ACCELERATING WORLD ACTION MODELS VIA PARALLEL SAMPLING

Anonymous authors
Paper under double-blind review

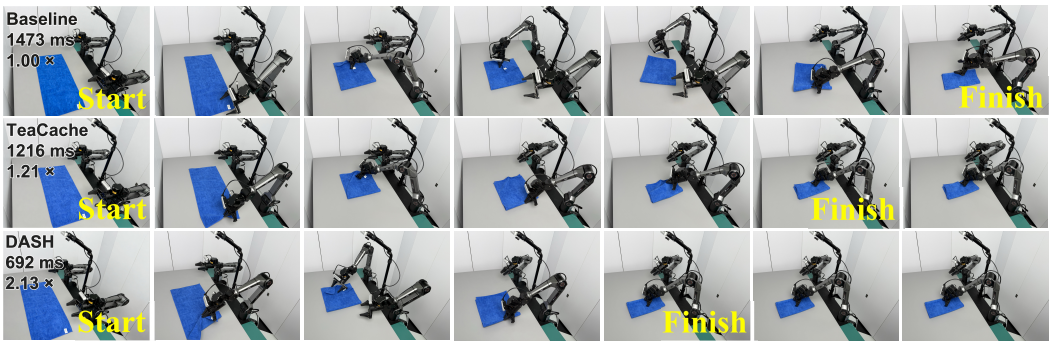

Figure 1: Comparison of sequential sampling, TeaCache (Liu et al., 2025a), and our DASH on the real-world "`Fold a Towel`" task with Cosmos Policy (Kim et al., 2026) as the backbone.

ABSTRACT

World Action Models (WAMs) offer a promising approach to robot learning, but iterative sampling incurs substantial inference latency, limiting closed-loop responsiveness. Fast samplers and caching methods reduce the number or cost of model evaluations but typically retain sequential sampling dependencies. We present **DASH**, a training-free framework that accelerates pretrained WAM sampling through parallel trajectory refinement to reduce sequential execution depth. DASH initializes new denoising states by extrapolating from model predictions at the current anchor and refines multiple positions concurrently through Picard iteration. To determine window advancement, it estimates how residual inconsistencies in video and action states contribute to deviations in the generated action chunk, accepting a contiguous prefix that satisfies a cumulative contribution budget at each position. Our approach achieves 1.71–2.66× wall-clock speedups over sequential sampling across multiple WAMs on RoboTwin 2.0 while maintaining comparable task performance. Evaluation on three real-world manipulation tasks of increasing difficulty further demonstrates a better performance–latency trade-off than the evaluated baselines.

1 INTRODUCTION

Robot learning requires policies that translate observations of the physical environment into effective actions. World Action Models (WAMs) (Intelligence et al.; Ye et al., 2026b; Kim et al., 2026; Li et al., 2026b; Bi et al., 2026; Li et al., 2026a; Ye et al., 2026a; Team et al., 2026b; Chen et al., 2026; Team et al., 2026a; Yuan et al., 2026) connect visual dynamics modeling with action generation, allowing policies to draw on physical priors learned from video. However, WAMs that jointly predict visual futures and actions must process high-dimensional video representations alongside action states during iterative sampling (Bi et al., 2026; Ye et al., 2026b). Generating an action chunk can involve repeated evaluations of an expensive video-generation backbone, with successive sampling updates depending on preceding states. The resulting latency delays action updates based on new observations, potentially slowing the robot's response to environmental changes. Reducing inference latency is important for translating WAM capabilities into responsive closed-loop control.

A range of methods has been developed to accelerate sampling in diffusion- and flow-based models, including those used for robot control. One category uses distillation to reduce sampling steps (Akbari et al., 2026; Lyu et al., 2026; Prasad et al., 2024) or lightweight architectures and simplified

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.

generation pipelines to reduce computation (Yuan et al., 2026; Li et al., 2026c). These approaches require additional training and model adaptation. Another category accelerates pretrained models at test time: fast numerical solvers reduce model evaluations (Lu et al., 2022; 2025a), while caching reuses features or model outputs across denoising steps (Liu et al., 2025a; Lyu et al., 2025). However, aggressive step reduction or computational reuse can compromise sampling accuracy. In WAMs, inaccurate visual estimates under few-step sampling can also impair action prediction (Ye et al., 2026b). These solver- and caching-based approaches primarily reduce the number or cost of model evaluations, while generally retaining sequential dependencies between sampling steps.

To accelerate sampling beyond step reduction, we build on parallel sampling methods that trade compute for lower latency in pretrained diffusion models. Shih et al. (2023) refine multiple denoising positions concurrently through Picard iteration, Tang et al. (2024) accelerate fixed-point iteration for triangular nonlinear equations, and Lu et al. (2025b) develop a hierarchical solver based on banded nonlinear equations. These methods require no retraining and empirically preserve quality comparable to sequential sampling. However, effective parallel refinement in WAMs requires deciding when to accept intermediate estimates without excessive refinement. In joint video–action generation, inconsistencies in either modality can affect the action chunk through subsequent updates. Local state changes or residual magnitudes alone do not directly quantify this downstream influence, motivating acceptance decisions guided by the estimated impact on action output.

