.1 | 19.6 | 2.40 |
| RoboFlamingo [34] | ABC→D | 82.4 | 61.9 | 46.6 | 33.1 | 23.5 | 2.47 |
| OpenVLA [2] | ABC→D | 91.3 | 77.8 | 62.0 | 52.1 | 43.5 | 3.27 |
| UniVLA [26] | ABC→D | 95.5 | 85.8 | 75.4 | 66.9 | 56.5 | 3.80 |
| Object-centric Methods | | | | | | | |
| ReconVLA [10] | ABC→D | 95.6 | 87.6 | 76.9 | 69.3 | 64.1 | 3.95 |
| SAM3D-VLA (Ours) | ABC→D | **96.2** | **89.1** | **80.5** | **73.6** | **71.6** | **4.11** |

may change across stages. Our subtask-aware processing associates each subtask with its corresponding object mask, providing stage-specific SAM3D supervision and helping the policy focus on different target objects during different manipulation phases.

**Simulation Experiments on CALVIN.** CALVIN benchmark [38] consists of 34 tasks and 4 different environments (A, B, C and D). The CALVIN long-horizon challenge is a sequential task comprising five subtasks. We report the success rates for each subtask and the average completed length across all five tasks. The method is evaluated over 500 rollouts to ensure a fair comparison. The metrics of CALVIN are the success rates of each subtask and the average length of all sequential 5 subtasks.

**Experiments results.** On CALVIN, as shown in Table 2, SAM3D-VLA achieves the best performance with an average length of **4.11** and a 5-step success rate of **71.6%**. Compared with ReconVLA, UniVLA, OpenVLA, and other baselines, our method consistently improves success rates across task lengths, showing stronger long-horizon consistency. These gains indicate that

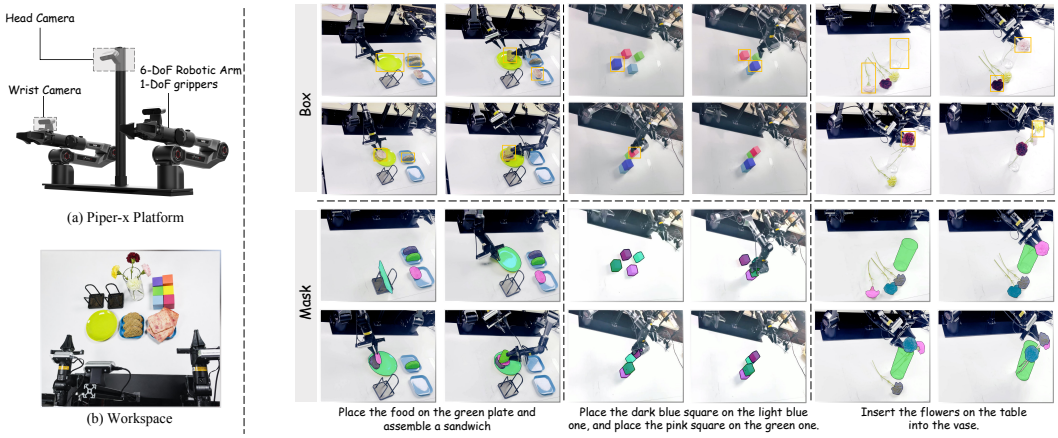


Figure 3: **Real-world experimental setup and Mask processing**. (a) Piper-x Platform. (b) Workspace. Dataset Processing: The target object is grounded and masked by open-source models.

SAM3D-guided object-centric alignment helps the policy encode target-object 3D priors for sequential manipulation, while preserving the original $\pi_0$ RGB-language-to-action inference pipeline without additional test-time inputs or modules.

**Real-world setup.** We further evaluate our method on the Piper-X robotic arm, a dual-arm mobile platform equipped with three omnidirectional wheels, two 6-DoF arms, two wrist cameras, and a head camera, as shown in Fig. 3. During deployment, the policy follows the original $\pi_0$ inference pipeline, taking only RGB observations and language instructions as input to predict continuous action chunks. We conduct experiments on three real-world manipulation scenarios: cooking, flower arrangement, and block stacking, covering object selection, grasping, insertion and placement. All methods are evaluated under the same robot setup and protocol, and training is conducted on 8 NVIDIA H100 GPUs.

