6. The Use of Large Language Models

During the preparation of this manuscript, we used large language models (LLMs) solely for text polishing and grammatical correction. The model design, experimental validation, and analytical reasoning presented in this work were entirely developed and executed by the authors without reliance on LLMs for scientific content generation. All research motivation, theoretical formulation, and extensive empirical evaluation are the independent intellectual contributions of the authors.

7. Implementation Details

7.1. Hybrid Datasets

Large-scale pre-training on multimodal datasets collected from diverse sources has demonstrated strong generalization capabilities in large language models (LLMs) and vision-language models (VLMs). To establish a unified paradigm for task- and cross-embodiment generalist robotic policies, we integrate visual, instructional, and action data from heterogeneous sources. Specifically, we construct a complementary multimodal dataset from the following three categories for use in the warmup and fine-tune stages: (1) Human demonstration video datasets collected from first-person perspectives, including Something-Something-v2 [18] and Ego4D [19]; (2) Robot operation videos and action trajectories in real-world environments, including the large-scale open-world manipulation dataset Open X-Embodiment (OXE) [44] and the reasoning-driven robotic benchmark RoboMIND [60]; (3) Robot operation videos and action trajectories in simulated environments, including CALVIN-ABC [42].

These sources encompass a wide range of agent hardware configurations and diverse task scenarios. To train the model to establish a unified motion perception from extensive multimodal data, we address discrepancies in collection frequency across sub-datasets. Specifically, to ensure a consistent 1-second time interval between keyframes, we set a dedicated frame sampling stride for each dataset. To maintain input consistency, we filter out video sequences lacking high-quality textual annotations and retain only those with more than 12 frames. We summarize the distribution of episodes and frames, along with their relative proportions, across these datasets in Table 10. To appropriately balance the contribution of data collected from different sources and modalities during training, we select specific subsets from OXE [44] and RoboMIND [60] to form our mixed training set. These strategies ensure that the model is consistently exposed to a diverse and balanced mixture of samples in every training iteration. This design is crucial for stabilizing optimization and preventing domain over fitting during large-scale pre-training.

7.2. Warmup and Co-train Details

To maintain consistency, we resize all camera visual observations to 256×256. VITA is configured to predict 12 future frames in parallel during inference, with a maximum sequence length of 8,000 tokens. The model is trained on 16 NVIDIA A100 GPUs. In the warmup stage, we perform 200K steps of full-parameter training with 324M trainable parameters, requiring approximately 1.5 days. In the co-train stage, we conduct 100K steps of full-parameter training with 2.8B trainable parameters, taking about 3.5 days. Additional training hyperparameters are provided in Table 11.

7.3. Finetune Details

We evaluate VITA in both simulated and real-world environments. Prior to evaluating performance on simulated benchmarks (e.g., CALVIN [42], SimplerEnv [33], and LIBERO [36]) and real-world tasks, we first fine-tune the model parameters. During fine-tuning, we load and freeze the pre-trained VLM Backbone visual decoder, while only fine-tuning the action decoder. Specifically, for evaluation on the CALVIN benchmark, we fine-tune the model on the CALVIN-ABC dataset and then evaluate its performance on the full CALVIN benchmark. Similarly, we fine-tune on the real-world BridgeV2 [54] dataset and Fractal [8] dataset and evaluate on the SimplerEnv benchmark to validate the model's real-to-sim transfer capability. Additionally, we collect over 3,300 real-world robot operation videos paired with action trajectories using a UR-5e robotic arm. The details of the dataset collection protocol can be found in Section 10.2. These real-world datasets are also used to fine-tune both VITA and other baseline models.

Specifically, for evaluation on the CALVIN benchmark [42], we fine-tune the model on the CALVIN-ABC dataset using RGB observations from both third-person and robot-mounted (gripper) cameras. We set the batch size to 192 and train for 40,000 steps, with each task sequence having a length of 5. For the LIBERO benchmark [36], comprising four task suites, we similarly fine-tune using RGB images from third-person and gripper viewpoints, with a batch size of 192 and 60,000 training steps.

Most notably, the real-to-sim evaluation on SimplerEnv benchmark [33] trains policies on real-world data and tests them in simulation. The test scenarios are divided into two categories based on data source: Google Robot (Fractal) and WidowX (Bridge V2). The Fractal split has a limited domain gap between training and testing, whereas the Bridge V2 branch presents a more challenging transfer scenario. For the Fractal benchmark, we use single-view RGB observations, with a batch size of 128 and train for 50,000 steps using an action chunk size of 5. For the WidowX (Bridge V2) branch, we extend the training to 80,000 steps under the same configuration. During evaluation, we con-

Table 10. We construct a hybrid multimodal dataset for training VITA, comprising: (1) human demonstration video datasets, Something-Something-v2 [18] and Ego4D [19]; (2) Robot video-action datasets from real-world settings, Open X-Embodiment (OXE) [44] and RoboMIND [60]; (3) Robot video-action datasets from simulated environments, CALVIN [42] and LIBERO [36].We report the number of videos and total frames in each dataset and compute the relative proportion of each sub-dataset based on frame count.

Datasets	Source	Videos	Frames	Proportion
(1) Human demonstration datasets				
Something-Something-v2 [18]	-	108499	10581365	20.47%
Ego4D [19]	-	59427	14202926	27.68%
(2) Real-world robot datasets				
Bridge [16, 54]	OXE [44]	25460	813372	1.56%
Fractal [8]	OXE [44]	87212	3786400	7.41%
FMB Dataset [40]	OXE [44]	8611	1137340	2.14%
Kuka [25]	OXE [44]	209880	2455879	4.87%
Berkeley Gnm Recon [47]	OXE [44]	11834	610907	1.17%
DROID [27]	OXE [44]	92233	27044326	5.26%
Single-Arm Franka	RoboMIND [60]	16018	2268033	4.48%
Single-Arm UR-5e	RoboMIND [60]	25721	2643322	5.07%
AgileX Cobot Magic V2.0	RoboMIND [60]	10059	6477564	12.67%
(3) Simulation robot datasets				
CALVIN-ABC [42]	-	-	1944237	3.70%
LIBERO [36]	-	6500	1865203	3.51%

duct 24 randomly initialized rollouts across four tasks, Pick Carrot, Pick Eggplant, Pick Spoon, and Stack Cube, and report the final success rate as the performance metric.

Throughout all fine-tuning stages, to maintain consistency with the pre-training process, we uniformly resize all camera visual observations to 256×256.

7.4. Inference

Our approach adopts a scheme that simultaneously generates future video frames and aligned action trajectories during training, requiring supervision from both future visual frames and ground-truth actions. During inference, VITA retains only the action decoder to produce action tokens, while future frame generation is disabled, enabling efficient and low-latency motor command execution.

7.5. Multimodal input protocol

7.5.1. Input frame

We follow the multi-frame observation processing protocols of existing methods [37, 52] to ensure fair comparison. For GR00T N1.5 [6], we concatenate historical observation frames and input them into the three original visual views (left, right, wrist) to enable multi-view modeling. For CogACT [31], since its original VLM supports only single-view, single-timestep processing, we extract the cognitive tokens at each timestep and perform temporal concatenation, thereby providing consistent cross-view semantic constraints for the action expert. If a particular view is unavailable in the dataset, the corresponding channel is zero-padded and masked via attention masking to maintain input format consistency.

7.5.2. State and Action

For state and action inputs, we construct a unified vector representation capable of accommodating both joint angles and end-effector signals.

7.5.3. Embodiment

In single-arm demonstrations, the available arm is mapped to the right-arm channel, while the left-arm channel is zero-padded with masking to maintain compatibility with the dual-arm setup.

8. Architecture of VITA

8.1. Modules utilized in warmup stage

To obtain rich semantic representations, we self-supervisedly train a visual decoder in the DINOv2 [43] feature space as a forward dynamics model. To achieve cross-embodiment generalizable robot action representations, we first normalize multi-source action trajectories

and then train an action decoder in the frequency-domain feature space obtained via Discrete Cosine Transform (DCT) [2] to model inverse dynamics.

