# SOLIS: STREAMING UNIFIED WORLD MODELS FOR ROBOTIC INTERACTION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Robotic interaction couples actions, changes in the environment, and progress toward a task. Modeling these modalities jointly offers a foundation for both generating actions toward desired outcomes and anticipating the consequences of actions. We present **Solis**, a streaming unified world model that learns a joint distribution over observations, actions, and progress-derived values using a single video diffusion transformer. Using different inputs as conditions, the same model can be used for simulation, planning, or evaluation. For practical deployment, closed-loop feedback is important for adapting to changes in the environment, yet restarting diffusion sampling after every observation makes frequent feedback costly. To address this, we employ temporal block-causal attention and natively pretrain our model with independently sampled chunk noise levels. At inference, we adopt a rolling diffusion schedule to maintain future video, action, and value chunks at progressively increasing noise levels: the leading chunk becomes executable while later chunks remain partially denoised. After execution, the horizon shifts with a new observation and a fresh noise chunk enters. This design raises feedback frequency without extra computational cost. With cross-embodiment pretraining and per-benchmark post-training, Solis achieves 98.4% average closed-loop success across four LIBERO suites and 88.8% and 89.1% across 12 RoboTwin 2.0 tasks in clean and randomized scenes. Furthermore, Solis shows strong future prediction and accurate value estimation for task evaluation.

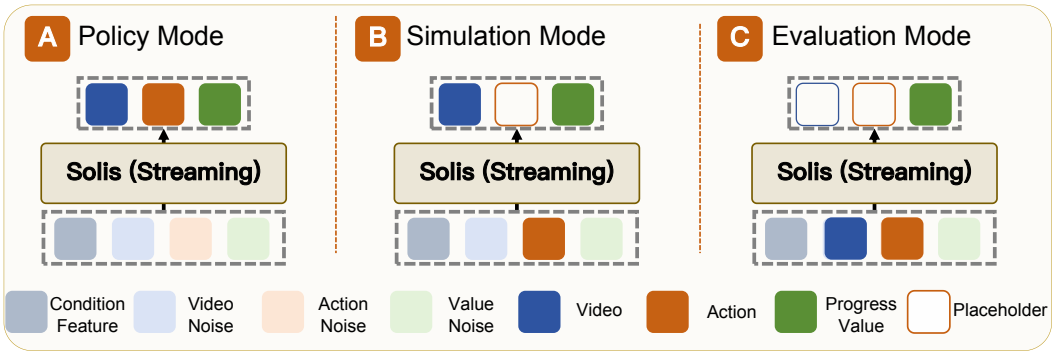



Figure 1: One trajectory and three conditional tasks. Observed video, state, and instruction condition a shared video transformer. Each future chunk contains frame, action, and value slots. Supplying actions selects world-model mode. Supplying a target value selects policy mode. Supplying both actions and visual observations selects evaluator mode.

## 1 INTRODUCTION

Recent advances in robot learning have enabled robots to follow natural-language instructions and perform diverse manipulation tasks across objects and environments (Zitkovich et al., 2023; Kim et al., 2024; Ye et al., 2026b). Vision-language-action (VLA) models transfer semantic knowledge from pretrained vision-language models to robot control, mapping observations and language instructions to executable actions (Zitkovich et al., 2023; Kim et al., 2024). World-action models



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

(WAMs) further couple action generation with predictions of future observations, letting robots use anticipated scene evolution to generate and evaluate actions (Ye et al., 2026b; Kim et al., 2026).

Most existing world-action models connect action generation with future visual prediction, while value estimation provides a way to assess the predicted outcomes (Ye et al., 2026b; Kim et al., 2026). These quantities describe complementary aspects of the same future interaction, motivating a unified distribution over observations, actions, and values. Such a formulation expresses simulation, action generation, and evaluation as different conditional queries to a shared model. Maintaining this unified prediction process during closed-loop control requires the joint trajectory to remain revisable as new observations arrive. Restarting diffusion at every control query regenerates the trajectory from noise and makes frequent feedback computationally expensive. This motivates a unified model that retains and progressively refines a joint future trajectory across control cycles, producing executable actions while updating their predicted visual consequences and values.

We present **Solis**, a streaming unified world model for continuous closed-loop robot control. Solis treats observations, actions, and progress-derived values as different modalities of a shared future trajectory and learns their joint distribution using a single video diffusion transformer, supporting simulation, planning, and evaluation within the same model (Figure 1). For example, conditioning on actions lets the model predict visual outcomes and their values, while conditioning on target values lets it generate actions and visual futures. To support continuous refinement of this joint trajectory, we employ temporal block-causal attention, allowing video, action, and value tokens to interact within each chunk while restricting temporal attention to the current and preceding chunks. During robot-data pretraining, we adopt diffusion forcing to independently sample noise levels across chunks and share each level among the predicted modalities within a chunk. This prepares the model to jointly refine future predictions at different stages of denoising. At inference, we adopt a rolling diffusion scheme to maintain future chunks at progressively increasing noise levels, so the leading chunk reaches zero noise and supplies executable actions while later chunks remain partially denoised. After execution, the model removes the leading chunk, shifts the retained chunks forward with their latent states and noise levels intact, and appends a new chunk initialized with pure noise. New camera and proprioceptive observations refresh the conditioning context and guide the remaining denoising of the retained video, action, and value predictions. This process refines the same joint future under successive observations and spreads its denoising across control cycles.

We pretrain Solis across embodiments on DROID, RoboMIND, and AgiBot World (Khazatsky et al., 2024; Wu et al., 2024b; AgiBot-World-Contributors et al., 2025), followed by post-training on each benchmark. We evaluate closed-loop control on the four LIBERO suites (Liu et al., 2023) and 12 bimanual manipulation tasks from RoboTwin 2.0 (Chen et al., 2025). Using joint generation and rolling diffusion, Solis achieves an average success rate of 98.4% on LIBERO and 88.8% and 89.1% on RoboTwin 2.0 under clean and randomized scene settings, respectively. Solis also shows strong future prediction abilities and estimates accurate values for task evaluation.

## 2 Related Work

**Video Models for Robot Control.** Pretrained video generators provide priors for how scenes change under interaction (Team Wan et al., 2025; Agarwal et al., 2025). One line of work infers actions from generated future frames (Du et al., 2023). Other methods adapt video backbones for action prediction (Wu et al., 2024a) or train action heads on video features (Hu et al., 2025; Pai et al., 2025). Action-conditioned video models predict the consequences of a proposed control sequence (Bruce et al., 2024; Agarwal et al., 2025). Solis builds on video priors to jointly model observations, actions, and progress-derived values, unifying simulation, action generation, and evaluation.

**Joint Video and Action Generation.** Joint generative models capture the relationship between robot actions and visual dynamics, supporting forward prediction, inverse dynamics, and policy learning through different conditioning patterns (Zhu et al., 2025; Li et al., 2025b). Existing architectures integrate the two modalities through shared latent representations or coupled prediction modules, using pretrained video models to support action generation (Kim et al., 2026; Ye et al., 2026b; Yuan et al., 2026). Value prediction further extends this formulation to evaluating imagined outcomes and selecting actions through model-based planning (Kim et al., 2026). Solis couples a unified conditional model of observations, actions, and progress-derived values with rolling inference, retaining partially denoised joint trajectories for continued refinement across control queries.