**Real-world data processing.** For real-world training data, we use the same subtask-aware object-centric pipeline as in simulation. As showen in Fig. 4, each high-level instruction is decomposed into subtasks by an LLM, and the corresponding target objects are grounded by Grounding DINO [15] and YOLO [14], followed by SAM2 [16] segmentation. The resulting image-mask pairs are fed into the frozen SAM3D [17] teacher to extract object-centric 3D features for training-time alignment. During deployment, no subtask decomposition, detection, segmentation, or SAM3D module is required.

Table 3: Real-world results.

| **Task** | $\pi_0$ [3] | | **SAM3D-VLA** | |
|---|---|---|---|---|
| | ST | OC | ST | OC |
| **Basic Tasks** | | | | |
| Stack bowls | 90% | 55% | **92%** | **81%** |
| Hang cup | 75% | 15% | **81%** | **36%** |
| Open drawer | **68%** | 30% | 62% | **40%** |
| **Long Tasks** | | | | |
| Cook | 15% | 5% | **40%** | **26%** |
| Flower | 30% | 13% | **62%** | **45%** |
| Stack | 23% | 10% | **54%** | **38%** |
| **Average** | 50.2% | 21.3% | **65.2%** | **44.3%** |

**Real-world results.** Table 3 reports the real-world success rates on the AgileX PiperX robot. We evaluate each task under two settings: the standard setting (ST), where objects are placed in canonical initial states, and the occlusion setting (OC), where we introduce additional disturbances such as changed object positions, irrelevant distractor objects, and partial occlusions. SAM3D-VLA improves the average success rate from 50.2% to **65.2%** under ST and from 21.3% to **44.3%** under OC, with clear gains on long-horizon tasks: cooking, inserting flower and stacking. These results suggest that subtask-aware SAM3D supervision helps the policy focus on the correct target object at each stage and encode object-level shape and layout priors, improving real-world robustness while preserving the original RGB-language-to-action inference pipeline without masks, SAM3D, depth maps, or point clouds during deployment.

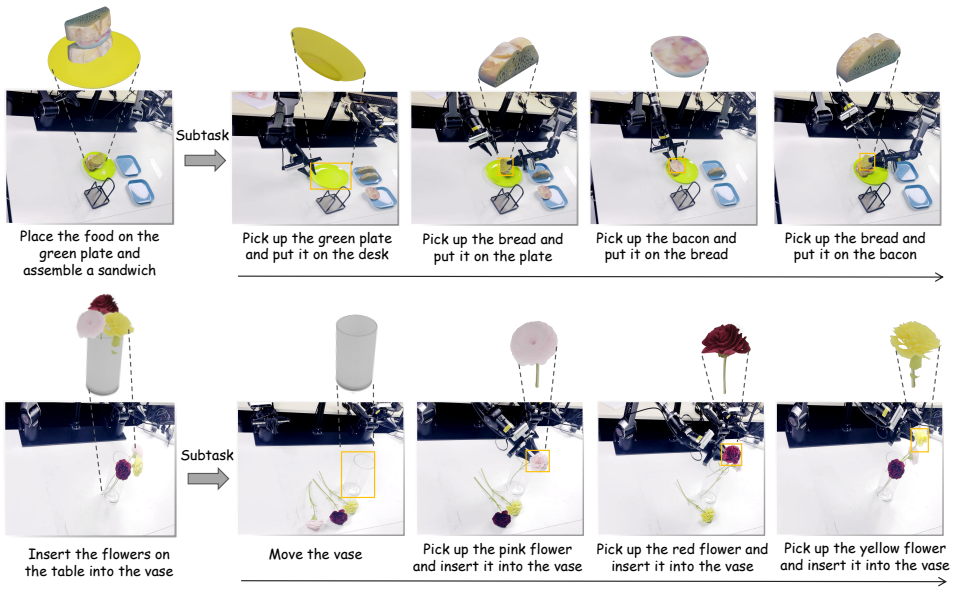


Figure 4: **Subtask-aware object-centric data processing**. High-level long-horizon instructions are decomposed into subtasks, and each subtask is associated with its corresponding target object mask. During training, the model receives stage-specific SAM3D supervision, encouraging it to focus on different task-relevant objects across manipulation steps.