Specifically, the visual encoder is built upon a pretrained DINOv2 encoder and an M-Former [12] module. The DINOv2 encoder extracts motion-aware spatiotemporal features from consecutive video frames, and its parameters are frozen during the warmup stage to ensure that training focuses solely on reconstructing dynamic visual content rather than relearning static visual priors. In contrast, the visual decoder adopts a Vision Transformer (ViT) [13] architecture with 12 transformer layers. Notably, we deliberately design this decoder to be lightweight compared to the encoder's capacity. This design reflects the auxiliary role of visual prediction in VLA models: rather than serving as an explicit reconstruction target, it acts as an implicit inductive bias that enhances the model's ability to reason about complex spatial scenes and generate robust motor commands.

Our action encoder consists of a lightweight MLP paired with a Discrete Cosine Transform (DCT), while the action decoder comprises multiple Transformer layers followed by an Inverse DCT. The action decoder is deliberately designed with a larger parameter capacity because it bears the core responsibility of accurately reconstructing high-dimensional, continuous, and temporally coherent robot action trajectories from the shared discrete latent space. It focuses on capturing fine-grained dynamics from contextual inputs, such as visual observations and task instructions, and ensures that the generated actions are physically smooth and executable.

In contrast, the action encoder only needs to extract compact latent representations, requiring significantly fewer parameters. By concentrating model capacity in the decoder, VITA achieves an optimal trade-off: the encoder remains lightweight for efficient training and storage, while the heavy-duty decoder guarantees high precision and robustness in the final motor commands.

During the entire warmup stage, all model parameters are fully trainable except for the DINOv2 encoder, which remains frozen to preserve its rich visual priors and stabilize representation learning.

8.2. Modules utilized in co-train stage

To ensure that performance gains stem from modality alignment and the implicit visual chain-of-thought design rather than architectural idiosyncrasies, our VITA implementation follows mainstream design choices established in works like Pi0 [7]. Specifically, we initialize the vision-language model (VLM) with the pretrained PaliGemma [4], which comprises the Gemma [51] language backbone and SigLIP [64] as the visual encoder. During the co-train and fine-tuning stage, we continue training the VLM backbone, visual decoder, and action decoder on synchronized robot video–action trajectory datasets.

8.3. Demonstration and reasoning stage

In demonstration, we retain only the PaliGemma-based backbone, visual decoder, and action decoder. Notably, the visual decoder is used solely for visualization to provide users with interpretable previews of anticipated scene outcomes based on the generated actions. However, during actual inference for control, the visual decoder is discarded, and only the action decoder is used to produce executable motor commands, ensuring low-latency operation.

8.4. Model architecture details and training hyperparameters

To facilitate reproducibility, we detail the architecture and training hyperparameters of all VITA components in Table 11. Specifically, The SigLIP-based visual encoder contains 400 million parameters. The Gemma Transformer backbone has approximately 2 billion parameters. Built upon these strong vision–language priors, we design a 96 million visual decoder and a 228 million action decoder to effectively capture visual dynamics and motion representations, respectively. Furthermore, we provide the pseudocode for the warmup, co-train and fine-tune stage in Algorithm 1.

9. Details of the Simulation Environment

9.1. CALVIN

The CALVIN simulation benchmark is designed for learning long-horizon robotic control policies and comprises four distinct manipulation environments. In each environment, multimodal inputs, including language instructions, multi-view RGB-D cameras, tactile, and proprioceptive sensors, are provided. CALVIN supports 34 manipulation tasks and 1,000 language instructions. In the most challenging ABC-D setup, models are first trained in the ABC environments, followed by evaluation on task completion in environment D, assessing the model's capability for long-horizon planning and zero-shot generalization.

9.2. LIBERO

The LIBERO benchmark comprises four long-horizon robotic manipulation test suites, each containing 10 tasks with 50 human demonstrations per task. These suites emphasize distinct aspects of robot control: (1) LIBERO-Goal evaluates goal-conditioned behavior by varying task objectives; (2) LIBERO-Spatial assesses spatial reasoning through changes in object layout while keeping the target objects fixed; (3) LIBERO-Object tests object-level generalization by altering object instances within a fixed scene configuration; (4) LIBERO-Long features long-horizon, compositional tasks that require complex sequential planning and execution.

Algorithm 1 VITA Training and Inference Pipeline

Require: Dataset D
Require: Shared codebook $\mathcal{C} = \{c_k\}_{k=1}^{K}$, visual encoder $\mathcal{E}_v$, action encoder $\mathcal{E}_a$
Require: Visual decoder $\mathcal{D}_v$, action decoder $\mathcal{D}_a$
/* Stage 1: Warmup */
for each batch from D **do**
  if sample is video pair (I_t, I_{t+1}) **then**
    $z_v \leftarrow$ M-Former$([DINOv2(I_t); DINOv2(I_{t+1})])$
    $\hat{z}_v \leftarrow \mathcal{Q}(z_v)$
    $\hat{I}_{t+1} \leftarrow \mathcal{D}_v(DINOv2(I_t), \hat{z}_v)$
    $L_v = \lambda_{L1}\|I_{t+1} - \hat{I}_{t+1}\|_1 + \lambda_{SSIM}(1 - SSIM(I_{t+1}, \hat{I}_{t+1}))$
    Update $\mathcal{E}_v, \mathcal{D}_v, \mathcal{C}$ via ∇L_v
  else if sample is action segment $a_{t:t+H}$ **then**
    $\hat{a} \leftarrow$ DCT$(a_{t:t+H})$
    $z_a \leftarrow$ MLP(flatten$(\hat{a})$)
    $\hat{z}_a \leftarrow \mathcal{Q}(z_a)$
    $\hat{a}_{t:t+H} \leftarrow$ MLP(InverseDCT$(\hat{z}_a)$)
    $L_a = \|a_{t:t+H} - \hat{a}_{t:t+H}\|_2^2$
    Update $\mathcal{E}_a, \mathcal{D}_a, \mathcal{C}$ via ∇L_a
  end if
end for
/* Stage 2: Co-train */
for each batch from D **do**
  Input: instruction x, image I_0, state s
  $h_{\text{ctx}} \leftarrow [\mathcal{T}_{\text{text}}(x); \mathcal{T}_{\text{image}}(I_0); s]$
  Generate textual CoT: $Z_{\text{sub}} = [z_1, ..., z_M]$ via $\arg\max p_\theta(z \mid h_{\text{ctx}})$
  $h_{\text{mul}} \leftarrow [h_{\text{ctx}}; \mathcal{T}_{\text{text}}(Z_{\text{sub}})]$
  Autoregressively sample hybrid tokens: $\{\tau_i\}_{i=1}^{L} \sim p_\theta(\cdot \mid h_{\text{mul}})$
  Decode: $\hat{I}_{1:T} \leftarrow \mathcal{D}_v(\{c_{\tau_i}\})$, $\hat{a}_{1:H} \leftarrow \mathcal{D}_a(\{c_{\tau_i}\})$
  if only video available **then**
    $L_{\text{co}} \leftarrow \sum_{t=1}^{T}(\lambda_{L1}\|I_t - \hat{I}_t\|_1 + \lambda_{SSIM}(1 - SSIM(I_t, \hat{I}_t)))$
  else
    $L_{\text{co}} \leftarrow \lambda_v\|I_{1:T} - \hat{I}_{1:T}\|_1 + \lambda_a\|a_{1:H} - \hat{a}_{1:H}\|_2^2$
  end if
  Update p_θ, $\mathcal{D}_v$, $\mathcal{D}_a$ (freeze $\mathcal{C}$)
end for
/* Stage 3: Fine-tune */
for each batch from task-specific data **do**
  Freeze p_θ and $\mathcal{C}$
  Update only $\mathcal{D}_a$ using $\|a_{1:H} - \hat{a}_{1:H}\|_2^2$
end for
/* Inference */
Require: x, I_0, s
Generate $\{\tau_i\}_{i=1}^{L} \sim p_\theta(\cdot \mid h_{\text{mul}})$ as in Co-train
$\hat{a}_{1:H} \leftarrow \mathcal{D}_a(\{c_{\tau_i}\})$
return $\hat{a}_{1:H}$

9.3. SimplerEnv

To address the limitations of simulated environments in terms of diversity in lighting, color, texture, and robot camera viewpoints compared to real-world scenarios, SimplerEnv integrates multiple robot configurations. Specifically, to compare with state-of-the-art generalist manipulation policies, we evaluate VITA's performance under two settings: Google Robot and WidowX.

Table 11. We demonstrated the details of the training hyperparameters and the model architecture.