**Streaming Diffusion.** Streaming diffusion combines causal temporal modeling with incremental sampling to generate sequences beyond a fixed prediction horizon (Chen et al., 2024; Yin et al., 2025b; Huang et al., 2025). Independent noise levels across temporal chunks allow a single model to refine predictions at different stages of denoising, while few-step generation and cached temporal context reduce inference computation (Chen et al., 2024; Yin et al., 2025b; Huang et al., 2025). Rolling schedules jointly denoise future video chunks at progressively increasing noise levels, emit the leading chunk, and append new noise as the window advances (Liu et al., 2025). Related robotic policies retain partially denoised action buffers and refine future actions under new observations, amortizing sampling across feedback updates (Høeg et al., 2024). Causal attention across buffered action chunks further supports incremental decoding and continuous execution in vision-language-action models (Li et al., 2026b). Solis extends this rolling formulation to joint prediction of observations, actions, and progress-derived values within a single video diffusion transformer. Each feedback update refines all three modalities in the retained trajectory, so future actions, their predicted visual consequences, and associated values are updated together as control proceeds.

# 3 METHOD

## 3.1 UNIFIED WORLD MODELING

Given a language instruction $l$, the robot aims to generate actions that complete the specified task using the information observed so far. At control step $t$, the robot receives a visual observation $\mathbf{o}_t$ and a proprioceptive state $\mathbf{q}_t$. We denote the available context by $\mathbf{c}_t = (\mathbf{o}_{t-P+1:t}, \mathbf{q}_t, l)$, where $P$ is the observation history length and additional camera histories are included when available. For a prediction horizon of $N$ steps, a policy models the future action sequence $\mathbf{a} = \mathbf{a}_{t:t+N-1}$ through

$$p_\theta(\mathbf{a} \mid \mathbf{c}_t). \tag{1}$$

World-action models augment action prediction with the future observations $\mathbf{o} = \mathbf{o}_{t+1:t+N}$ produced by those actions, relating executable behavior to its visual consequences (Zhu et al., 2025; Li et al., 2025b; Ye et al., 2026b). This yields a joint model of actions and future observations,

$$p_\theta(\mathbf{o},\, \mathbf{a} \mid \mathbf{c}_t). \tag{2}$$

A general embodied agent should perceive its environment, predict how it will change, act toward a goal, and evaluate progress toward that goal. The observed context supports perception, while future observations, actions, and values describe the outcomes, behavior, and progress of the same interaction. This motivates Solis to model these future quantities jointly, with a value trajectory $\mathbf{V} = V_{t+1:t+N}$ accompanying the visual and action predictions. The value supervision is derived from demonstration progress. Solis learns the unified distribution

$$p_\theta(\mathbf{o},\, \mathbf{a},\, \mathbf{V} \mid \mathbf{c}_t). \tag{3}$$

During training, randomized conditioning masks expose the same model to both joint generation and conditional prediction of the component modalities. At inference, supplied future variables are held fixed and the remaining variables are sampled from the conditional distribution, without retraining or introducing separate prediction networks. This supports three conditional modes.

**World-Model Mode.** Given a candidate action sequence and the observed context, the model predicts future observations and their associated values through

$$p_\theta(\mathbf{o},\, \mathbf{V} \mid \mathbf{a},\, \mathbf{c}_t). \tag{4}$$

The visual predictions simulate the consequences of the supplied actions, and the value predictions estimate the corresponding task progress, supporting evaluation before execution.

**Policy Mode.** Given a target value trajectory that specifies the desired evolution of task progress, the model generates actions and future observations through

$$p_\theta(\mathbf{o},\, \mathbf{a} \mid \mathbf{V},\, \mathbf{c}_t). \tag{5}$$

This mode enables value-conditioned planning, with visual predictions for the generated actions.

**Evaluator Mode.** Given an action sequence and its corresponding visual observations, the model predicts the value trajectory through

$$p_\theta(\mathbf{V} \mid \mathbf{o},\, \mathbf{a},\, \mathbf{c}_t). \tag{6}$$

This mode evaluates task progress along a fixed visual-action trajectory.

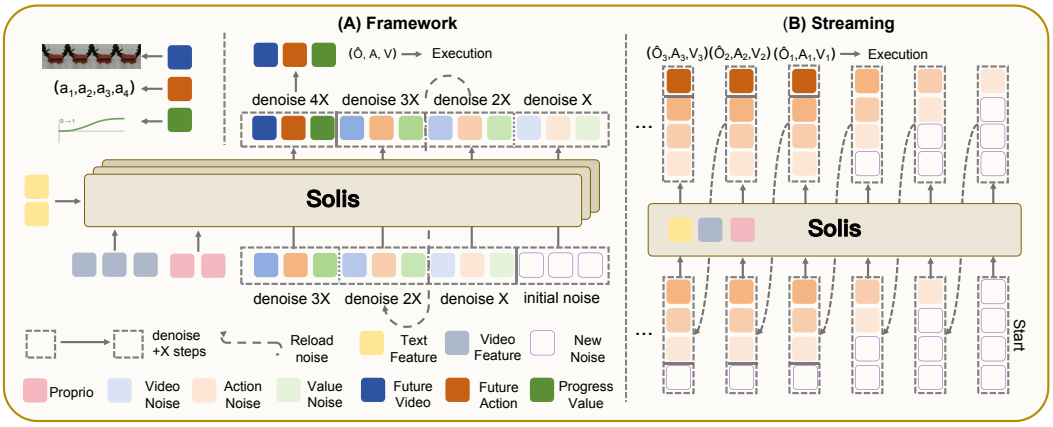



Figure 2: Rolling horizon across two control queries. After the clean leading chunk supplies actions, the remaining chunks persist at progressively increasing noise levels. The next observation refreshes the conditioning context. The retained chunks advance, and a new noisy chunk enters at the back.

### 3.2 Temporal Causal Multimodal Diffusion

To expose observations, actions, and values to the same backbone, we represent them as frame-shaped latent slots and jointly denoise them with a video diffusion transformer. Temporal block-causal attention couples modalities within each chunk while keeping chunks causally ordered.

**Multimodal Trajectory Representation.** We divide the future into $H$ chunks of $n$ frames, so $N = nH$. Chunk $c \in \{1, \dots, H\}$ gathers its frames $\mathbf{o}_c = \mathbf{o}_{t+n(c-1)+1:t+nc}$, the actions that produce them, $\mathbf{a}_c = \mathbf{a}_{t+n(c-1):t+nc-1}$, and their value labels $\mathbf{V}_c = V_{t+n(c-1)+1:t+nc}$. A causal video autoencoder maps each future frame chunk to a visual latent $\mathbf{z}_c^o$ of shape $C \times h \times w$. Actions and values are tiled into the same latent shape with an operator $\mathcal{T}$, and a read-out operator $\mathcal{R}$ recovers each vector by averaging its repeated copies. Each chunk therefore contains a visual slot, an action slot, and a value slot. The sequence places the conditioning slots first and then the future chunks:

$$\mathbf{S} = \big(\mathbf{S}_{\text{cond}},\ \underbrace{\mathbf{z}_1^o,\ \mathcal{T}(\mathbf{a}_1),\ \mathcal{T}(\mathbf{V}_1)}_{\text{chunk } 1},\ \dots,\ \underbrace{\mathbf{z}_H^o,\ \mathcal{T}(\mathbf{a}_H),\ \mathcal{T}(\mathbf{V}_H)}_{\text{chunk } H}\big). \tag{7}$$

Here, $\mathbf{S}_{\text{cond}}$ contains the encoded visual history and tiled proprioception, with an additional observed-camera slot when available. The backbone patchifies each slot into visual tokens and processes them with its existing attention and output head, with cross-attention to the instruction embeddings. Slots occupy consecutive temporal positions in the rotary embedding, preserving the ordering of modalities within each chunk and of chunks across time.