## 5 Conclusion

In this paper, we propose SAM3D-VLA, an object-centric 3D representation alignment framework for vision-language-action models. By using a frozen SAM3D teacher during training, our method distills target-object 3D priors into intermediate VLA features while preserving the original RGB-language-to-action inference pipeline. For long-horizon tasks, we further decompose high-level instructions into subtasks and provide stage-specific object-centric supervision. Experiments on LIBERO, CALVIN, and real-world Piper-X tasks show that our method consistently improves manipulation performance, especially under long-horizon and object-centric generalization settings. These results suggest that training-time object-level 3D priors are an effective way to enhance VLA policies without requiring depth, point clouds, masks, or 3D modules during deployment.

## 6 Limitations

Although SAM3D-VLA improves VLA policies with object-centric 3D priors, it still has limitations. The training pipeline depends on automatically generated subtask annotations and object masks, so errors in decomposition, grounding, or segmentation may introduce noisy supervision. SAM3D also operates on single images, which may limit its reliability under severe occlusion, transparent objects, or poor viewpoints. In addition, our real-world evaluation is limited to tabletop tasks on one robot platform. While this work builds upon $\pi_0$, the proposed alignment framework is not restricted to it. Future work will improve subtask-object association, explore multi-view or temporal 3D teachers, and evaluate the method on other mainstream VLA backbones and broader robot settings.

**Acknowledgments**

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

## References

[1] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

[2] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[3] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. $\pi_0$: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[4] C. Li, J. Wen, Y. Peng, Y. Peng, and Y. Zhu. Pointvla: Injecting the 3d world into vision-language-action models. *IEEE Robotics and Automation Letters*, 11(3):2506–2513, 2026.

[5] P. Li, Y. Chen, H. Wu, X. Ma, X. Wu, Y. Huang, L. Wang, T. Kong, and T. Tan. Bridgevla: Input-output alignment for efficient 3d manipulation learning with vision-language models. *Advances in Neural Information Processing Systems*, 38:63635–63673, 2026.

[6] L. Sun, B. Xie, Y. Liu, H. Shi, T. Wang, and J. Cao. Geovla: Empowering 3d representations in vision-language-action models. *arXiv preprint arXiv:2508.09071*, 2025.

[7] V. Bhat, Y.-H. Lan, P. Krishnamurthy, R. Karri, and F. Khorrami. 3d cavla: Leveraging depth and 3d context to generalize vision language action models for unseen tasks. *arXiv preprint arXiv:2505.05800*, 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] F. Li, W. Song, H. Zhao, J. Wang, P. Ding, D. Wang, L. Zeng, and H. Li. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. *arXiv preprint arXiv:2510.12276*, 2025.

[10] W. Song, Z. Zhou, H. Zhao, J. Chen, P. Ding, H. Yan, Y. Huang, F. Tang, D. Wang, and H. Li. Reconvla: Reconstructive vision-language-action model as effective robot perceiver. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 18549–18557, 2026.

[11] H. Huang, X. Chen, Y. Chen, H. Li, X. Han, Z. Wang, T. Wang, J. Pang, and Z. Zhao. Roboground: Robotic manipulation with grounded vision-language priors. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22540–22550, 2025.

[12] M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*, 2024.

[13] S. Deng, M. Yan, S. Wei, H. Ma, Y. Yang, J. Chen, Z. Zhang, T. Yang, X. Zhang, W. Zhang, et al. Graspvla: a grasping foundation model pre-trained on billion-scale synthetic action data. *arXiv preprint arXiv:2505.03233*, 2025.

[14] Y. Tian, Q. Ye, and D. Doermann. Yolov12: Attention-centric real-time object detectors. *Advances in neural information processing systems*, 38:78433–78457, 2026.

[15] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European conference on computer vision*, pages 38–55. Springer, 2024.

[16] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, et al. Sam 2: Segment anything in images and videos. In *International Conference on Learning Representations*, volume 2025, pages 28085–28128, 2025.

[17] Y. Yang, X. Wu, T. He, H. Zhao, and X. Liu. Sam3d: Segment anything in 3d scenes. *arXiv preprint arXiv:2306.03908*, 2023.