Hyperparameters of pretraining	**Value**
Batch size	1024
Learning rate	1e-4
Warmup iterations	200K
Co-train iterations	100K
Model precision	Bf16
Optimizer	AdamW
Weight decay	0.01
Optimizer momentum	$\beta_1, \beta_2 = 0.9, 0.95$
Model Architecture	**Value/Parameters**
Visual Module	
DINOv2-Base	86M
M-Former	20M
Visual decoder layer	12
Visual decoder hidden dim	768
Visual decoder parameter	96M
Visual optimization objective	L1 loss + SSIM loss
Image size	256×256
Predicted future frames	12
Action Module	
Action encoder parameter	10M
Action decoder layer	12
Action decoder hidden dim	1024
Action decoder parameter	228M
Action optimization objective	MSE loss
Action dimension	7
Sampling steps	12
Shared quantizer	
Size of codebook	8192
Dimension of encoding vector	1024
VLM backbone	
SigLIP (Visual Tokenizer)	400M
Gemma backbone	2B
Total parameters	2.8B

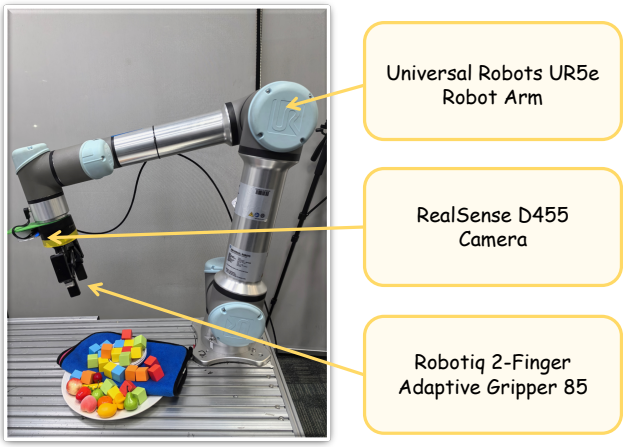

Figure 5. Demonstration of our real-world robotic platform.

10. Real-world robot platform

10.1. UR-5e Experimental Setup

As shown in Figure 5, we use a Universal Robots UR5e (UR-5e) robotic arm equipped with a Robotiq 2-Finger Adaptive Gripper 85. The robot is configured with two cameras providing distinct viewpoints: a first-person view (camera mounted above the end-effector) and a third-person view (camera fixed at the edge of the table). Additionally, we present a series of manipulation tasks collected from real-world scenarios in Figure 6.

10.2. Real-World Train Dataset Protocol

We summarize the real-world data collected from the UR-5e robotic platform in Table 12. All robot control commands are executed at an actual frequency of 30 Hz. We designed a total of 12 tasks, categorized into four types, covering a full spectrum of capabilities from basic manipulation to long-horizon reasoning. We have clearly defined these 12 task scenarios, including execution steps, success criteria, failure conditions, and instruction templates. Details can be found in Section 10.3. Each task includes 200–800 randomized scenes to ensure generalization. For each task, we collect 3,300 robot action trajectories paired with synchronized operation videos by varying the initial positions of objects on the tabletop, forming the UR-5e dataset.

Our randomization strategy is as follows: (1) In each scene, object initial positions are uniformly sampled within the 0.8m×0.8m workspace to avoid fixed patterns; (2) Towels and plates are randomly offset by ±15 cm within each scene; (3) While instruction templates remain consistent across different instances of the same task, object combinations, quantities, and color distributions are randomly varied to enhance diversity and test robustness.

In our data collection protocol, we operate at the granularity of the robot's minimal executable action: for every

(a) The real-world dataset collected on the UR-5e robot platform

(b) The simulated environment consisting of CALVIN, LIBERO and SimplerEnv

Figure 6. (a) Based on the established real-world robotic platform, we designed multiple scenarios to collect training data for the UR-5e agent. (b) Furthermore, we also presented the benchmark in the simulation environment.

Table 12. Real-World task trajectories collected from the Robots UR-5e. In the second column of the table, we use numbers to represent the corresponding training tasks. The complete definitions of the 12 tasks can be found in Section 10.3.

Task type	Task ID	Scenes	Frames/Scene	Frames	Proportion
Basic Manipulation and Short-term Reasoning	Task_{1-5}	300	209	314,592	27.1%
Multi-step Execution	Task_{6-8}	200	473	283,607	24.4%
Textual-Visual Reasoning	Task_{9-10}	400	321	256,854	22.1%
Long-horizon Planning	Task_{11-12}	200	768	307,364	26.4%
UR-5e Single Arm Dataset (Total)	-	3,300	352	1,162,417	100%

teleoperated command issued, we synchronously capture one RGB frame and record the full robot state, including joint angles, end-effector pose, and gripper width. The resulting dataset comprises 1,162,417 time steps, each strictly aligned with one robot action command and one 640×480 RGB image captured by a RealSense D455 camera, along with the complete proprioceptive state at that moment.

During preprocessing, each RGB image is resized to 256×256 (uint8) to match the model input specification, and the corresponding depth map is also downsampled to 256×256 (float32). Each robot action is represented as a 7-dimensional float32 vector. Additionally, we retain comprehensive metadata for every time step, including: full robot state, natural language instruction (average length 50 characters), timestamp, and camera calibration parameters.

10.3. Real-World Task Definition

In this section, we provide detailed designs for each task, including execution steps, success criteria, failure conditions, and language instruction templates, to facilitate data annotation and model evaluation.

Task 1: Select Celebrity Portrait

The robot must identify a specified celebrity portrait (e.g., "Taylor Swift") from a set of displayed images. It then moves its gripper to a position 5 cm above the target portrait and slowly lowers it until it makes light contact with the surface, applying a controlled force of no more than 2 N. Success is defined as stable contact with the correct por-

trait without slippage or tipping. Failure occurs if the gripper touches an incorrect portrait, applies excessive force, or fails to make any contact. The instruction template includes: "Please gently touch Taylor Swift's portrait." and "Point to Anne Hathaway's photo with your finger."

Task 2: Color Match to Towel
The robot identifies two colored blocks (red and blue) and two corresponding colored towels. It must place the red block entirely on the red towel and the blue block on the blue towel. Success requires both blocks to be correctly positioned with no misplacement. Failure occurs if blocks are placed on the wrong towel, fall off during placement, or cause the towel to be pulled or deformed. Instruction templates include: "Place the red block on the red towel and the blue block on the blue towel." and "Sort by color: red to red towel, blue to blue towel."

Task 3: Color Match to Plate
Identical to Task 2, but the target containers are plates instead of towels. The robot must place the red block inside the red plate and the blue block inside the blue plate. Success is achieved when both blocks are fully contained within their respective plates, with no part of the block extending beyond the rim. Failure occurs if blocks roll out of the plates or are placed on the wrong plate. Instruction templates include: "Put the red block into the red plate and the blue block into the blue plate." and 'Sort the blocks by color into the corresponding plates."

Task 4: Fold Towel
The robot grasps one end of a flat towel and folds it once along its central axis, aligning the edges as precisely as possible. Success is defined as a neat, rectangular fold with minimal wrinkles or distortion. Failure occurs if the fold is uneven, the towel slips from the gripper, or multiple folds are attempted. The instruction template is: "Please fold this towel once and make it neat."

Task 5: Stack Cups/Bowls
The robot identifies a stack of paper cups or bowls and sequentially lifts and places each item on top of the previous one to form a stable vertical tower. Success requires the final stack to remain upright and intact without any collapse. Failure occurs if the stack topples during or after placement. The instruction template is: "Please stack the paper cups one by one."

Task 6: Error Correction (Color Matching)
The robot observes a scene where some blocks are incorrectly placed (e.g., three blue blocks and one red block on a blue towel). It must detect the anomaly (the red block on the blue towel), retrieve it, and relocate it to the correct towel (red towel). Success is achieved when all blocks are correctly matched to their respective colors with no omissions or new errors. Failure occurs if the robot fails to detect the error, moves a correct block, or creates a new misplacement. Instruction templates include: "Check if any blocks are misplaced? If so, correct them." and "There's a red block on the blue towel, which is wrong. Please move it to the red towel."

Task 7: Wipe Plate (Anti-Tip Test)
A plate is balanced precariously atop three stacked blocks, creating an unstable platform. The robot must use a gray towel to gently wipe the surface of the plate without disturbing the blocks beneath. Success is defined as the plate remaining upright and the towel completing a full wiping motion. Failure occurs if the plate tips over, the blocks are displaced, or the towel fails to contact the plate surface. Instruction templates include: "Gently wipe this plate with the gray towel, be careful not to touch the blocks underneath." and "Be cautious, the plate is wobbly, don't let it fall."