**Temporal Block-Causal Attention.** We apply a token-level block-causal mask from robot-data pretraining onward. Let $\gamma(i) = 0$ for a conditioning slot and $\gamma(i) = c$ for a slot in future chunk $c$. A token of slot $i$ attends to a token of slot $i'$ if and only if $\gamma(i) \geq \gamma(i')$. The conditioning slots form one bidirectional block that attends only to itself. Each future chunk attends to the conditioning block, all preceding chunks, and all three modalities within its own chunk. Within-chunk attention couples the visual, action, and value predictions, while temporal causality prevents later, noisier chunks from affecting earlier predictions. This dependency structure supports appending a new noisy chunk without changing the context available to preceding chunks.

Each slot $i$ also receives a noise level $\sigma_i \in [0, 1]$, where $i = 1, \dots, M$ and $M$ is the total number of slots. The corresponding timestep $\tau_i = \tau_{\max}\sigma_i$ enters the backbone through its timestep embedding and adaptive normalization. The same transformer thus handles chunks at different denoising stages.

### 3.3 Streaming World Modeling with Closed-Loop Feedback

To reuse joint predictions across control cycles, the model must refine partially denoised future chunks under updated observations. We train with diffusion forcing and use rolling diffusion to retain these chunks as the robot acts and receives feedback.

**Diffusion Forcing Training.** Following diffusion forcing (Chen et al., 2024), we independently sample a noise level $\sigma_c$ for each future chunk and share it among the visual, action, and value slots

to be predicted in that chunk. Together with temporal block-causal attention, this training procedure exposes the model to the unequal chunk noise levels used during rolling inference. Both designs are used from robot-data pretraining onward. Following flow matching (Lipman et al., 2023), the noisy version of a target slot $i$ and its velocity target are

$$\mathbf{z}_i(\sigma_i) = (1-\sigma_i)\mathbf{z}_i + \sigma_i \boldsymbol{\epsilon}_i, \qquad \mathbf{u}_i = \boldsymbol{\epsilon}_i - \mathbf{z}_i, \qquad \boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{8}$$

The network $v_\theta$ maps the noisy sequence $\mathbf{S}(\boldsymbol{\sigma})$, timesteps $\boldsymbol{\tau}$, and instruction $l$ to per-slot velocities.

A conditioning mask selects which future modalities are supplied. Given slots remain clean with $\tau_i = 0$ and contribute no loss. World-model mode fixes the action slots, policy mode fixes the value slots, and evaluator mode fixes both the visual and action slots. Joint generation leaves all future modalities as prediction targets. Observed visual history and proprioception are always conditioning inputs and receive noise augmentation during training.

Let $\mathcal{I}_o$, $\mathcal{I}_a$, and $\mathcal{I}_V$ denote the predicted visual, action, and value slots, excluding padded chunks beyond the demonstration. The training objective averages the squared velocity error over the predicted slots of each modality:

$$\begin{aligned} \mathcal{L}_* &= \frac{1}{|\mathcal{I}_*|} \sum_{i \in \mathcal{I}_*} \frac{1}{Chw} \left\| v_\theta(\mathbf{S}(\boldsymbol{\sigma}), \boldsymbol{\tau}, l)_i - \mathbf{u}_i \right\|_2^2, \\ \mathcal{L} &= \lambda_a \mathcal{L}_a + \lambda_o \mathcal{L}_o + \lambda_V \mathcal{L}_V, \end{aligned} \tag{9}$$

where $* \in \{o, a, V\}$ and $\lambda_a$, $\lambda_o$, and $\lambda_V$ weight the modality losses. A modality with no prediction targets contributes zero loss, so evaluator mode uses only the value loss. The noise distribution emphasizes high noise levels, where actions and values must be inferred from context.

**Rolling Diffusion Inference.** At inference, rolling diffusion maintains future chunks at progressively increasing noise levels. The leading chunk reaches a clean state and supplies executable actions while later chunks remain partially denoised (Figure 2). The schedule follows pyramid sampling in diffusion forcing (Chen et al., 2024) and its streaming use in Rolling Forcing (Liu et al., 2025), applied here to the joint visual, action, and value trajectory.

For a decreasing noise schedule $\sigma^{(1)} = 1, \ldots, \sigma^{(K+1)} = 0$, we offset the denoising schedules of the $H$ future chunks by $L = K/H$ steps, with $K$ a multiple of $H$. Each predicted slot $i$ retains its latent and schedule position $j_i$. A network evaluation updates every active slot at its own noise level:

$$\mathbf{z}_i \leftarrow \mathbf{z}_i + \left(\sigma^{(j_i+1)} - \sigma^{(j_i)}\right) v_\theta(\mathbf{S}(\boldsymbol{\sigma}), \boldsymbol{\tau}, l)_i, \qquad j_i \leftarrow j_i + 1. \tag{10}$$

Clean slots and supplied conditioning variables remain fixed. At the first query, all chunks begin as pure noise, and their denoising starts at successively delayed steps to establish the rolling schedule. After $K$ network evaluations, the leading chunk is clean and the rest are partially denoised.

**Closed-Loop Feedback.** After executing the leading action chunk, the sampler removes it, advances the retained chunks with their latents and schedule positions intact, and appends a new chunk initialized with pure noise. New visual observations and proprioception replace the conditioning slots and condition the remaining denoising steps of the retained trajectory. Generated frames remain future predictions and are never fed back as observed history. The sampler then performs $L$ network evaluations to complete the next leading chunk and repeats this process throughout the episode. Retained chunks are not re-noised, and the horizon resets only at episode boundaries.

After initialization, each executed chunk requires $K/H$ network evaluations, compared with $K$ when restarting the full denoising schedule at every query. In steady operation, each chunk is refined under $H$ successive observations before execution. This reduces the number of network evaluations between feedback updates while preserving intermediate predictions across control cycles. The counts describe network evaluations rather than measured wall-clock latency.

Supplied future modalities remain fixed according to the conditional mode in Section 3.1. For the reported closed-loop experiments, no future modality is supplied, so the sampler jointly denoises visual, action, and value slots and reads out the leading actions for execution. Standalone world-model and evaluator use fixes the supplied variables and predicts the rest without an execution loop.