To address these challenges, we introduce **DASH**, a parallel sampling framework for pretrained WAMs that requires no changes to model architectures or parameters. Since parallel refinement starts from estimated intermediate states, their initialization affects the number of refinement rounds needed. We use model predictions at the last accepted state to extrapolate newly entering denoising states, then refine them concurrently through Picard iteration within a sliding window. To guide window advancement, DASH estimates how inconsistencies with the sequential sampling updates in both video and action states contribute to deviations in the generated action chunk. It accepts a contiguous prefix whose cumulative estimated contribution, including contributions from previously accepted positions, remains within the prescribed budget. This directs acceptance toward preserving the action output while allowing the remaining states to undergo further refinement.

We evaluate DASH on RoboTwin 2.0 (Chen et al., 2025) using four WAMs: Motus (Bi et al., 2026), Lingbot-VA (Li et al., 2026b), X-WAM (Guo et al., 2026), and ImageWAM-4B (Zhang et al., 2026), comparing against TeaCache (Liu et al., 2025a) and ParaSolver (Lu et al., 2025b). Beyond simulation, we evaluate Motus, Efficient-WAM (Li et al., 2026a), and Cosmos Policy (Kim et al., 2026) on three real-world manipulation tasks of increasing difficulty. Figure 1 illustrates towel folding with Cosmos Policy, achieving a $2.13\times$ inference speedup over sequential sampling. The key contributions of our work are summarized as follows:

- We propose **DASH**, a parallel sampling framework for pretrained WAMs that reduces sequential evaluation depth by refining multiple denoising positions concurrently, without modifying model architectures or parameters.
- We accelerate sampling in pretrained WAMs through anchor-based initialization and parallel refinement guided by video and action sampling residuals, allowing prefixes to advance once their estimated cumulative contribution to final-action deviation satisfies the budget.
- We evaluate multiple WAMs, obtaining 1.71–$2.66\times$ wall-clock speedups on RoboTwin 2.0 and 1.71–$2.21\times$ speedups based on average inference latency across three real-world manipulation tasks, while maintaining competitive task performance.

2 Related Work

World Action Models. Advances in video generation have motivated robot policies that predict future observations to support dynamics modeling and action prediction (Du et al., 2023; Wu et al., 2024; Zhou et al., 2024; Bharadhwaj et al., 2024; Won et al., 2025; Cheang et al., 2024; Kim et al., 2026; Zhou et al., 2025). World Action Models (WAMs) further integrate visual prediction with action generation. Joint approaches include DreamZero and Cosmos Policy, which incorporate actions into video generative models (Ye et al., 2026b; Kim et al., 2026), and UWM and CoVAR, which couple visual and action generation within diffusion frameworks (Zhu et al., 2025; Yang et al., 2025). Other approaches generate future visual trajectories and infer actions through inverse dynamics or

causal action prediction (Pai et al., 2025; Li et al., 2026b; Hu et al., 2024; Feng et al., 2025). Iterative denoising can introduce substantial inference cost (Liang et al., 2025; Bi et al., 2026), while open-loop visual predictions may diverge from actual execution (Du et al., 2023; Zhou et al., 2025). We focus on training-free parallel sampling for pretrained WAMs by concurrently refining multiple denoising positions without modifying their architectures or parameters.

Efficient Inference for Generative Policies. Iterative sampling incurs substantial inference latency through repeated, sequential model evaluations. Fast diffusion and flow solvers reduce the number of evaluations (Song et al., 2020a; Lu et al., 2022; 2025a), while distillation enables few-step generation, including action generation in robot policies (Prasad et al., 2024; Wang et al., 2024). Caching and feature prediction reduce evaluation costs by reusing intermediate computations (Liu et al., 2025a;b; Lyu et al., 2025), but retain sequential dependencies between sampling positions. Parallel sampling addresses these dependencies by refining multiple positions concurrently (Shih et al., 2023; Lu et al., 2025b; Tang et al., 2024). We extend this direction to pretrained diffusion- or flow-based WAMs through trajectory initialization and adaptive window advancement. Our advancement criterion accounts for video–action coupling using estimated contributions of sampling inconsistencies in both modalities to deviations in the generated action chunk.

3 PRELIMINARIES

Sequential Sampling in World Action Models. Generative World Action Models (WAMs) produce action chunks and, depending on the architecture, future visual predictions conditioned on observations, proprioception, and instructions, collectively denoted c. Our formulation uses flow-based WAMs (Section 4). The framework extends to diffusion-based WAMs with deterministic single-step samplers, including Euler-discretized probability-flow ODEs (Song et al., 2020b) (Appendix M).

In the joint setting, let $\mathbf{X}(u) = (\mathbf{V}(u), \mathbf{A}(u))$ denote the state at sampling progress $u \in [0, 1]$, initialized as $\mathbf{X}(0) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. A continuous, piecewise-smooth noise schedule $\boldsymbol{\tau}(u) = (\tau^v(u), \tau^a(u))$ has nonincreasing components and runs from $(1, 1)$ to $(0, 0)$. Let f_θ^m denote the model velocity with respect to the modality-specific noise variable τ^m. Along the prescribed schedule, $d\mathbf{X}^m/du = \dot{\tau}^m(u) f_\theta^m(\mathbf{X}(u), \boldsymbol{\tau}(u), c)$, giving

$$\mathbf{X}^m(u) = \mathbf{X}^m(0) + \int_0^u \dot{\tau}^m(r) f_\theta^m(\mathbf{X}(r), \boldsymbol{\tau}(r), c)\, dr, \qquad m \in \{v, a\}. \tag{1}$$