Task 8: Multi-step Transfer (Large Plate $\rightarrow$ Small Plate)
The robot must perform a two-step sequence: First, transfer all blue blocks from the table into a large plate and wait for 5 seconds (simulating user confirmation). Then, transfer all blue blocks from the large plate into a smaller plate. Success requires both steps to be completed in sequence without interruption and with no blocks lost or dropped. Failure occurs if the robot begins the second step before the first is complete, or if blocks fall during either transfer. Instruction templates include: "First, put all blue blocks into the large plate, then transfer them all to the small plate." and "Step 1: Blue blocks into large plate; Step 2: Blue blocks into small plate."

Task 9: Return to Original Position (Towel $\rightarrow$ Plate $\rightarrow$ Towel)
The robot begins with red and blue blocks correctly placed on their respective colored towels. It must first collect all blocks and place them into a central plate, then return each block to its original towel. Success is defined as the final configuration matching the initial state exactly. Failure occurs if blocks are returned to the wrong towel or if any block is missing. Instruction templates include: "First, collect all blocks into the plate, then return them to their original positions." and "Remember where they started; they must return there at the end."

Task 10: Imitate Arrangement Strategy
The robot observes a reference arrangement in a blue basket, e.g., a stack of blocks from top to bottom: red, green, blue. It must then replicate this exact vertical order by arranging scattered blocks of the same colors into a red basket. Success is achieved when the sequence of colors in the red basket matches the reference sequence precisely. Failure occurs if the order is altered, inverted, or incomplete.

The instruction template is: "Look at how the blocks are arranged in the blue basket; replicate that arrangement with the blocks on the table in the red basket."

Task 11: Categorize by Object Type (Fruit/Blocks)
The robot identifies objects as either fruit (e.g., apple, banana) or blocks (e.g., colored cubes). It must place all fruits into a plate and all blocks onto a gray towel. Success requires accurate classification and correct placement with no misplacement. Failure occurs if fruits are placed on the towel or blocks are placed in the plate. Instruction templates include: "Put the fruits in the plate and the blocks on the gray towel." and "Distinguish between fruits and blocks: fruits go in the plate, blocks go on the towel."

Task 12: Categorize by Object Color (Red/Green)
The robot identifies red and green objects, including both fruits and blocks, and must place all red items into the plate and all green items onto the gray towel. Success is achieved through accurate color-based categorization with no misplacements or omissions. Failure occurs if red objects are placed on the towel, green objects are placed in the plate, or any object is left unsorted. Instruction templates include: "Put red items in the plate and green items on the towel." and "Regardless of type, put anything red in the plate and anything green on the towel."

10.4. Real-Robot Finetuning.

For practical evaluation, we conducted real-world experiments on the UR-5e robotic platform using images captured from two camera perspectives: one fixed at the edge of the tabletop (third-person view) and one mounted on the robotic arm (first-person view). All input images were resized to a resolution of 256×256. The model outputs a 7-dimensional action vector, with an action chunk size of 15. For each task, we used a learning rate of 5×10^{-5}, a batch size of 128, and trained for 40,000 steps.

10.5. Evaluation on the Real-World Experiment

We designed a series of representative and challenging real-world tasks to thoroughly evaluate VITA's performance in robotic motion planning under both in-distribution (ID) and out-of-distribution (OOD) settings. ID tasks are those that appear in the training set, whereas OOD tasks are held out during training and are specifically included to assess the model's generalization capability. These tasks span a broad spectrum of real-world scenarios, including dexterous manipulation, long-horizon action sequences, visual-contextual reasoning, and dynamic interaction with the environment. For each task, we report the average success rate over 100 repeated trials. Collectively, these experiments demonstrate VITA's robustness and generalization across diverse real-world conditions. The 4 ID tasks and 4 OOD tasks we designed are as follows:

Evaluation 1: Select Object (ID)
Select the Kaiming He's portrait and place a fruit on it.

Evaluation 2: Color Match (ID)
Place the red and blue blocks onto the towel of their corresponding colors.

Evaluation 3: Object Map (ID)
Put red objects on the towel, and yellow objects in the plate.

Evaluation 4: Visual Reason (ID)
Put the fruit into the plate that contains the most blue blocks.

Evaluation 5: Inverse Execution (OOD)
First move the blocks from the towel into the plate, then return them to their original positions.

Evaluation 6: Conditional Decision (OOD)
If the letter in the plate is "B", place a strawberry on plate.

10.6. Visualization of Real-World Task

In addition to quantitative results and analysis, we provide further visualizations in Figures 7-16 to demonstrate the reliability of deploying the VITA model in real-world scenarios. We analyze our real-world results as follows:

(1) In complex scenarios, the Internal CoT enables the model to capture three-dimensional spatial features, allowing it to generate precise action trajectories in 3D space and successfully complete challenging tasks such as "stacking multiple plates together."

(2) The visual context empowers the model to plan and search for potential solutions by leveraging longer-term historical spatial information—for example, "the model records the initial spatial relationships between objects and restores them to their original positions after task completion." This capability is critical for home environments, such as a robot returning jam and utensils to their original locations after assembling a sandwich.

Additionally, we present visualizations of VITA's generated action trajectories under simulation environments CALVIN, LIBERO, SimplerEnv Google Robot and WidowX in Figures 17-19.

10.7. Visualization of the Ablation Model

Figure 20 presents visual comparisons from our ablation studies. Specifically, we design an ablated variant "w/o modality alignment": this variant removes the shared codebook, skips the warmup stage, and proceeds directly to co-training and fine-tuning.

10.8. Real-Robot Latency

In the real-world experiments, the UR-5e robotic platform receives two image observations, each with a resolution of 256×256 pixels, and outputs discrete action chunks over a

Figure 7. VITA executes the "Color Matching" task. Following the instruction (e.g., "Place red and blue blocks on towels of matching colors"), VITA controls the gripper to place the red block on the red towel and the blue block on the blue towel, successfully completing the classification.

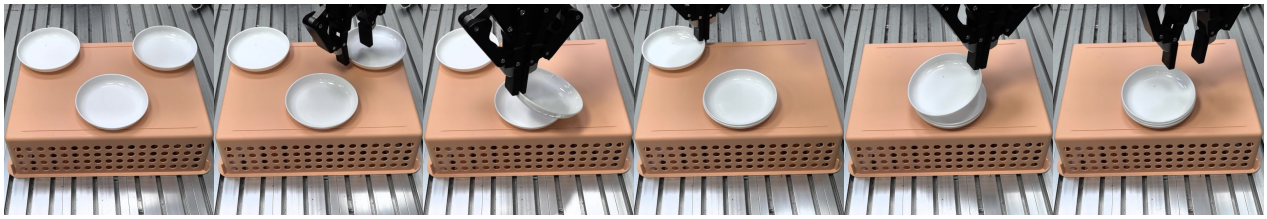

Figure 8. VITA executes the "Stack Plates" task. Following the instruction (e.g., "Stack the plates one by one"), VITA controls the gripper to stack them sequentially. This process demonstrates VITA's planning and execution capabilities in multi-step manipulation and physical interaction.

horizon of 15 time steps, where each chunk contains 12 consecutive action steps. Each inference step takes approximately 2 seconds on an NVIDIA RTX 4090 GPU (24GB), and considering communication and data I/O overhead, the overall system latency is around 3 seconds. The system operates at an action-chunk frequency of 5 Hz, while maintaining an average action-level inference throughput close to 60 Hz (i.e., number of action steps generated per second). This demonstrates that despite leveraging large-scale pretraining, VITA remains efficient for deployment on a single, commercially available computing device, without requiring extensive computational resources.

11. Discussion

11.1. Limitations

Although we have unified visual perception and action generation, our current framework does not fully integrate the visual chain-of-thought (V-CoT) with the textual chain-of-thought (T-CoT). Consequently, in complex, multi-step tasks that require deep textual reasoning and instruction grounding, such as "making a sandwich". Our method may struggle to fully capture the user's high-level intent. Nevertheless, VITA achieves remarkable advances in spatial scene modeling, motion coherence, and inference efficiency, demonstrating strong performance across a wide range of robotic manipulation benchmarks.