Table 1: Closed-loop success rate (%) on the four LIBERO suites; Long denotes LIBERO-10. The Source column identifies the paper reporting each baseline. "–" denotes an unreported value. The middle block contains video-generation-based methods. Solis uses joint mode and 500 episodes per suite, with $K = 40$ and $K = 4$ denoising steps, 10 and one network evaluations per executed chunk.

| Method | Source | Spatial | Object | Goal | Long | Avg. |
|---|---|---|---|---|---|---|
| $\pi_0$ | Akbari et al. (2026) | 96.8 | 98.8 | 95.8 | 85.2 | 94.1 |
| $\pi_{0.5}$ | Kim et al. (2026) | 98.8 | 98.2 | 98 | 92.4 | 96.9 |
| OpenVLA-OFT | Kim et al. (2026) | 98.4 | 97.9 | 97.6 | 94.5 | 97.1 |
| X-VLA | Akbari et al. (2026) | 98.2 | 98.6 | 97.8 | 97.6 | 98.1 |
| UVA | Li et al. (2025b) | – | – | – | 90.0 | – |
| mimic-video | Pai et al. (2025) | 94.2 | 96.8 | 90.6 | – | – |
| Flash-WAM (1v/2a) | Akbari et al. (2026) | 97.0 | 92.8 | 96.4 | 98.0 | 95.7 |
| LingBot-VA | Akbari et al. (2026) | 98.5 | 99.8 | 98.0 | **98.3** | **98.6** |
| Fast-WAM | Yuan et al. (2026) | 98.2 | **100.0** | 97.0 | 95.2 | 97.6 |
| Fast-WAM-Joint | Yuan et al. (2026) | **99.6** | 99.4 | 98.2 | 96.8 | 98.5 |
| Cosmos Policy | Kim et al. (2026) | 98.1 | **100.0** | 98.2 | 97.6 | 98.5 |
| **Solis**, $K = 40$ | | 98.2 | 99.2 | **98.6** | 97.6 | 98.4 |
| **Solis**, $K = 4$ | | 98.4 | 99.0 | 97.8 | 98.0 | 98.3 |

# 4 Experiments

## 4.1 Experimental setup

**Pretraining Data.** We pretrain on DROID (Khazatsky et al., 2024), RoboMIND (Wu et al., 2024b), and AgiBot World (AgiBot-World-Contributors et al., 2025) with weights 0.30, 0.30, and 0.40. DROID's two exterior views are separate sources at 0.15 each. An epoch draws 1,000,000 samples.

**Benchmarks.** LIBERO (Liu et al., 2023) has four suites of 10 tasks each; we train on its 1,877 demonstrations, with a head and a wrist camera at $224 \times 224$. RoboTwin 2.0 (Chen et al., 2025) is a 50-task bimanual benchmark. We use the 12-task subset of Li et al. (2025a), which spans four horizon levels and has since been used in full or in part by Yin et al. (2025a); Wang et al. (2026); Zang et al. (2025); Wu et al. (2026); Chen et al. (2026) (Table 7), and train on the benchmark's 50 clean and 500 randomized demonstrations per task as in Bi et al. (2025); Ye et al. (2026a); Peng et al. (2026), with a head and two wrist cameras at $192 \times 256$. Actions on both benchmarks are absolute end-effector targets in the shared 20-D layout.

**Implementation Details.** We build on Wan2.2-TI2V-5B (Team Wan et al., 2025). Its causal video autoencoder encodes the first frame of a clip separately and each subsequent group of four frames as one latent with $C = 48$ channels and $16\times$ spatial downsampling. For a $224 \times 224$ input, the latent spatial dimensions are $h \times w = 14 \times 14$. The diffusion transformer has 30 blocks, a hidden size of 3072, and 24 attention heads, with cross-attention to T5 instruction embeddings. More implementation details are provided in Appendix A.

## 4.2 Closed-loop control on LIBERO

Table 1 reports four LIBERO suites with 500 episodes per suite. Baseline values come from the cited sources rather than a common re-evaluation, and their evaluation protocols differ. Flash-WAM reports an average of 95.7%, although its displayed suite rates average to 96.1%; we retain its reported value. Solis obtains 98.2%, 99.2%, 98.6%, and 97.6% success on Spatial, Object, Goal, and LIBERO-10, respectively, for a 98.4% average, within the 98.1%–98.6% range of the strongest listed baselines. With $K = 4$ (one evaluation per chunk instead of 10), the average is 98.3%.

## 4.3 Bimanual control on RoboTwin 2.0

RoboTwin 2.0 tests transfer to bimanual control, with two arms, three cameras, and 16-D actions in the shared layout, with the same architecture and pretrained weights as LIBERO. Table 2 reports success on the 12 tasks. Solis reaches 88.8% (clean) and 89.1% (randomized).

Table 2: Success rate (%) on the 12 RoboTwin 2.0 tasks of Li et al. (2025a), 100 episodes per task and setting. Baselines average the per-task results of the source papers over the same 12 tasks, listed in Appendix B. Feedback: observations per second. Solis: joint mode, $K = 40$.

| Method | Source | Clean | Rand. | Feedback (Hz) |
|---|---|---|---|---|
| $\pi_0$ | Li et al. (2026a) | 71.0 | 64.0 | – |
| $\pi_{0.5}$ | Li et al. (2026a) | 86.6 | 81.0 | 4.4 |
| X-VLA | Bi et al. (2025) | 74.1 | 69.5 | – |
| GigaWorld-Policy | Ye et al. (2026a) | 85.3 | 84.0 | 2.8 |
| Motus | Bi et al. (2025) | 86.4 | 85.3 | 0.8 |
| FACT | Peng et al. (2026) | 87.6 | 87.7 | 2.6 |
| FACT w/ failure co-training | Peng et al. (2026) | 91.5 | 89.8 | 2.6 |
| LingBot-VA | Li et al. (2026a) | 96.4 | **95.3** | – |
| Fast-WAM | Yuan et al. (2026) | **96.8** | 95.2 | – |
| **Solis** (ours) | | 88.8 | 89.1 | **9.1** |

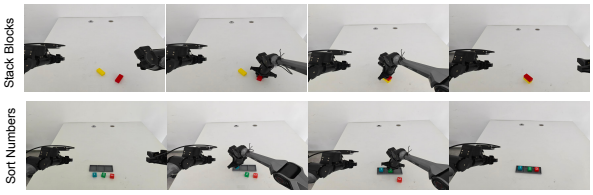



Figure 3: Real-robot stacking (top) and sorting (bottom).

Table 3: Real-robot success rate (%), 30 trials per task with randomized object positions; joint mode, rolling $K = 4$. Details in Appendix C.

| Task | $\pi_0$ | **Solis** |
|---|---|---|
| Stack blocks | 66.7 | **80.0** (24/30) |
| Sort numbers | – | **63.3** (19/30) |

Among the world-action models trained under the same protocol, Solis is level with FACT, above Motus and GigaWorld-Policy, and well above the VLAs. LingBot-VA and Fast-WAM are about 7 points higher. They train one policy on all 50 tasks with 27,500 demonstrations, whereas Solis is trained on the 12 tasks with 6,600, i.e., with four times more bimanual data.

### 4.4 Real-robot experiments

We evaluate two tasks (Figure 3) on an AgileX Cobot Magic bimanual platform (Appendix C): stacking a red block on a yellow block, and sorting numbered blocks, with 145 and 158 teleoperated demonstrations and one model fine-tuned per task. Each task has 30 trials with randomized object positions, and Solis uses joint mode with the rolling $K = 4$ sampler. Solis succeeds in 24 of 30 stacking trials and 19 of 30 sorting trials (Table 3), the latter requiring several pick-and-place cycles. Figure 4 (bottom) shows the frames the rolling sampler generated during one of the stacking trials next to the observed frames: the policy re-observes after every chunk of four actions, and its predicted frames track the observed reach, grasp, and release.

### 4.5 Ablations and Analysis

All comparisons below fix the reported checkpoints and vary only the sampler or conditioning mode.

**Rolling horizon.** Table 4 compares the rolling sampler at $K = 4$ with a non-rolling sampler of the same cost, one network evaluation per executed chunk, on the checkpoint of Table 1. The non-rolling sampler generates all four chunks from noise under the current observation and executes the 16 actions open loop, so every denoising step of a chunk sees one observation, whereas in the rolling horizon a chunk takes its four steps under four successive observations. The rolling sampler is 0.3 points higher on average and 2.0 points higher on LIBERO-10, the suite with the longest tasks.