Here, $\mathbf{X}^v = \mathbf{V}$ and $\mathbf{X}^a = \mathbf{A}$, and both velocity components depend on the joint state. For $0 = u_0 < \cdots < u_T = 1$, let $\boldsymbol{\tau}_i = \boldsymbol{\tau}(u_i)$ and $\mathbf{X}_i \approx \mathbf{X}(u_i)$. Freezing the velocity at the preceding state and noise levels gives the Euler update in noise coordinates, $\mathbf{X}_i^m = \mathbf{X}_{i-1}^m + \Delta\tau_{i-1}^m f_\theta^m(\mathbf{X}_{i-1}, \boldsymbol{\tau}_{i-1}, c)$, where $\Delta\tau_{i-1}^m = \tau_i^m - \tau_{i-1}^m$. We denote this joint update by $\mathbf{X}_i = G_i(\mathbf{X}_{i-1}, c)$, $i = 1, \ldots, T$. Fixed initial noise, conditioning, and schedule uniquely determine the sequential discrete reference. Each update requires its predecessor state, creating a sequential chain of model evaluations.

Picard Iteration for Parallel Sampling. To parallelize this discrete sampling process, existing methods treat intermediate states as jointly unknown variables and iteratively solve for the trajectory (Shih et al., 2023; Tang et al., 2024). Define the increment $\Delta_i(\mathbf{X}, c) = G_i(\mathbf{X}, c) - \mathbf{X}$, corresponding to the step-scaled joint velocity above, and collect the unknown states into $\mathbf{Z} = (\mathbf{X}_1, \ldots, \mathbf{X}_T)$. Summing the sequential increments yields

$$\mathbf{Z} = \mathcal{P}(\mathbf{Z}), \qquad [\mathcal{P}(\mathbf{Z})]_j = \mathbf{X}_0 + \sum_{i=1}^{j} \Delta_i(\mathbf{X}_{i-1}, c), \quad j = 1, \ldots, T. \tag{2}$$

Subtracting adjacent cumulative equations recovers the sequential recurrence, so the fixed point is unique and coincides with the sequential discrete reference. Starting from initial trajectory estimates $\mathbf{X}_1^{(0)}, \ldots, \mathbf{X}_T^{(0)}$, Picard iteration evaluates this operator on previous-round states:

$$\mathbf{X}_j^{(k+1)} = \mathbf{X}_0 + \sum_{i=1}^{j} \Delta_i(\mathbf{X}_{i-1}^{(k)}, c), \qquad j = 1, \ldots, T, \tag{3}$$

where $k = 0, 1, \ldots$ indexes refinement rounds and $\mathbf{X}_0^{(k)} = \mathbf{X}_0$. Increments are evaluated independently within a round and assembled through a prefix sum. In exact arithmetic, full-trajectory

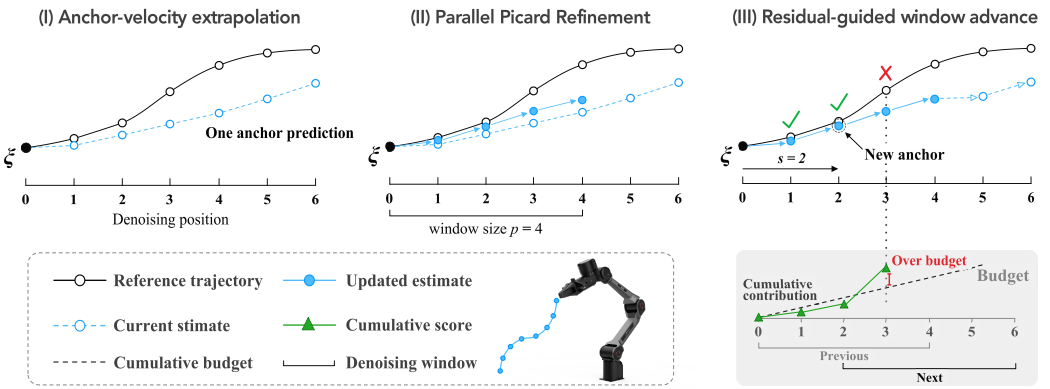

Figure 2: Overview of **DASH**. The reference trajectory denotes sequential Euler sampling from the same initial state, shown for illustration. (I) Anchor-velocity extrapolation initializes window states from one joint video–action velocity prediction at the anchor. (II) Parallel Picard refinement evaluates velocities concurrently and updates states by cumulative integration from the fixed anchor. (III) Residual-guided window advance compares cumulative contribution scores against position-dependent budgets. With $p = 4$, the first budget violation occurs at position 3, so the window advances by $s = 2$ to anchor 2, retaining unaccepted estimates and extrapolating new states.

iteration gives $\mathbf{X}_j^{(k)} = \mathbf{X}_j^*$ for $0 \le j \le \min(k, T)$, recovering the sequential reference $\mathbf{X}^*$ within T rounds (Shih et al., 2023) (Appendix I). Practical acceleration depends on reaching sufficient accuracy in fewer rounds and on the cost of parallel evaluation.

4 Method

4.1 Problem Formulation

Sequential sampling requires each update to wait for its predecessor state, limiting parallel execution. We instead use estimated predecessor states to refine multiple denoising positions concurrently, then determine which estimates to accept. After defining the sampling problem, we describe three stages: anchor-velocity extrapolation for initialization, parallel Picard refinement for state updates, and residual-guided window advance for acceptance (Figure 2).