11.2. Future Works

A promising direction for future work lies in redesigning the discrete tokenization of action sequences to more faithfully preserve fine-grained temporal dynamics. Currently, VITA encodes action trajectories by first applying the Discrete Cosine Transform (DCT) to compress them into the frequency domain, followed by quantization via a shared codebook. While this strategy effectively reduces redundancy, the fixed basis functions of DCT are ill-suited for capturing non-stationary or aperiodic motion patterns commonly found in real-world tasks, such as abrupt stops, multi-phase maneuvers, or reactive corrections. More critically, this approach treats the entire action window as a monolithic unit, sacrificing precise time-step alignment between visual context and motor output. This limitation can be particularly detrimental in real-robot settings that demand millisecond-level timing precision for safe and robust execution.

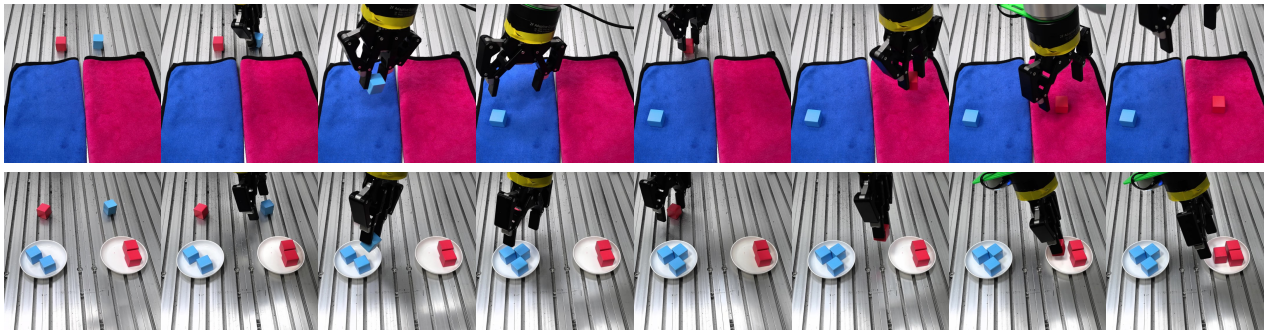

Figure 9. VITA executes the "Color Matching" task. Following the instruction (e.g., "Place red and blue blocks on towels (or plates) of matching colors"), VITA controls the gripper to place the blue block on the blue towel (plate) and the red block on the red towel (plate).

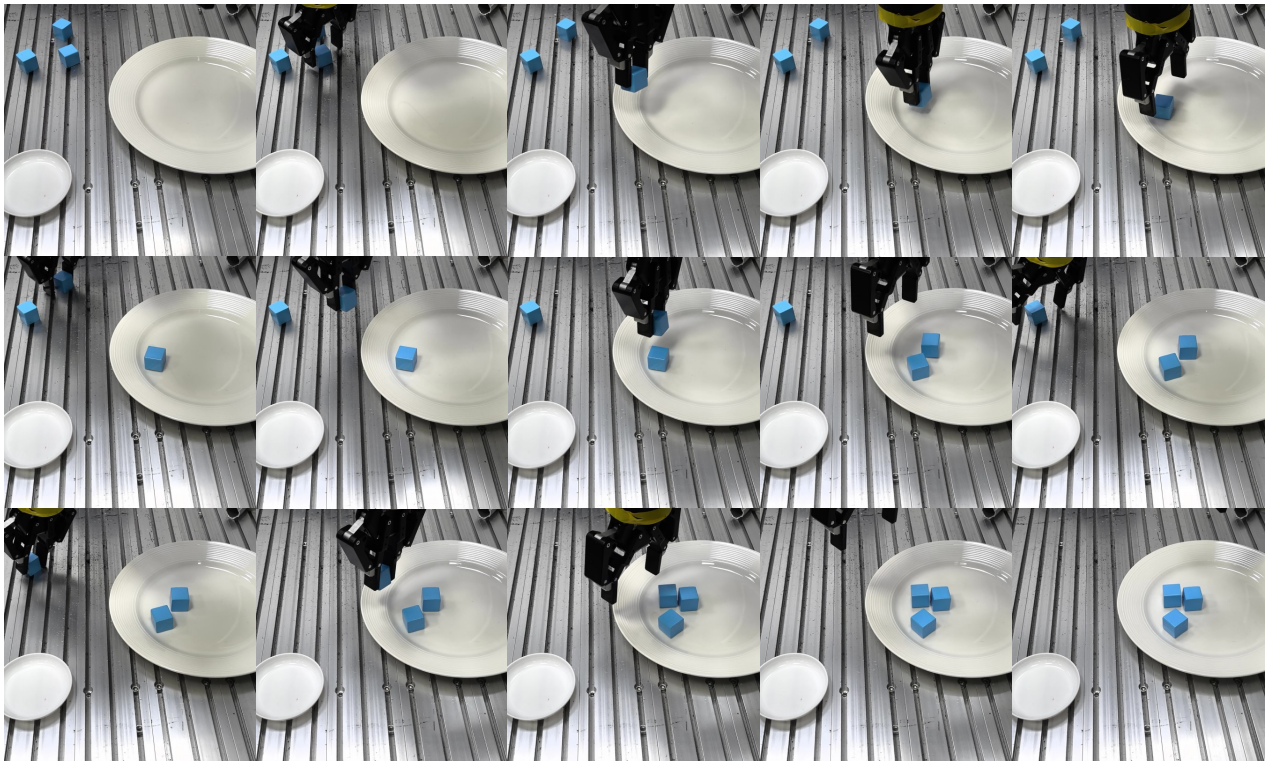

Figure 10. VITA executes the "Multi-step Transfer" task. Following the instruction (e.g., "Move all blue blocks from the table to the large plate"), VITA first picks up each blue block scattered on the table and places them into the large plate.

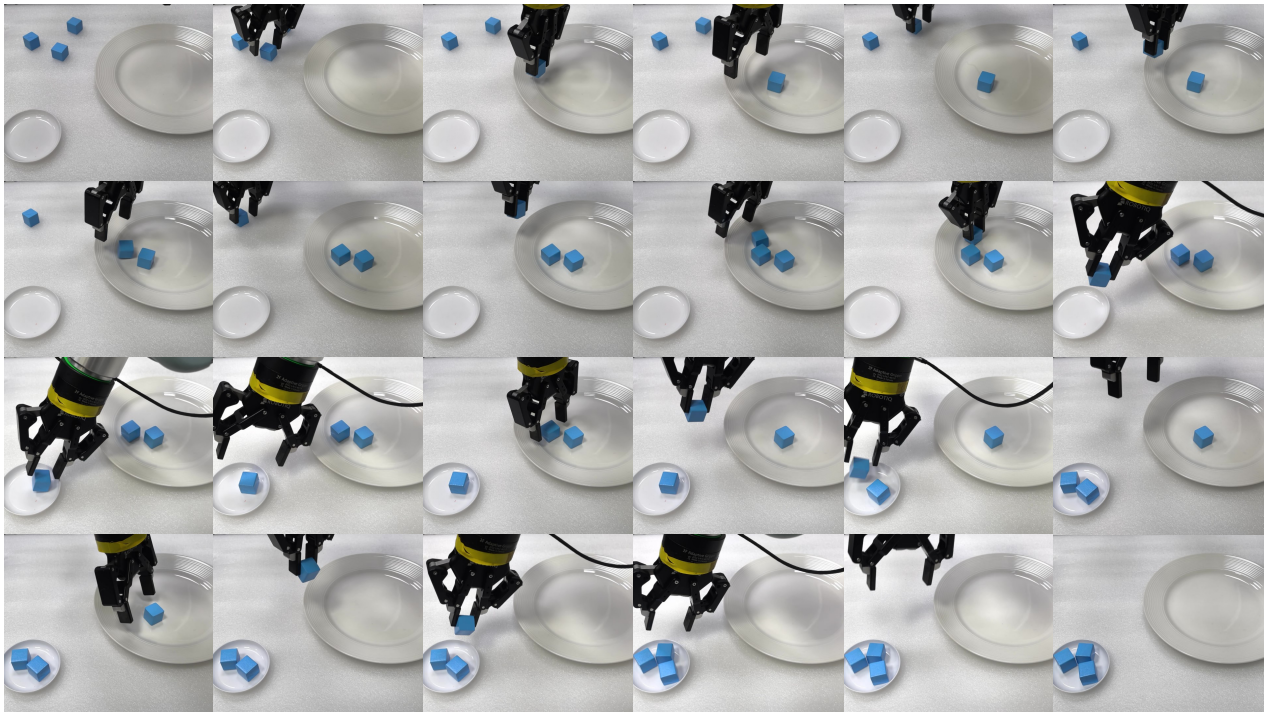

Figure 11. VITA executes the “Long-horizon” task. Following the instruction (e.g., “Move all blue blocks from the large plate to the small plate”), VITA first picks up all blue blocks scattered on the table and places them into the large plate, then retrieves them and neatly stacks them into the small plate.