**Conditioning mode.** Policy mode conditions on a target value trajectory and therefore needs a nominal task length $T$ in equation 11; joint mode generates the value slots with the frames and actions and needs no target. Table 5 compares the two. Policy mode averages 97.4% with $T$ set to the longest demonstration of each task and 96.8% with the median. With the longest $T$ it matches joint mode on Spatial and Goal but loses 2.4 points on LIBERO-10, whose tasks are the longest; with the median $T$, which is shorter than half of the demonstrations, the target reaches $-1$ before the task can be completed, the policy rushes or stalls, and the average drops further. Joint mode reaches 98.3%, above both policy settings without needing $T$, so all other closed-loop results use it.

Table 4: Rolling versus non-rolling sampling on LIBERO with the checkpoint of Table 1, joint mode, $K = 4$, 500 episodes per suite, success rate (%). Both samplers cost one network evaluation per executed chunk of four actions in steady state (Evals; first-chunk cost in parentheses). Non-rolling generates all four chunks from noise at every query and executes all 16 actions open loop.

| Sampler | Evals | Spatial | Object | Goal | Long | Avg. |
|---|---|---|---|---|---|---|
| Rolling, $K = 4$ | 1 (4) | 98.4 | 99.0 | **97.8** | **98.0** | **98.3** |
| Non-rolling, $K = 4$ | 1 (4) | **99.0** | **99.2** | 97.6 | 96.0 | 98.0 |

Table 5: Conditioning mode on LIBERO with the Table 1 checkpoint, rolling sampler, $K = 4$, 500 episodes per suite, success rate (%). Policy mode uses a target value trajectory from equation 11 with task length $T$ set to the longest or median demonstration. Joint mode receives no target.

| Mode | Spatial | Object | Goal | Long | Avg. |
|---|---|---|---|---|---|
| Joint (Table 1) | 98.4 | **99.0** | **97.8** | **98.0** | **98.3** |
| Policy, $T$ = longest | **99.0** | 98.0 | 97.2 | 95.6 | 97.4 |
| Policy, $T$ = median | 98.2 | 97.8 | 95.0 | 96.4 | 96.8 |

Table 6: Future-frame and value prediction on LIBERO with the Table 1 checkpoint.

| | PSNR ↑ | SSIM ↑ | LPIPS ↓ | Value MAE ↓ |
|---|---|---|---|---|
| *World-model mode* | | | | |
| Chunk 1 (frames 1–4) | 35.7 | 0.971 | 0.018 | 0.096 |
| Chunk 2 (frames 5–8) | 33.5 | 0.961 | 0.023 | 0.102 |
| Chunk 3 (frames 9–12) | 32.7 | 0.956 | 0.025 | 0.109 |
| Chunk 4 (frames 13–16) | 32.3 | 0.954 | 0.027 | 0.116 |
| All 16 frames | 33.6 | 0.960 | 0.023 | 0.106 |
| *Rolling sampler, closed loop* | | | | |
| Leading chunk | 35.6 | 0.972 | 0.018 | 0.091 |

### 4.6 World-model and value prediction

Table 6 evaluates the generated frames and values on 400 windows from 80 held-out closed-loop episodes, 2 per task. Figure 4 shows generated frames on a LIBERO-Object episode, on a teleoperated real-robot trajectory in world-model mode, and during a closed-loop real-robot run with the rolling sampler. In world-model mode, the model receives four observed frames and the 16 executed actions and predicts the 16 future frames and values, denoising the four chunks together in $K = 40$ steps. Frame metrics are computed against the simulator frames after the policy's preprocessing. Given the executed actions, the PSNR is 35.7 dB on the first chunk and 33.6 dB over all 16 frames, with SSIM 0.971 and 0.960; the error grows with the horizon, as expected for a 1.6 s prediction of a moving arm, but the fourth chunk is still at 32.3 dB. The rolling-sampler row evaluates, in closed loop, the leading chunk whose actions the robot executed; it gives 35.6 dB and SSIM 0.972, in line with the first chunk above, so carrying a chunk across queries does not degrade its frames. The last column measures the value read-out against the demonstration-derived label of equation 11, with the episode length as $T$, on successful episodes. In world-model mode the mean absolute error is 0.096 on the first chunk and 0.106 over the horizon, on a label range of 2, and the rolling sampler's leading chunk in closed loop is at 0.091. The value slot tracks task progress from the frames and actions alone, making the joint-mode read-out usable as a progress estimate.

Figure 4 illustrates the two ways of generating frames. On the LIBERO-Object episode, both the world-model-mode prediction and the leading chunk of the rolling sampler match the simulator frames at the grasp and the release. The teleoperated real-robot trajectory was recorded after training and is not part of the fine-tuning data; in world-model mode, the frames follow the recorded actions: the arm reaches, closes the gripper on the red block, and places it on the yellow block at the same steps as in the recording, and the stacked blocks remain at the end, while fine details such as the exact block position drift at the later steps of a 16-frame window. In the closed-loop run, the rolling sampler generates each leading chunk together with the actions the robot then executes, so its frames predict the outcome of the policy's own motion; they track the observed frames through the reach, the grasp, the stacking, and the retreat, occasionally leading the observation by a few frames as the arm enters the view. The same slots therefore serve as an action-conditioned simulator when actions are supplied and as a preview of the policy's own intended motion when they are not.