We consider a pretrained flow-based WAM with joint velocity field $f_\theta = (f_\theta^v, f_\theta^a)$, conditioned on visual observations, proprioception, and a language instruction, collectively denoted c. The model jointly generates future video $\mathbf{v}$ and an action chunk $\mathbf{a}$ from $p_\theta(\mathbf{v}, \mathbf{a} \mid c)$. Sampling proceeds through states $\mathbf{X}_i = (\mathbf{V}_i, \mathbf{A}_i)$, $i = 0, \dots, T$, containing future-video latents and action-chunk representations at noise levels $\boldsymbol{\tau}_i = (\tau_i^v, \tau_i^a)$. Given a fixed schedule, sequential Euler sampling maps initial noise $\mathbf{X}_0$ to $\mathbf{X}_T$ through T model evaluations. Video decoding and required action de-normalization yield $(\mathbf{v}, \mathbf{a})$. For fixed initial noise and conditioning, this trajectory defines our *sequential reference*. We aim to reduce sequential evaluation depth and latency while limiting action-chunk deviations from this reference, without modifying model architecture or parameters.

4.2 Anchor-Velocity Extrapolation

Parallel refinement starts from estimated window states, and their initialization can affect the refinement needed before acceptance. We initialize newly entering positions by extrapolating both video and action states using a single joint velocity prediction from the WAM f_θ at the current anchor, while retaining previously refined estimates.

At refinement round k, the joint state at denoising position i is $\mathbf{X}_i^{(k)} = (\mathbf{V}_i^{(k)}, \mathbf{A}_i^{(k)})$, where $\mathbf{V}_i^{(k)}$ contains future-video latents and $\mathbf{A}_i^{(k)}$ contains the action chunk. The corresponding noise levels for video and action are $\boldsymbol{\tau}_i = (\tau_i^v, \tau_i^a)$. We use T denoising steps, with $\tau_0^v = \tau_0^a = 1$ and $\tau_T^v = \tau_T^a = 0$. Sampling starts from independent Gaussian noise,

$$\mathbf{X}_0^{(0)} = (\boldsymbol{\xi}^v, \boldsymbol{\xi}^a), \quad \boldsymbol{\xi}^v \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \boldsymbol{\xi}^a \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{4}$$

with the observed video, proprioception, and language held fixed as conditioning c throughout the entire sampling process.

The current anchor b marks the end of the accepted prefix, with $b = 0$ initially. The window contains candidate states $b+1, \ldots, b+w$, where $w = \min(p, T-b)$ and p is the maximum window width. When additional states are needed, we obtain the joint velocity $f_\theta(\mathbf{X}_b^{(k)}, \boldsymbol{\tau}_b, c)$ at the anchor and extend the retained trajectory using this fixed prediction. Each newly introduced position i is initialized in increasing index order as

$$\begin{aligned} \mathbf{V}_i^{(k)} &\leftarrow \mathbf{V}_{i-1}^{(k)} + \Delta\tau_{i-1}^v f_\theta^v \left(\mathbf{X}_b^{(k)}, \boldsymbol{\tau}_b, c\right), \\ \mathbf{A}_i^{(k)} &\leftarrow \mathbf{A}_{i-1}^{(k)} + \Delta\tau_{i-1}^a f_\theta^a \left(\mathbf{X}_b^{(k)}, \boldsymbol{\tau}_b, c\right), \end{aligned} \tag{5}$$

where $\Delta\tau_{i-1}^m = \tau_i^m - \tau_{i-1}^m$ is the signed step size for modality $m \in \{v, a\}$. The video and action velocity components, f_θ^v and f_θ^a, are predicted from the same joint anchor state.

Initially, we extend $\mathbf{X}_0$ across the first window. After the window advances, updated overlapping states are preserved, and only newly entering positions are extrapolated from the retained rightmost state using the new anchor's velocity. Without window advancement, no states are reinitialized.

Extrapolation supplies starting estimates rather than accepted updates. Subsequent Picard refinement evaluates the WAM at the window's predecessor states and their corresponding noise levels.

4.3 Parallel Picard Refinement

We instantiate the Picard iteration in Eq. (3) within a sliding window anchored at the last accepted state. After anchor-velocity extrapolation initializes newly entering positions (Section 4.2), each round evaluates velocities concurrently at the window's predecessor states and assembles updated estimates through cumulative sums.

At refinement round k, the anchor b defines a window containing positions $b+1, \ldots, b+w$, where $w = \min(p, T-b)$. Updating these positions requires velocity predictions at predecessor positions $i = b, \ldots, b+w-1$, using their current states and noise levels. An available anchor prediction is reused only when its state, noise levels, and conditioning remain unchanged. All remaining predictions are evaluated concurrently. The window states are updated from the fixed anchor:

$$\begin{aligned} \mathbf{V}_{b+j}^{(k+1)} &= \mathbf{V}_b^{(k)} + \sum_{i=b}^{b+j-1} \Delta\tau_i^v f_\theta^v \left(\mathbf{X}_i^{(k)}, \boldsymbol{\tau}_i, c\right), \\ \mathbf{A}_{b+j}^{(k+1)} &= \mathbf{A}_b^{(k)} + \sum_{i=b}^{b+j-1} \Delta\tau_i^a f_\theta^a \left(\mathbf{X}_i^{(k)}, \boldsymbol{\tau}_i, c\right), \qquad j = 1, \ldots, w. \end{aligned} \tag{6}$$

Here, $\Delta\tau_i^m = \tau_{i+1}^m - \tau_i^m$ is the signed step size for modality $m \in \{v, a\}$. The anchor remains fixed within the round, $\mathbf{X}_b^{(k+1)} = \mathbf{X}_b^{(k)}$. Both velocity components depend on the joint state, preserving video–action coupling during refinement. Observed-video latents, proprioception, and language remain fixed as conditioning c. The estimates are passed to the window-advancement stage.