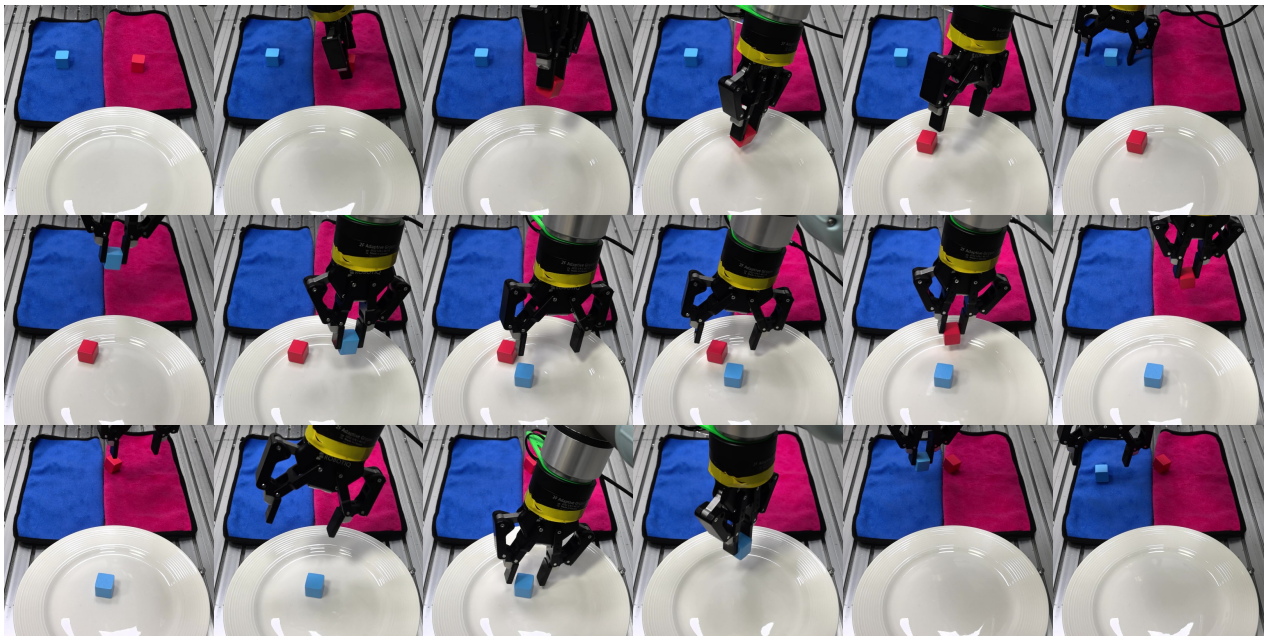

Figure 12. VITA executes the “Multi-step Transfer and Restore” task. Following the instruction (e.g., “Move all blocks from the towel to the plate, then restore them to their original positions”), this task evaluates the model’s long-horizon planning and contextual memory. It requires the model not only to perform simple pick-and-place actions but also to understand and execute a two-stage composite instruction: first moving objects from their initial position (towel) to an intermediate location (plate), then using memory to return them precisely to their original positions.

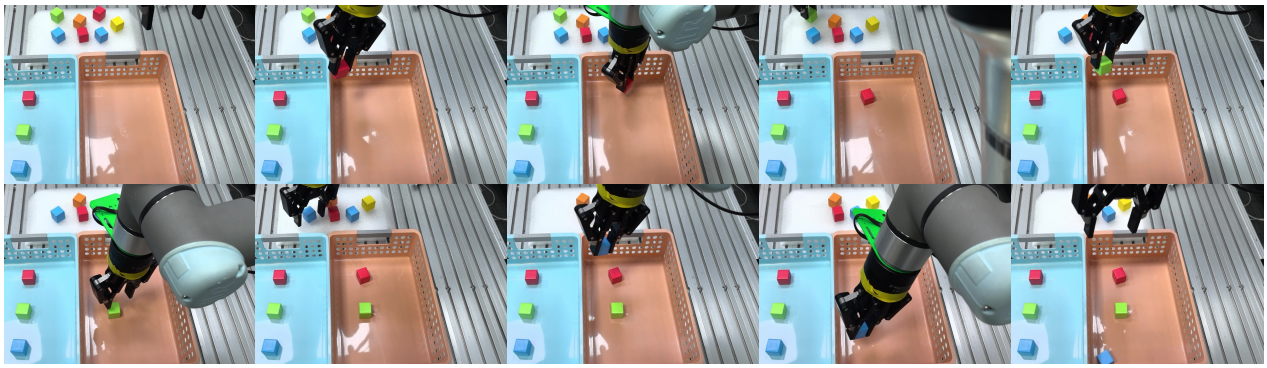

Figure 13. VITA executes the "Imitate Arrangement Strategy" task. Following the instruction (e.g., "Arrange scattered blocks in the red basket following the same pattern"), VITA replicates the observed spatial sequence.

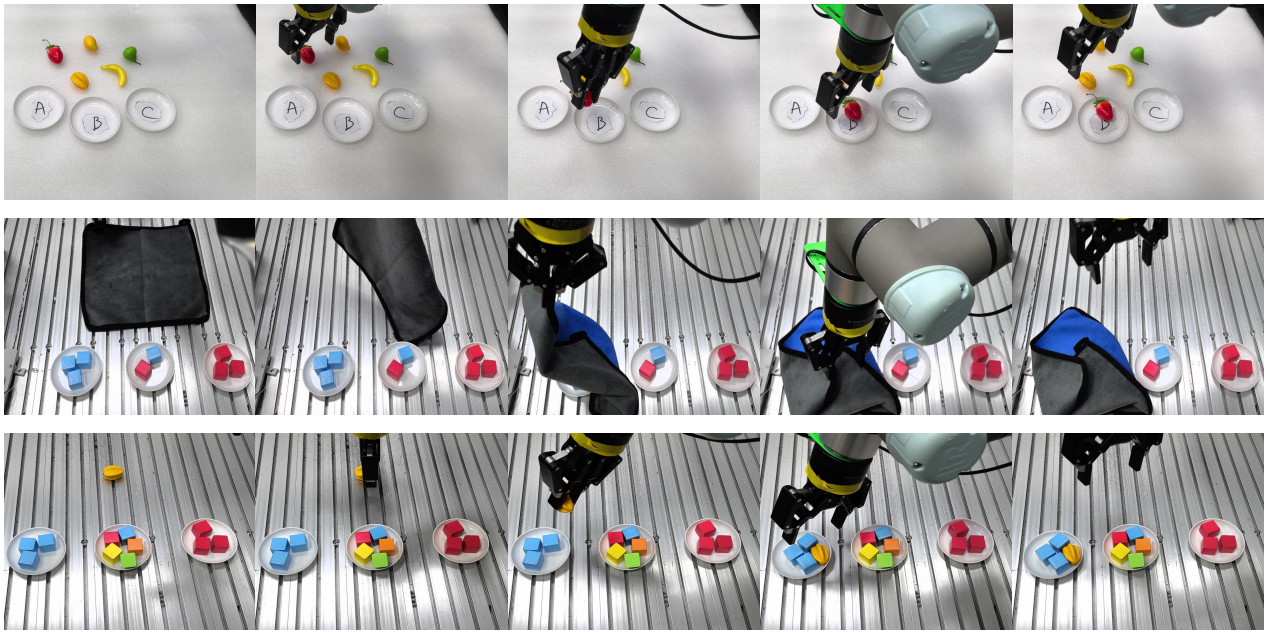

Figure 14. The figure demonstrates the model's performance on three "visual reasoning" tasks. For example, in the top sub-figure, we instruct the model to place a fruit into the plate labeled with the letter "B"; in the middle sub-figure, the instruction is to cover the plate containing the most blue blocks with a towel; and in the bottom sub-figure, we require the model to place the fruit onto the plate with the highest number of blue blocks.