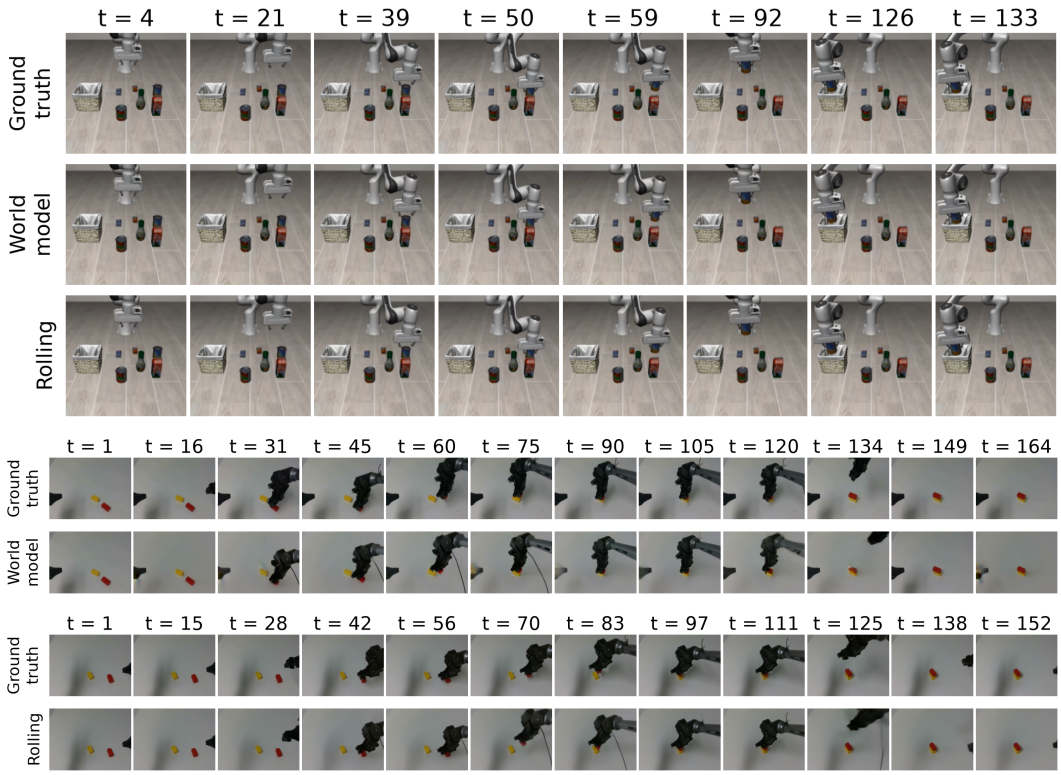



Figure 4: **Generated frames.** Top: a held-out closed-loop LIBERO-Object episode (*pick up the alphabet soup and place it in the basket*) at eight steps around the grasp and the release. Rows show the simulator frames, the world-model-mode prediction given the executed actions (one 16-frame pass per 16 steps, each anchored on the real observation), and the leading chunk of the rolling sampler during the closed-loop run. Middle: world-model mode on a stacking trajectory teleoperated on the real robot after training, at 12 evenly spaced steps (15 Hz). Rows show the recorded frames and the prediction given the recorded actions (one 16-frame pass per 16 steps, each anchored on the recorded observation). Bottom: a closed-loop run of Solis on the real robot at 12 evenly spaced steps. Rows show the frames observed by the head camera and the leading chunk generated by the rolling sampler at the same steps, i.e., the outcome predicted one chunk ahead for the actions that the model then executed on the real robot during the run.

## 5 Conclusion and Discussions

Solis treats future observations, actions, and a task-progress-derived value signal as one trajectory generated by a pretrained video diffusion transformer. Conditioning masks give that trajectory its roles: with actions supplied, the model is a world model; with a target value trajectory, a policy; with nothing supplied, a joint generator of frames, actions, and values. Temporal block-causal attention and chunk-wise noise levels during training let a rolling sampler keep partially denoised chunks across control cycles and refresh them with every new observation, so the policy re-observes after each chunk of four actions at the cost of one network evaluation per chunk in steady state.

**Limitations and Future Work**. The value label is a proxy derived from demonstration length rather than a learned return, and the evaluator mode has not yet been used to select among candidate trajectories at test time. The RoboTwin evaluation covers 12 tasks, and models trained on all 50 tasks with four times more data remain about seven points higher, and generalization to unseen instructions is untested. Sampling remains sequential: with a 5B backbone and $K = 4$, one query takes about 0.5 s on the real robot. Since $K = 4$ matches $K = 40$ on LIBERO, few-step or distilled samplers should raise the control rate. Frame prediction degrades with the horizon and under distribution shift, and the benefit of the rolling horizon concentrates on long-horizon tasks. Scaling real-robot data, failure-aware training, and test-time planning with the world-model and evaluator modes are natural next steps toward faster and more reliable closed-loop control.

## AI USE STATEMENT

In this work, we have not used generative AI tools for any tasks with required disclosure, and all such tasks are not applicable to this work. We used generative AI tools only to polish the writing of parts of the manuscript. The authors reviewed all AI-assisted edits to ensure that they preserve the intended technical meaning and contain no unsupported claims. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

## REPRODUCIBILITY STATEMENT

For reproducibility, the main paper and Appendices A, B, and C detail our methodology and implementation. We will release the complete source code and model weights.

## REFERENCES

Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical AI. *arXiv preprint arXiv:2501.03575*, 2025. URL https://arxiv.org/abs/2501.03575.

AgiBot-World-Contributors, Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, et al. AgiBot World Colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025. URL https://arxiv.org/abs/2503.06669.

Arman Akbari, Ci Zhang, Arash Akbari, Lin Zhao, Yixiao Chen, Weiwei Chen, Xuan Zhang, Geng Yuan, and Yanzhi Wang. Flash-WAM: Modality-aware distillation for world action models. *arXiv preprint arXiv:2606.05254*, 2026. URL https://arxiv.org/abs/2606.05254.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. *arXiv preprint arXiv:2512.13030*, 2025. URL https://arxiv.org/abs/2512.13030.

Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4603–4623, 2024. URL https://proceedings.mlr.press/v235/bruce24a.html.

Boyuan Chen, Diego Marti Monso, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2407.01392.

Siyao Chen, Jiakang Yuan, Jiaxin Wang, and Tao Chen. Trust your instincts: Confidence-driven test-time RL for vision-language-action models. *arXiv preprint arXiv:2606.29892*, 2026. URL https://arxiv.org/abs/2606.29892.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025. URL https://arxiv.org/abs/2506.18088.

Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2302.00111.

Sigmund H. Høeg, Yilun Du, and Olav Egeland. Streaming diffusion policy: Fast policy synthesis with variable noise diffusion models. *arXiv preprint arXiv:2406.04806*, 2024. URL https://arxiv.org/abs/2406.04806.

Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24328–24346, 2025. URL https://arxiv.org/abs/2412.14803.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2506.08009.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. DROID: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems (RSS)*, 2024. URL https://arxiv.org/abs/2403.12945.

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. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. URL https://arxiv.org/abs/2406.09246.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026. URL https://arxiv.org/abs/2601.16163.

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, et al. SimpleVLA-RL: Scaling VLA training via reinforcement learning. *arXiv preprint arXiv:2509.09674*, 2025a. URL https://arxiv.org/abs/2509.09674.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026a. URL https://arxiv.org/abs/2601.21998.

Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model. In *Robotics: Science and Systems (RSS)*, 2025b. URL https://arxiv.org/abs/2503.00200.

Zekai Li, Jiaming Tang, and Zhijian Liu. FlashVLA: Streaming action decoding for fast and asynchronous VLA inference. *arXiv preprint arXiv:2608.27384*, 2026b. URL https://arxiv.org/abs/2608.27384.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2210.02747.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023. URL https://arxiv.org/abs/2306.03310.

Kunhao Liu, Wenbo Hu, Jiale Xu, Ying Shan, and Shijian Lu. Rolling forcing: Autoregressive long video diffusion in real time. *arXiv preprint arXiv:2509.25161*, 2025. URL https://arxiv.org/abs/2509.25161.

Jonas Pai, Liam Achenbach, Victoriano Montesinos, Benedek Forrai, Oier Mees, and Elvis Nava. mimic-video: Video-action models for generalizable robot control beyond VLAs. *arXiv preprint arXiv:2512.15692*, 2025. URL https://arxiv.org/abs/2512.15692.

Quanquan Peng, Yutong Liang, Rui Yan, Nicklas Hansen, and Xiaolong Wang. FACT: Failure-aware causal training for world-action models. *arXiv preprint arXiv:2608.10232*, 2026. URL https://arxiv.org/abs/2608.10232.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. URL https://arxiv.org/abs/2503.20314.

Chaoyang Wang, Wenrui Bao, Sicheng Gao, Bingxin Xu, Yu Tian, Yogesh S. Rawat, Yunhao Ge, and Yuzhang Shang. VLA-Thinker: Boosting vision-language-action models through thinking-with-image reasoning. *arXiv preprint arXiv:2603.14523*, 2026. URL https://arxiv.org/abs/2603.14523.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations (ICLR)*, 2024a. URL https://arxiv.org/abs/2312.13139.

Kun Wu, Chengkai Hou, Jiaming Liu, Zhengping Che, Xiaozhu Ju, Zhuqin Yang, Meng Li, Yinuo Zhao, Zhiyuan Xu, Guang Yang, et al. RoboMIND: Benchmark on multi-embodiment intelligence normative data for robot manipulation. *arXiv preprint arXiv:2412.13877*, 2024b. URL https://arxiv.org/abs/2412.13877.