4.4 Residual-Guided Window Advance

Parallel refinement produces updated estimates but does not determine how many positions to accept. We first estimate how their remaining inconsistencies with sequential Euler updates contribute to deviations in the generated action chunk. We then use these contribution scores to accept a contiguous prefix under a cumulative budget and advance the window.

Residual Contribution Estimation. Because Picard refinement evaluates velocities at pre-update states, an updated state may still disagree with the Euler update from its updated predecessor. We express this inconsistency as the local equation residual:

$$\mathbf{d}_i^{(k+1)} = \mathbf{X}_i^{(k+1)} - G_i \left(\mathbf{X}_{i-1}^{(k+1)}, c\right), \qquad i = b+1, \ldots, b+w, \tag{7}$$

where G_i is the sequential Euler map defined in Section 3. Subtracting adjacent updates in Eq. (6) and substituting into Eq. (7) gives, in exact arithmetic,

$$\mathbf{d}_i^{m,(k+1)} = \Delta\tau_{i-1}^m \left[f_\theta^m \left(\mathbf{X}_{i-1}^{(k)}, \boldsymbol{\tau}_{i-1}, c \right) - f_\theta^m \left(\mathbf{X}_{i-1}^{(k+1)}, \boldsymbol{\tau}_{i-1}, c \right) \right], \qquad m \in \{v, a\}. \tag{8}$$

The residual arises from the velocity change induced by updating the predecessor (Appendix J). Direct evaluation would require additional model predictions at updated predecessors. We use calibrated sensitivities to estimate residual magnitudes from predecessor changes and account for their propagation to the terminal action. This gives the terminal-action contribution score

$$\eta_i^{(k+1)} = \gamma_i^v \left\| \mathbf{V}_{i-1}^{(k+1)} - \mathbf{V}_{i-1}^{(k)} \right\|_v + \gamma_i^a \left\| \mathbf{A}_{i-1}^{(k+1)} - \mathbf{A}_{i-1}^{(k)} \right\|_a . \tag{9}$$

Here, $\eta_i^{(k+1)}$ estimates the contribution of local inconsistency at position i to deviation of the generated action chunk from the sequential reference. The nonnegative weights γ_i^v, γ_i^a account for local velocity sensitivity and subsequent cross-modal propagation. The video norm $\|\cdot\|_v$ is a fixed normalized RMS over future-video latent elements. The action norm $\|\cdot\|_a$ is the maximum absolute change across chunk positions and dimensions after fixed per-dimension scaling (Appendix H).

To construct the weights, we incorporate sampling step sizes into the calibrated velocity sensitivities:

$$\widehat{\mathbf{L}}_i = \begin{pmatrix} \widehat{L}_i^{vv} & \widehat{L}_i^{va} \\ \widehat{L}_i^{av} & \widehat{L}_i^{aa} \end{pmatrix}, \qquad \mathbf{K}_i = \operatorname{diag}\left(|\Delta\tau_{i-1}^v|, |\Delta\tau_{i-1}^a| \right) \widehat{\mathbf{L}}_i. \tag{10}$$

The matrix $\widehat{\mathbf{L}}_i \in \mathbb{R}_{\geq 0}^{2\times 2}$ estimates velocity sensitivities at position $i-1$, with rows indexing output modalities and columns indexing input modalities. Each entry is calibrated using the corresponding input and output norms. Its off-diagonal entries estimate the sensitivity of each modality's predicted velocity to changes in the other modality. Accordingly, $\mathbf{K}_i$ maps predecessor-change magnitudes to estimated video and action residual magnitudes.

An Euler update adds a velocity increment to its state. We use $\mathbf{I}_2 + \mathbf{K}_\ell$ to estimate error propagation through step ℓ, where $\mathbf{I}_2$ accounts for the input error and $\mathbf{K}_\ell$ accounts for the change in the velocity increment. Propagating the local residual estimate through subsequent steps gives

$$[\gamma_i^v, \gamma_i^a] = [0, 1](\mathbf{I}_2 + \mathbf{K}_T) \cdots (\mathbf{I}_2 + \mathbf{K}_{i+1})\mathbf{K}_i. \tag{11}$$

Here, $\mathbf{I}_2$ is the 2×2 identity matrix and $[0, 1]$ selects the terminal action component. The rightmost factor $\mathbf{K}_i$ estimates the local residual, while the remaining factors propagate its contribution toward the terminal state. At $i = T$, the product is empty and equals $\mathbf{I}_2$, yielding $[\gamma_T^v, \gamma_T^a] = [0, 1]\mathbf{K}_T$.