[18] Z. Li, L. Ren, J. Yang, Y. Zhao, X. Wu, Z. Xu, X. Bai, and H. Zhao. Vip: Vision instructed pre-training for robotic manipulation. *arXiv preprint arXiv:2410.07169*, 2024.

[19] H. Chen, B. Sun, A. Zhang, M. Pollefeys, and S. Leutenegger. Vidbot: Learning generalizable 3d actions from in-the-wild 2d human videos for zero-shot robotic manipulation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 27661–27672, 2025.

[20] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11975–11986, 2023.

[21] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

[22] 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*, 44(10-11):1684–1704, 2025.

[23] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[24] Z. Hou, T. Zhang, Y. Xiong, H. Duan, H. Pu, R. Tong, C. Zhao, X. Zhu, Y. Qiao, J. Dai, et al. Dita: Scaling diffusion transformer for generalist vision-language-action policy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7686–7697, 2025.

[25] Q. Zhao, Y. Lu, M. J. Kim, Z. Fu, Z. Zhang, Y. Wu, Z. Li, Q. Ma, S. Han, C. Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1702–1713, 2025.

[26] Q. Bu, Y. Yang, J. Cai, S. Gao, G. Ren, M. Yao, P. Luo, and H. Li. Univla: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025.

[27] M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[28] Y. Du, S. Yang, B. Dai, H. Dai, O. Nachum, J. Tenenbaum, D. Schuurmans, and P. Abbeel. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36:9156–9172, 2023.

[29] K. Black, M. Nakamoto, P. Atreya, H. Walke, C. Finn, A. Kumar, and S. Levine. Zero-shot robotic manipulation with pre-trained image-editing diffusion models. In *International Conference on Learning Representations*, volume 2024, pages 33431–33452, 2024.

[30] H. Wu, Y. Jing, C. Cheang, G. Chen, J. Xu, X. Li, M. Liu, H. Li, and T. Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, volume 2024, pages 10641–10662, 2024.

[31] Y. Wen, J. Lin, Y. Zhu, J. Han, H. Xu, S. Zhao, and X. Liang. Vidman: Exploiting implicit dynamics from video diffusion model for effective robot manipulation. *Advances in Neural Information Processing Systems*, 37:41051–41075, 2024.

[32] Q. Bu, J. Zeng, L. Chen, Y. Yang, G. Zhou, J. Yan, P. Luo, H. Cui, Y. Ma, and H. Li. Closed-loop visuomotor control with generative expectation for robotic manipulation. *Advances in Neural Information Processing Systems*, 37:139002–139029, 2024.

[33] W. Zhao, P. Ding, Z. Min, Z. Gong, S. Bai, H. Zhao, and D. Wang. Vlas: Vision-language-action model with speech instructions for customized robot manipulation. In *International conference on learning representations*, volume 2025, pages 51676–51693, 2025.

[34] X. Li, M. Liu, H. Zhang, C. Yu, J. Xu, H. Wu, C. Cheang, Y. Jing, W. Zhang, H. Liu, et al. Vision-language foundation models as effective robot imitators. In *International Conference on Learning Representations*, volume 2024, pages 26703–26721, 2024.

[35] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

[36] 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. In *International Conference on Learning Representations*, volume 2025, pages 54277–54296, 2025.

[37] 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. *arXiv preprint arXiv:2501.09747*, 2025.