Yuhao Wu, Yitian Liu, Weijie Shen, Mishuo Han, Wenjie Xu, Haotian Liang, Zhongshan Liu, Yinan Mao, Lei Xu, Xinping Guan, et al. dVLA-RL: Reinforcement learning over denoising trajectories for discrete diffusion vision-language-action models. *arXiv preprint arXiv:2606.23623*, 2026. URL https://arxiv.org/abs/2606.23623.

Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Hengtao Li, Jie Li, Jindi Lv, Jingyu Liu, et al. GigaWorld-Policy: An efficient action-centered world-action model. *arXiv preprint arXiv:2603.17240*, 2026a. URL https://arxiv.org/abs/2603.17240.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026b. URL https://arxiv.org/abs/2602.15922.

Cheng Yin, Yankai Lin, Wang Xu, Sikyuen Tam, Xiangrui Zeng, Zhiyuan Liu, and Zhouping Yin. DeepThinkVLA: Enhancing reasoning capability of vision-language-action models. *arXiv preprint arXiv:2511.15669*, 2025a. URL https://arxiv.org/abs/2511.15669.

Tianwei Yin, Qiang Zhang, Richard Zhang, William T. Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025b. URL https://arxiv.org/abs/2412.07772.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026. URL https://arxiv.org/abs/2603.16666.

Hongzhi Zang, Mingjie Wei, Si Xu, Yongji Wu, Zhen Guo, Yuanqing Wang, Hao Lin, Peihong Wang, Liangzhi Shi, Yuqing Xie, et al. RLinf-VLA: A unified and efficient framework for reinforcement learning of vision-language-action models. *arXiv preprint arXiv:2510.06710*, 2025. URL https://arxiv.org/abs/2510.06710.

Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. In *Robotics: Science and Systems (RSS)*, 2025. URL https://arxiv.org/abs/2504.02792.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183, 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

# A Implementation Details

**Trajectory Configuration.** The observation history contains $P = 4$ frames, including the current frame, and the prediction horizon contains $N = 16$ future frames organized into $H = 4$ chunks of $n = 4$ frames each. Including the initial blank frame, the primary video clip has 21 frames. The latent sequence has 15 slots for a single camera and 16 when an observed wrist-camera latent is included. At $224 \times 224$ input resolution, the backbone divides each latent slot into $2 \times 2$ patches, yielding 49 tokens per slot and 735 tokens for the single-camera sequence.

**Slot Encoding.** We encode $[\mathbf{o}_\emptyset, \mathbf{o}_{t-P+1:t}, \mathbf{o}_{t+1:t+N}]$ as one clip, where $\mathbf{o}_\emptyset$ is a blank frame. The blank frame occupies the initial position of the causal autoencoder, aligning the observed history and future chunks with its temporal groups. The observed history maps to one latent $\mathbf{z}_{\text{obs}}$, and each future chunk maps to one latent $\mathbf{z}_c^o$. The blank latent $\mathbf{z}_\emptyset$ remains in the sequence. When a wrist camera is available, we separately encode $[\mathbf{o}_\emptyset, \mathbf{o}^{\text{w}}_{t-P+1:t}]$, discard its blank latent, and retain its observed-history latent $\mathbf{z}_{\text{w}}$. The conditioning block is $\mathbf{S}_{\text{cond}} = (\mathbf{z}_\emptyset, \mathbf{z}_{\text{obs}}, \mathcal{T}(\mathbf{q}_t), [\mathbf{z}_{\text{w}}])$, with the wrist slot included when available. For a vector $\mathbf{x} \in \mathbb{R}^d$ with $d \le Chw$, $\mathcal{T}$ repeats the flattened vector to fill a $C \times h \times w$ slot and truncates the final copy. The read-out operator $\mathcal{R}$ averages the $\lfloor Chw/d \rfloor$ complete copies, giving $\mathcal{R}(\mathcal{T}(\mathbf{x})) = \mathbf{x}$.

**Action Representation.** All embodiments share a 20-dimensional layout with one 10-dimensional block per arm, comprising end-effector position (3), a 6D rotation (6), and a gripper command (1). The block for an absent arm is zero-filled, and actions are normalized to $[-1, 1]$. A four-action chunk is an 80-dimensional vector, so a $48 \times 14 \times 14$ action slot holds 117 complete copies.

**Value Targets.** A demonstration contains $T$ frames indexed by $k = 0, \ldots, T-1$. For $T > 1$, we assign each frame the scalar target

$$V_k = 2\,\frac{T-k-1}{T-1} - 1 \in [-1, 1]. \tag{11}$$

This target uses the fraction of the demonstration remaining as a proxy for task progress and assumes steady progress. It decreases from $V_0 = 1$ to $V_{T-1} = -1$, so higher values indicate more work remaining. The target requires no reward function and does not represent a reward-based return. For policy-mode inference, we set $T$ to the longest demonstration or successful rollout of the task and clamp $k$ to $T-1$ when execution exceeds this nominal length.

**Diffusion Configuration.** We use $\tau_{\max} = 1000$ for the timestep embedding and sample $\eta$ from a uniform grid of 1000 levels on $(0, 1]$ with shift parameter $s = 5$. The shifted noise level is $\sigma = s\eta/(1 + (s-1)\eta)$. Timestep embeddings are computed per token and feed the adaptive normalization in every transformer block and the output head. Conditioning slots remain clean with probability $p_{\text{clean}} = 0.5$. Otherwise, their noise levels are sampled from $\mathcal{U}(0, \sigma_{\max})$, with $\sigma_{\max} = 0.3$ during pretraining and $0.1$ during fine-tuning. This augmentation is applied independently to observed visual and proprioception slots, which never contribute to the loss. The blank slot always remains clean. The conditioning mask is not a separate network input, as supplied future slots are represented by clean latents with zero timesteps. The loss coefficients in equation 9 are $\lambda_a = 2$ and $\lambda_o = \lambda_V = 1$. Pretraining samples the three conditional modes uniformly.

**Training.** We train all transformer weights and freeze the autoencoder, in bfloat16 with AdamW, weight decay 0.01, linear warmup, and activation checkpointing, on 8 A800 80GB GPUs. Pretraining uses 4 sequences per GPU and 2 gradient-accumulation steps, 64 sequences per optimizer step, learning rate $2 \times 10^{-5}$, 1,000 warmup steps, and a checkpoint every 2,000 steps, 23 in total. Fine-tuning samples world-model, joint, and policy mode with probabilities 0.3, 0.4, and 0.3, uses $\sigma_{\max} = 0.1$, and proceeds in stages, each continuing from the previous checkpoint. LIBERO fine-tuning uses 64 sequences per GPU and 2 accumulation steps, 1,024 per optimizer step, at learning rate $2 \times 10^{-5}$ with action loss weight 2 (equation 9). The stage before the reported checkpoint ran 20,000 steps and the reported stage 5,000. RoboTwin fine-tuning uses 48 sequences per GPU, 768 per optimizer step, at learning rate $5 \times 10^{-5}$ in two stages of 10,000 and 5,000 steps.

**Sampling Configuration.** Unless stated, all results use joint mode and the rolling horizon of Section 3.3 with the $K = 40$ schedule. This schedule costs 40 network evaluations for the first chunk of an episode and 10 per executed chunk afterwards. At the end of initialization, the four chunks have noise levels $\sigma^{(41)} = 0$, $\sigma^{(31)}$, $\sigma^{(21)}$, and $\sigma^{(11)}$. Each query executes the four actions of the leading chunk and then refills the conditioning slots with the new observations.