The sensitivity weights are calibrated offline with the WAM parameters fixed and reused during sampling, so scoring requires no additional model evaluations (Appendix L). These empirical scores estimate contributions to final-action deviation without imposing a separate constraint on the final video. They provide no certified error guarantee, and the derivation above assumes exact arithmetic.

Cumulative Budgeting. We use the contribution scores to select a contiguous prefix of the updated window. A cumulative budget lets neighboring positions share the available allowance while retaining contributions recorded in earlier rounds.

Let C_b denote the accumulated contribution at anchor b, with $C_0 = 0$. We allocate a total indicator budget $\varepsilon_A \geq 0$ uniformly by denoising-step index, giving a prefix ending at global position $b + j$ the allowance $\varepsilon_A(b + j)/T$. The advancement stride is

$$s = \min \left(\left\{ j \in \{1, \ldots, w\} : C_b + \sum_{i=b+1}^{b+j} \eta_i^{(k+1)} > \varepsilon_A \frac{b+j}{T} \right\} \cup \{w + 1\} \right) - 1. \tag{12}$$

Here, j is a candidate prefix length. The stride s selects the prefix ending immediately before the first cumulative budget violation. If all prefixes satisfy their allowances, $s = w$. In exact arithmetic, the fixed anchor gives $\eta_{b+1}^{(k+1)} = 0$. Together with the previously satisfied condition $C_b \leq \varepsilon_A b/T$, this ensures that the first candidate passes and $s \geq 1$.

After accepting positions $b + 1, \ldots, b + s$, we update the accumulated contribution:

$$C_{b+s} = C_b + \sum_{i=b+1}^{b+s} \eta_i^{(k+1)}, \tag{13}$$

Table 1: **Evaluation on RoboTwin 2.0** under clean and randomized settings. We report average success rate, closed-loop inference latency, and speedup over sequential inference for each model and setting.

Method	Clean			Random		
	SR (%) ↑	Latency (ms) ↓	Speedup ↑	SR (%) ↑	Latency (ms) ↓	Speedup ↑
Motus (Bi et al., 2026)	88.7%	1772	1.00×	86.8%	1833	1.00×
+ TeaCache	87.4%	1295	1.37×	84.8%	1348	1.36×
+ ParaSolver	87.2%	1237	1.43×	86.3%	1246	1.47×
+ DASH (ours)	**89.5%**	**936**	**1.89×**	**88.3%**	**949**	**1.93×**
Lingbot-VA (Li et al., 2026b)	92.9%	8174	1.00×	91.5%	8233	1.00×
+ TeaCache	91.3%	9190	0.89×	90.0%	8860	0.93×
+ ParaSolver	90.8%	15480	0.53×	89.8%	15560	0.53×
+ DASH (ours)	**92.7%**	**4725**	**1.73×**	**92.1%**	**4812**	**1.71×**
X-WAM (Guo et al., 2026)	**90.5%**	686	1.00×	89.9%	695	1.00×
+ TeaCache	90.1%	357	1.92×	88.3%	384	1.81×
+ ParaSolver	88.7%	483	1.42×	87.5%	478	1.45×
+ DASH (ours)	90.2%	**258**	**2.66×**	**90.3%**	**267**	**2.60×**
ImageWAM-4B (Zhang et al., 2026)	**93.2%**	275	1.00×	**93.6%**	278	1.00×
+ TeaCache	92.0%	168	1.64×	92.9%	173	1.61×
+ ParaSolver	90.9%	206	1.33×	92.2%	209	1.33×
+ DASH (ours)	92.7%	**116**	**2.37×**	93.2%	**115**	**2.42×**

and advance the anchor to $b + s$, using $\mathbf{X}_{b+s}^{(k+1)}$ as the anchor state for round $k + 1$. Accepted states and their recorded contributions remain fixed. Updated but unaccepted states are retained for subsequent refinement. Their scores are recomputed in later rounds and added to the accumulated contribution only upon acceptance. AVE initializes only newly entering positions.

The allowance follows the global denoising position without resetting as the window advances. In exact arithmetic, each round advances at least one position, reaching T within T rounds. Completion ensures $C_T \leq \varepsilon_A$, constraining the accumulated indicator without certifying terminal-action deviation. Bounding this deviation from sequential Euler sampling with identical initial noise, conditioning, and schedule requires valid sensitivity bounds and numerical-error accounting (Appendix K).

5 Experiments

5.1 Simulation Evaluation

Baselines and Setup. We evaluate **DASH** on RoboTwin 2.0 (Chen et al., 2025), a benchmark covering diverse bimanual manipulation tasks, using Motus (Bi et al., 2026), Lingbot-VA (Li et al., 2026b), X-WAM (Guo et al., 2026), and ImageWAM-4B (Zhang et al., 2026). Following the multi-task training protocol of Bi et al. (2026), models are trained on 2,500 clean and 25,000 domain-randomized demonstrations, with variations in backgrounds, clutter, table height, and lighting. For each model, we compare standard sequential inference with TeaCache (Liu et al., 2025a) (feature caching), ParaSolver (Lu et al., 2025b) (parallel sampling), and DASH. We conduct 100 trials per task in each of the clean and randomized settings, reporting mean success rate across tasks (%) ↑, closed-loop inference latency (ms/chunk) ↓, and speedup over sequential inference ↑.