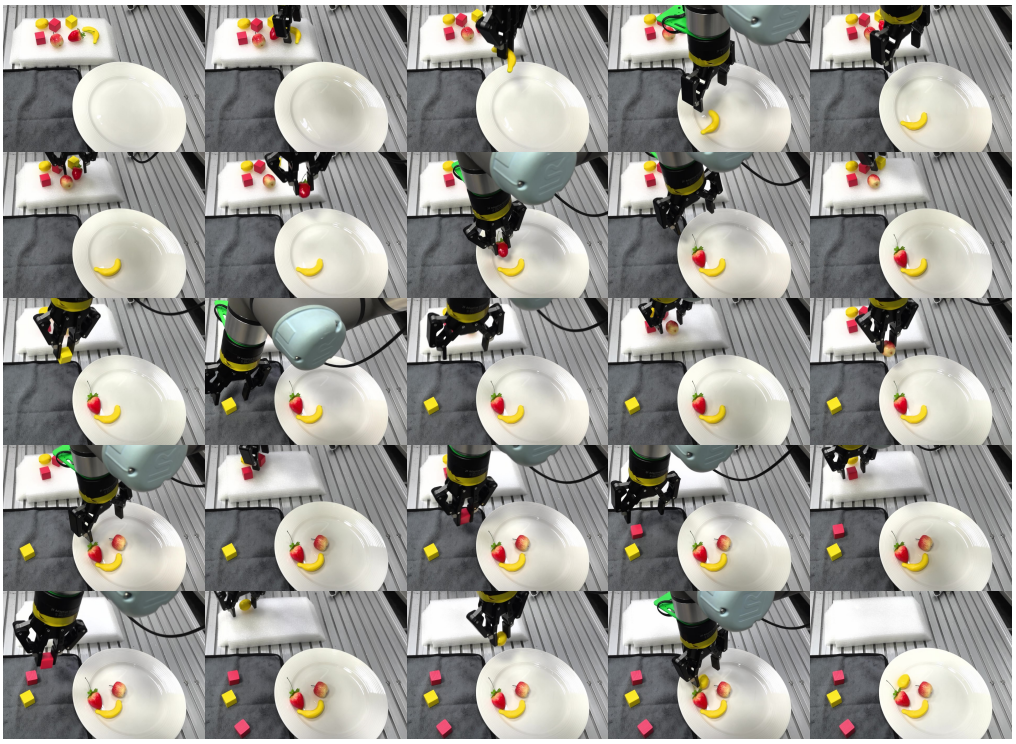

Figure 15. In this long-horizon task, we require the model to classify objects by their type—for example, placing blocks on the towel and fruits in the plate.

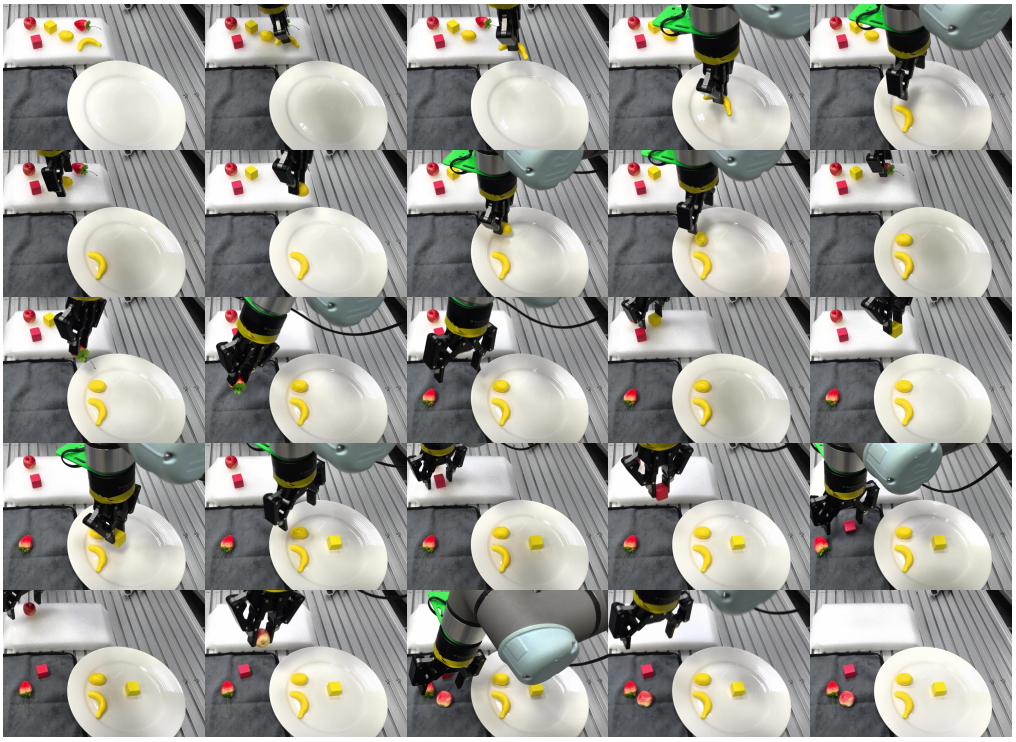

Figure 16. In this long-horizon task, we require the model to classify objects by their color—for example, placing red objects on the towel and yellow objects in the plate.

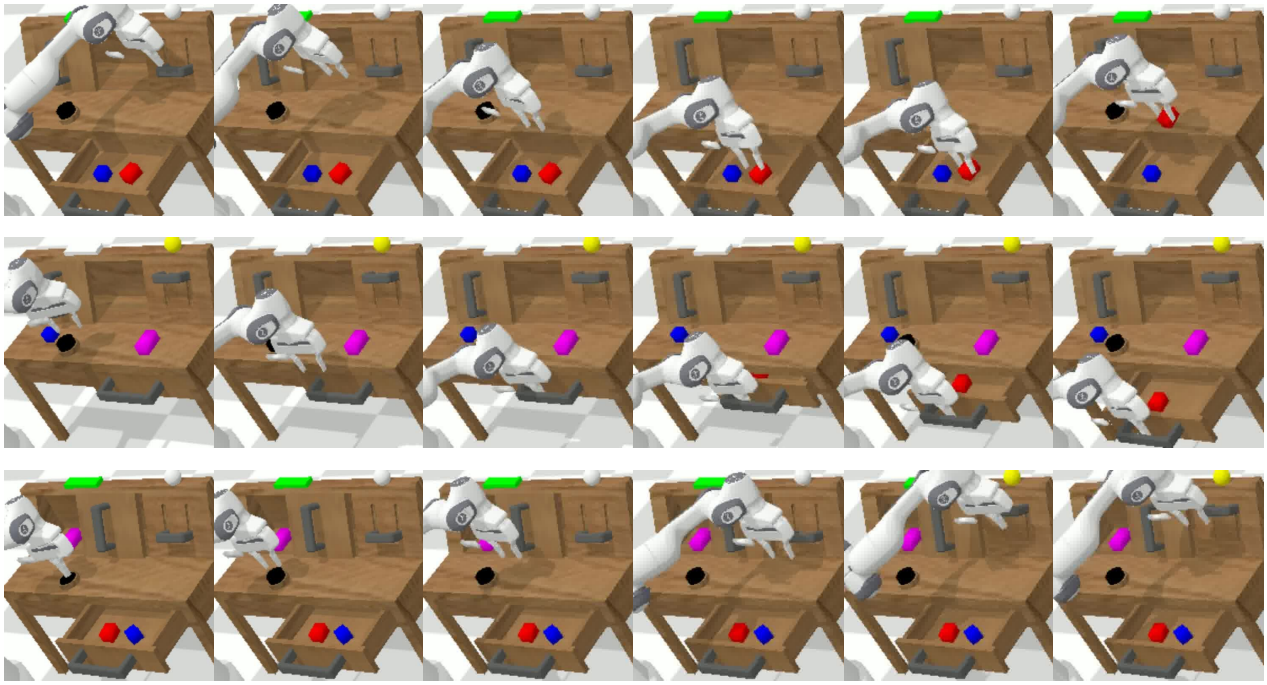

Figure 17. Visualization of action trajectory generation by VITA in the CALVIN simulation environment.