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

# Supplementary Material

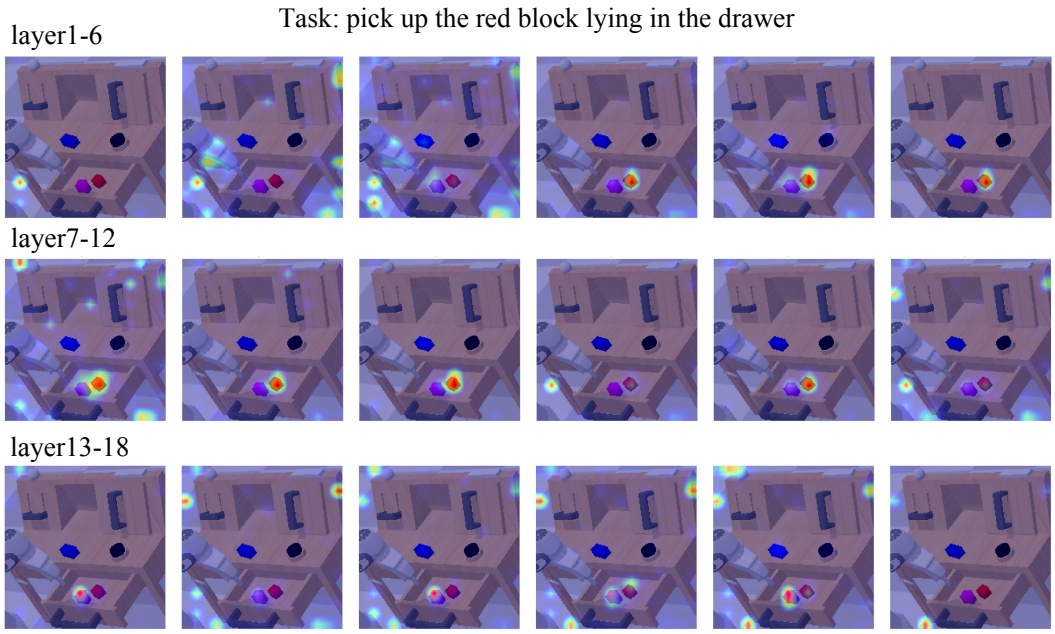


Figure 5: **Layer-wise target-object attention visualization.** We visualize the attention from the target-object language tokens to image tokens across all 18 VLA layers. Shallow layers already show certain responses around the target object, while the attention becomes more concentrated in middle-to-deep layers. In particular, the 11th layer provides a clearer focus on the task-relevant object while still preserving visual spatial details, supporting our choice of using this layer for SAM3D-guided alignment.

### 6.1 Analysis of Alignment Layer Selection

We further analyze which VLA layer is more suitable for SAM3D-guided feature alignment. Our $\pi_0$ VLM backbone contains 18 causal attention layers. As shown in Fig. 5, shallow layers already show some attention to the target object, but their responses are still mixed with low-level visual cues and local texture patterns. More importantly, if the alignment is applied to very shallow layers, the injected object-centric 3D information has to pass through many subsequent transformer layers, where it may be weakened or overwritten during visual-language and action-related feature fusion. Therefore, shallow-layer supervision may not be effectively preserved in the final policy representation.
supervision.

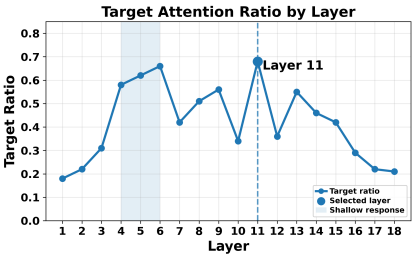


Figure 6: **Target-object attention ratio across layers.** Layer 11 achieves the highest ratio, while shallow layers also show non-negligible target responses.

On the other hand, the last layers are closer to action prediction and language-conditioned decision making. At this stage, visual and language features tend to become more modality-agnostic, and part of the vision-specific spatial details may be lost. To further quantify this trend, we select ten tasks from CALVIN and compute the proportion of attention mass falling inside the target-object region for each layer. As shown in Fig. 6, the target attention ratio peaks at the 11th layer, while shallow layers such as Layers 4–6 also show relatively high but less stable target responses. We therefore choose the 11th layer as the alignment layer, which provides a good trade-off between target-object awareness and vision-specific spatial representation, making it suitable for SAM3D object-centric supervision.