Results. As shown in Table 1, DASH achieves 1.71–2.66× speedups across four WAMs on RoboTwin 2.0 (Chen et al., 2025), consistently outperforming TeaCache (Liu et al., 2025a) and ParaSolver (Lu et al., 2025b) in both latency and success rate. Compared with sequential inference, success-rate decreases are limited to 0.5 percentage points, with improvements in several settings. Notably, on Lingbot-VA (Li et al., 2026b), TeaCache and ParaSolver both increase latency, whereas DASH achieves 1.71–1.73× acceleration with comparable success rates. On X-WAM (Guo et al., 2026), DASH delivers 2.60–2.66× speedups while maintaining success rates around 90%, demonstrating a favorable balance between sampling efficiency and task performance.

5.2 Real-World Evaluation

Baselines and Setup. We use an AgileX PiPER platform (Figure 5) with two 6-DoF arms, each equipped with a two-finger gripper, and three RealSense D435i cameras, two wrist-mounted and

Table 2: **Evaluation on three real-world manipulation tasks of increasing difficulty.** We report task-wise success rates (SR) and average closed-loop inference latency across the three tasks.

Method	Success Rate (%) ↑			Inference Efficiency	
	Cube into Plate	Bottle to Tray	Fold Towel	Latency (ms/chunk) ↓	Speedup ↑
Motus (Bi et al., 2026)	**95.0%**	65.0%	35.0%	3264	1.00×
+ TeaCache	75.0%	55.0%	20.0%	2458	1.33×
+ ParaSolver	80.0%	55.0%	25.0%	2915	1.12×
+ DASH (ours)	90.0%	**70.0%**	**35.0%**	**1914**	**1.71×**
Efficient-WAM (Li et al., 2026a)	90.0%	**75.0%**	30.0%	461	1.00×
+ TeaCache	80.0%	60.0%	25.0%	389	1.19×
+ ParaSolver	80.0%	55.0%	10.0%	352	1.31×
+ DASH (ours)	**95.0%**	70.0%	**40.0%**	**209**	**2.21×**
Cosmos Policy (Kim et al., 2026)	100.0%	**95.0%**	**75.0%**	1473	1.00×
+ TeaCache	90.0%	80.0%	50.0%	1216	1.21×
+ ParaSolver	100.0%	90.0%	65.0%	950	1.55×
+ DASH (ours)	**100.0%**	90.0%	70.0%	**692**	**2.13×**

one head-mounted. As shown in Figure 3, we evaluate three tasks of increasing difficulty: *Place a Cube into a Plate*, *Transfer a Bottle to a Target Tray*, and *Fold a Towel*. We evaluate Motus (Bi et al., 2026), Efficient-WAM (Li et al., 2026a), and Cosmos Policy (Kim et al., 2026), comparing sequential inference with TeaCache (Liu et al., 2025a), ParaSolver (Lu et al., 2025b), and DASH. For each task, we collect 100 demonstrations and fine-tune each policy for 20k steps. Each model–method combination is evaluated over 20 trials per task.

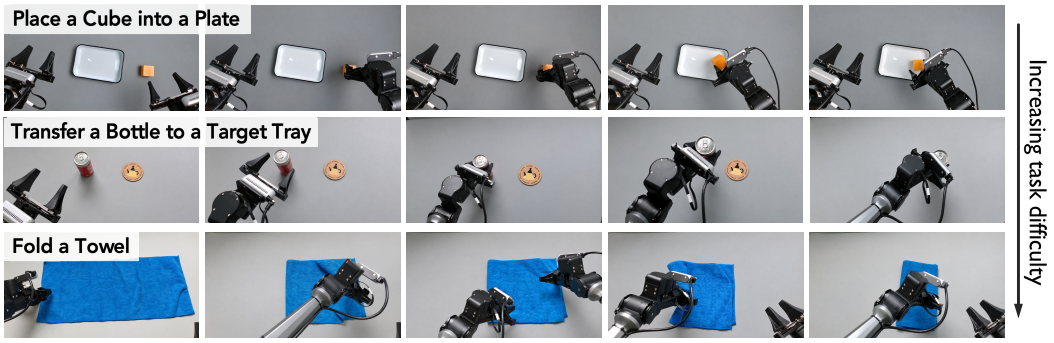

Figure 3: **Real-world manipulation tasks**, with difficulty increasing from top to bottom.

Results. Table 2 shows that DASH maintains competitive performance while reducing inference latency across three real-world tasks. On towel folding, it matches Motus (Bi et al., 2026) under sequential inference at 35% success and improves Efficient-WAM (Li et al., 2026a) from 30% to 40%, with respective speedups of 1.71× and 2.21× based on average latency across tasks. For Cosmos Policy (Kim et al., 2026), latency falls from 1473 to 692 ms, while success rates remain within 5 percentage points of sequential inference. Compared with TeaCache (Liu et al., 2025a) and ParaSolver (Lu et al., 2025b), DASH achieves lower latency for every model and equal or higher success rates on every task, supporting its effectiveness in real-world deployment.

5.3 Ablation Studies

Q1: Does anchor-velocity extrapolation improve sampling efficiency?

Informative initialization may reduce refinement, but obtaining the anchor velocity may require an additional model evaluation. We test whether anchor-velocity extrapolation (AVE) reduces overall latency and whether video extrapolation adds value beyond action extrapolation. We compare *constant initialization*, which copies the rightmost retained joint state to new positions; *action-only AVE*, which extrapolates actions using the anchor's action velocity and copies the rightmost retained video state; and *joint AVE*, which extrapolates both modalities using the anchor's joint velocity. All variants retain previously refined states and use identical sampling schedules, Picard updates, and verification settings, with window size $p = 5$.