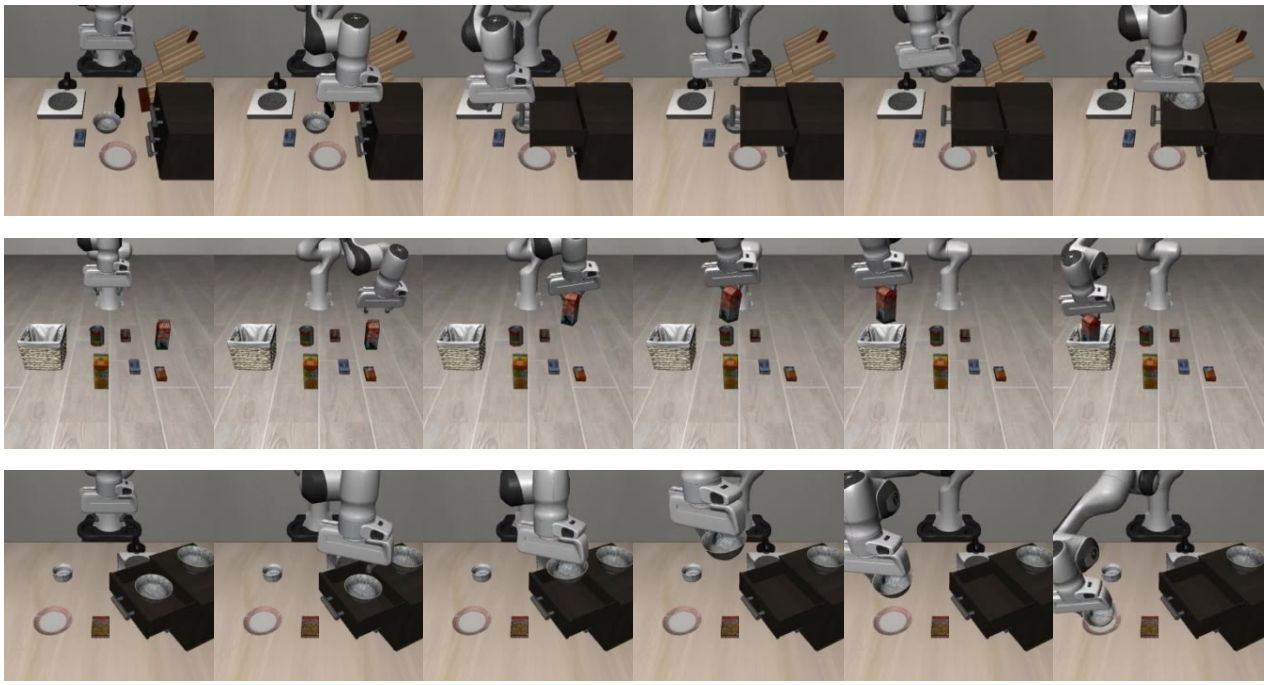

Figure 18. Visualization of action trajectory generation by VITA in the LIBERO simulation environment.

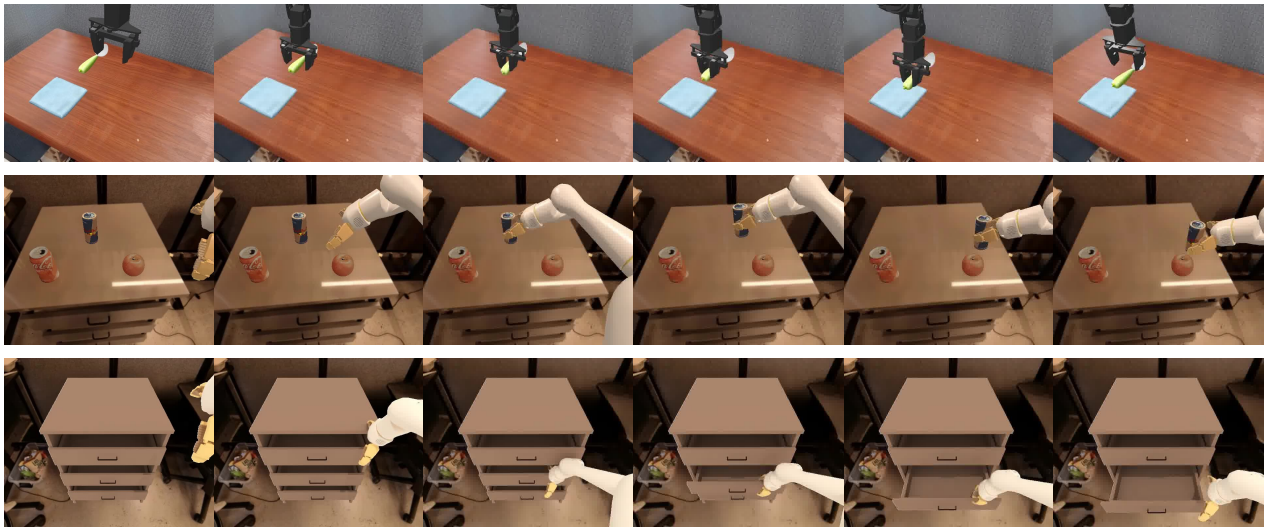

Figure 19. Visualization of action trajectory generation by VITA in the SimplerEnv Google Robot and WidowX simulation environment.

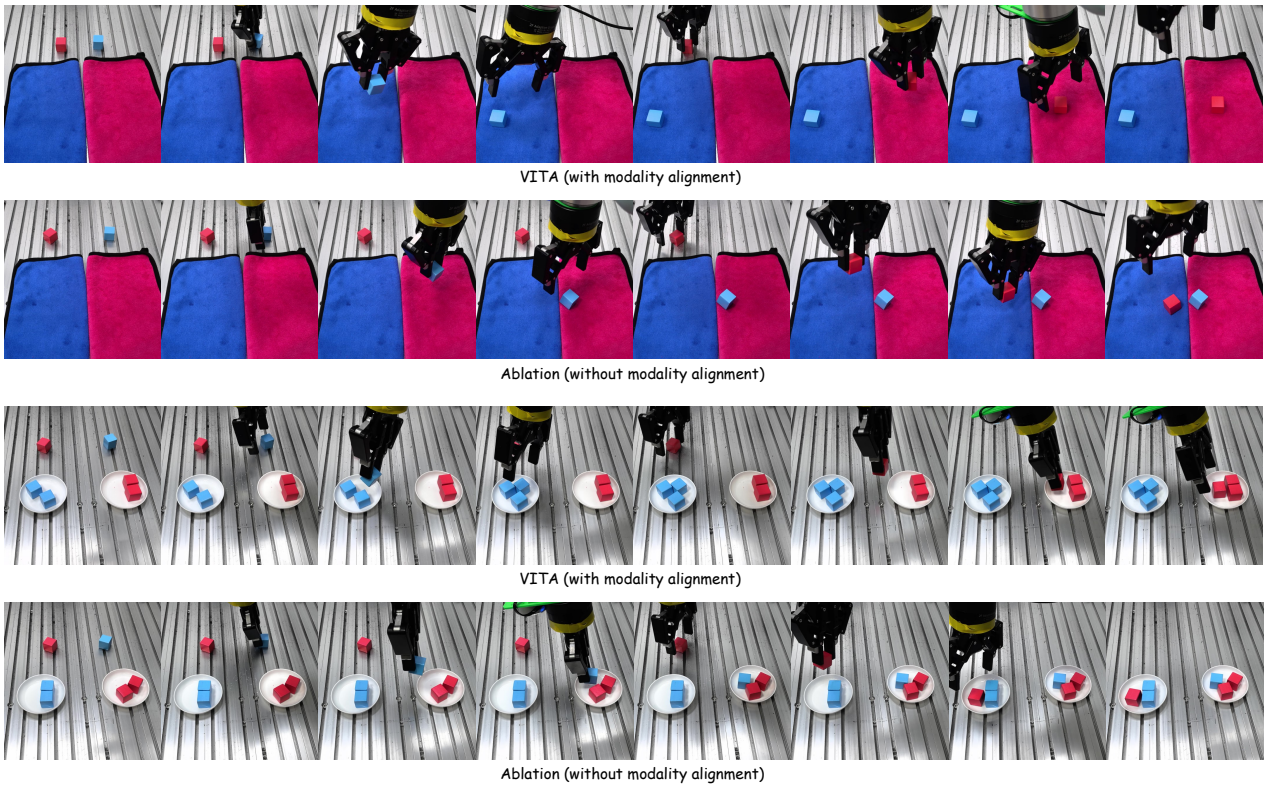

Figure 20. We visualize the performance of the model and its ablated variants in real-world “Color Matching” tasks.