Table 4: Ablation study on the choice of SAM3D alignment layer on the CALVIN benchmark. **Bold** denotes the best performance.

| Alignment Layer | Splits | Success Rate (%) | | | | | Avg. Len |
|---|---|---|---|---|---|---|---|
| | | 1/5 | 2/5 | 3/5 | 4/5 | 5/5 | |
| Layer 9 | ABC→D | 95.4 | 87.8 | 78.2 | 70.1 | 66.5 | 3.98 |
| Layer 10 | ABC→D | 95.8 | 88.3 | 79.1 | 71.4 | 68.4 | 4.03 |
| Layer 11 | ABC→D | **96.2** | **89.1** | **80.5** | **73.6** | **71.6** | **4.11** |
| Layer 12 | ABC→D | 95.7 | 88.0 | 78.8 | 70.6 | 66.9 | 4.00 |
| Layer 13 | ABC→D | 95.2 | 87.1 | 77.6 | 69.2 | 65.9 | 3.95 |

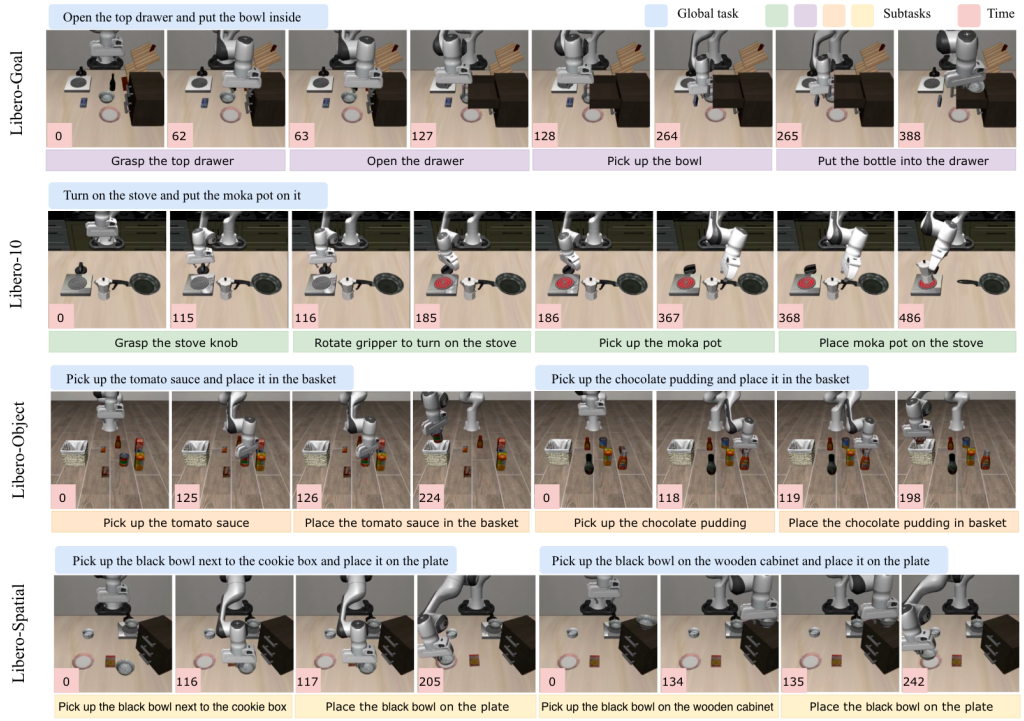


Figure 7: **Subtack-processing on Simulation Benchmark** We use a large language model (LLM) to further decompose each demonstration trajectory and its high-level task instruction into subtasks,

### 6.2 Subtask-aware Data Processing

We employ a large language model (LLM) to break down each demonstrated trajectory along with its high-level task instruction into a sequence of subtasks. For every subtask, the LLM also provides a natural-language description and the indices of its starting and ending frames. Concretely, given the high-level instruction, we feed the LLM a sampled set of video frames together with the task context, and ask it to partition the entire demonstration into a number of atomic subtasks. The LLM returns a JSON list of tuples, each in the form $\{(\ell_i, s_i, e_i)\}_{i=1}^{M}$, where $i$ indexes the subtask, $\ell_i$ is the corresponding subtask instruction, and $s_i, e_i$ denote the start and end frames. To ensure consistent granularity, we explicitly constrain the prompt so that the generated subtasks are aligned with a small set of primitive manipulation actions, rather than overly detailed intermediate descriptions. For example, for pick-and-place tasks, we standardize subtask expressions such as "Pick up [object]", "Place [object] on [target position]", "Open/Close [object]", and "Push [object]". Under this rule, a full task is typically decomposed into 2 to 5 subtasks. Details of the prompt design are provided in the supplementary material. In our implementation, we use GPT-4o as the LLM for subtask decomposition and annotation.