Table 7: Per-task success (%) on 12 RoboTwin 2.0 tasks (clean/randomized). GW-P: GigaWorld-Policy; FACT+f: FACT with failure co-training; LB-VA: LingBot-VA; F-WAM: Fast-WAM.

| Task | $\pi_0$ | $\pi_{0.5}$ | X-VLA | Motus | GW-P | FACT | FACT+f | LB-VA | F-WAM | **Solis** |
|---|---|---|---|---|---|---|---|---|---|---|
| beat_block_hammer | 79/84 | 96/93 | 92/88 | 95/88 | 86/86 | 75/80 | 81/80 | 96/**98** | **99**/97 | 98/95 |
| blocks_ranking_rgb | 80/63 | 92/85 | 83/83 | 99/97 | 92/96 | 96/92 | 94/93 | 99/98 | **100/100** | 98/99 |
| handover_block | 47/31 | 66/57 | 73/37 | 86/73 | 80/80 | 65/62 | 84/72 | **99**/78 | 95/**81** | 82/78 |
| handover_mic | 97/97 | 98/97 | 0/0 | 78/63 | 72/72 | 99/**100** | **100**/99 | 94/96 | 99/**100** | 86/93 |
| lift_pot | 80/72 | 96/85 | 99/**100** | 96/99 | 98/98 | 99/**100** | **100**/99 | **100**/99 | **100/100** | 98/97 |
| move_can_pot | 68/48 | 51/55 | 89/86 | 34/74 | 76/78 | 80/86 | 80/90 | **94/97** | 90/88 | 92/92 |
| pick_dual_bottles | 59/37 | 93/63 | 47/36 | 96/90 | 86/86 | 87/74 | 97/83 | **100/99** | **100**/96 | 73/76 |
| place_a2b_left | 43/47 | 87/82 | 48/49 | 88/79 | 94/88 | 92/89 | 95/89 | **97/93** | 95/**93** | 80/80 |
| place_empty_cup | 91/85 | **100**/99 | **100**/98 | 99/98 | 90/90 | 99/**100** | **100/100** | **100/100** | **100/100** | **100**/99 |
| place_phone_stand | 49/53 | 81/81 | 88/87 | 87/86 | 82/72 | 89/95 | 90/91 | **97**/97 | **97/99** | 89/95 |
| put_bottles_dustbin | 65/56 | 84/79 | 74/77 | 81/79 | 72/70 | 73/81 | 83/88 | 87/**91** | **95**/90 | 77/75 |
| stack_bowls_two | 94/95 | 95/96 | 96/93 | **98/98** | 96/92 | 97/93 | 94/93 | 94/**98** | 92/**98** | 92/90 |
| Average | 71.0/64.0 | 86.6/81.0 | 74.1/69.5 | 86.4/85.3 | 85.3/84.0 | 87.6/87.7 | 91.5/89.8 | 96.4/**95.3** | **96.8**/95.2 | 88.8/89.1 |

**Rolling Schedule.** The discrete levels are $\sigma^{(j)} = s\eta_j/(1 + (s-1)\eta_j)$, where $\eta_j = 1 - (j-1)/K$ for $j = 1, \ldots, K+1$. At initialization, each chunk begins at schedule position $j = 1$. Chunk $c$ is held at pure noise with zero update for $(c-1)L$ evaluations, where $L = K/H$, and then advances one schedule step per evaluation. After $K$ evaluations, chunk 1 is clean and chunks $2, \ldots, H$ are $L, 2L, \ldots, (H-1)L$ steps behind it. After execution and a horizon shift, the next leading chunk advances from position $K - L + 1$ to $K + 1$ over $L$ evaluations. Each evaluation is a full forward pass under the block-causal mask, without a key-value cache. Clean slots receive no updates, and the sampler restores conditioning slots to their supplied values after each step. At the start of an episode, the initial observation is repeated to fill the history buffer. For standalone world-model sampling, all action slots are fixed while visual and value slots are denoised together over $K$ steps with a common schedule. The autoencoder decoder converts predicted visual slots to frames. Policy-mode targets follow equation 11; evaluator mode fixes visual and action slots and denoises only values.

**Evaluation Protocol.** On LIBERO we run 50 episodes per task, 500 per suite and 2,000 per full evaluation, with step limits of 220, 280, 300, and 520 for Spatial, Object, Goal, and LIBERO-10. On RoboTwin we run 100 episodes per task in each setting. We report success rate in percent, from one policy server per suite on one GPU, with evaluation seed 0 unless stated.

## B Per-Task RoboTwin 2.0 Results

Table 7 lists the per-task success rates behind Table 2. The 12 tasks are the RoboTwin 2.0 subset of Li et al. (2025a), who grouped them by average episode length into short (lift_pot, beat_block_hammer, pick_dual_bottles, place_phone_stand), medium (move_can_pot, place_a2b_left, place_empty_cup, handover_mic), long (handover_block, stack_bowls_two), and extra-long (blocks_ranking_rgb, put_bottles_dustbin) horizons. We do not include their numbers because they fine-tune on 1,000 randomized demonstrations per task and then apply reinforcement learning, a different protocol from the one shared by the methods below. Baseline values are copied from the per-task tables of the cited papers: $\pi_0$, $\pi_{0.5}$, and LingBot-VA from Table S1 of Li et al. (2026a); X-VLA and Motus from Table 14 of Bi et al. (2025); GigaWorld-Policy from Table 8 of Ye et al. (2026a); FACT from Table 6 of Peng et al. (2026); and Fast-WAM from Table 3 of Yuan et al. (2026). All of them train on the benchmark's 50 clean and 500 randomized demonstrations per task, jointly over all 50 tasks, and evaluate 100 episodes per task and setting. The Motus and X-VLA values are reprinted by Peng et al. (2026), Yuan et al. (2026), and Ye et al. (2026a); the only discrepancy is Motus on place_a2b_left in clean scenes, printed as 82 by Li et al. (2026a) and as 88 in the Motus paper, whose value we use. Solis values use the Table 2 checkpoint in joint mode with $K = 40$.

## C Real-Robot Setup and Task Prompts

The platform is an AgileX Cobot Magic with two Piper arms and parallel grippers, a head camera, and a wrist camera on each arm. Demonstrations are collected by leader-follower teleoperation at 15 fps; actions are 16-D absolute end-effector targets for the two arms, position, quaternion, and gripper, converted to the 20-D layout of Section 3.2. At test time the policy receives the head and

Table 8: Real-robot tasks: prompt, number of fine-tuning demonstrations, and mean duration.

| Task | Prompt | Demos | Length |
|---|---|---|---|
| Stack blocks | “Stack the red block on top of the yellow block.” | 145 | 7.5 s |
| Sort numbers | “Arrange the wooden blocks into the designated slots.” | 158 | 9.4 s |

wrist images and the current end-effector pose and executes four actions per query with the rolling sampler at $K = 4$, i.e., one network evaluation per query after the first, which takes about 0.5 s per query on a single NVIDIA RTX 4090 in the robot workstation (about 1.2 s for the first query). A trial succeeds when the red block rests on the yellow block (stacking) or when every block sits in its designated slot (sorting); object positions are randomized between trials. Table 8 lists the prompts, the number of demonstrations, and their mean duration for each task.