We evaluate Motus (Bi et al., 2026) on RoboTwin 2.0 (Chen et al., 2025) under the randomized setting, reporting task-averaged success rate (SR), end-to-end inference latency including anchor

Table 3: **Ablation of Anchor-velocity extrapolation** on RoboTwin 2.0 under the randomized setting.

Method	SR ↑	Latency ↓	Speedup ↑
Motus (Bi et al., 2026)	86.8%	1833 ms	1.00×
+ Constant initialization	84.8%	1344 ms	1.36×
+ Action-only AVE	85.3%	1273 ms	1.44×
+ Joint AVE (ours)	**88.3%**	**949 ms**	**1.93×**

Table 4: **Ablation of contribution budget** ε_A on RoboTwin 2.0 under the randomized setting.

Budget ε_A	SR ↑	NFE/GPU ↓	Latency ↓	Speedup ↑
Motus (Bi et al., 2026)	86.8%	10	1833 ms	1.00×
$0.25\varepsilon_0$	87.8%	4.43	1358 ms	1.35×
$0.50\varepsilon_0$	87.2%	4.04	1151 ms	1.59×
$\mathbf{1.00}\varepsilon_0$	**88.3%**	3.63	949 ms	1.93×
$2.00\varepsilon_0$	85.6%	**3.23**	**925 ms**	**1.98×**

evaluations, and speedup over sequential inference. Table 3 shows similar success rates and latencies for constant initialization and action-only AVE, suggesting limited benefit from action extrapolation alone in this setting. Joint AVE reduces latency by 25.5% relative to action-only AVE and improves SR by 3.0 percentage points, achieving a 1.93× speedup over sequential inference. These results favor joint extrapolation for its performance–latency trade-off.

Q2: How does the contribution budget ε_A affect speed and task success?

A larger contribution budget ε_A can allow longer accepted prefixes and fewer refinement rounds, at the cost of tolerating greater estimated inconsistency. We vary $\varepsilon_A \in \{0.25, 0.5, 1, 2\}\varepsilon_0$, where ε_0 is the default budget, with other settings fixed. We report SR, end-to-end inference latency, and NFE/GPU on RoboTwin 2.0 under the randomized setting. NFE/GPU denotes the total number of joint-state velocity-field evaluations per action chunk, including additional anchor evaluations, divided by the number of GPUs used by the corresponding method. Sequential inference uses one GPU, whereas DASH distributes evaluations across multiple GPUs.

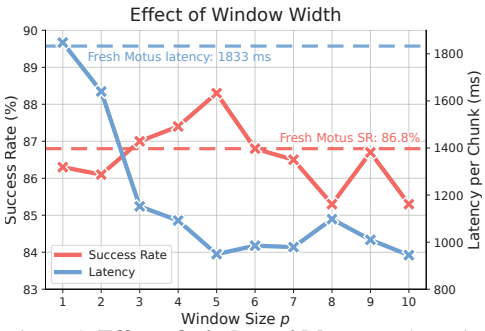

Figure 4: **Effect of window width** p on RoboTwin 2.0 under the randomized setting.

As shown in Table 4, increasing ε_A from $0.25\varepsilon_0$ to ε_0 reduces NFE/GPU from 4.43 to 3.63 and latency from 1358 to 949 ms, while SR remains between 87.2% and 88.3%. Increasing the budget further to $2\varepsilon_0$ lowers NFE/GPU to 3.23 but yields a 2.5% latency reduction, while SR drops to 85.6%. These results favor the default budget for its balance between latency and task success.

Q3: How does window size p affect efficiency?

The window size p sets the maximum number of denoising positions refined concurrently in each Picard round. A larger window permits greater advancement, but the accepted prefix is determined by the cumulative budgeting rule, and evaluating more states can increase per-round cost. We vary p from 1 to 10 while fixing the model, sampling schedule, total contribution budget, and GPU resources, and measure inference latency and task success.

As shown in Figure 4, $p = 1$ performs sequential sampling, with latency and SR close to the sequential baseline. Increasing p to 5 reduces latency to 949 ms and yields the highest SR of 88.3%, achieving a 1.93× speedup. Larger windows offer no consistent latency improvement and yield lower SR, making $p = 5$ a favorable choice in this setting. This may reflect higher per-round costs and limited advancement when estimates farther from the anchor require more refinement.

6 Conclusion

In this paper, we presented **DASH**, a parallel sampling framework that reduces sequential evaluation depth in pretrained WAMs without modifying their architectures or parameters. It initializes denoising states through anchor-velocity extrapolation and refines multiple positions concurrently through Picard iteration. Window advancement uses estimated contributions of video and action sampling inconsistencies to final-action deviation, accepting contiguous prefixes under a cumulative contribution budget. Across multiple WAMs, experiments show wall-clock inference speedups over sequential inference of 1.71–2.66× on RoboTwin 2.0 and 1.71–2.21× based on average latency across three real-world manipulation tasks, while maintaining competitive task success rates.

AI USE STATEMENT

We used generative AI for language polishing, LaTeX formatting, numerical consistency checks, review of mathematical arguments, interpretation of results, and feedback on evaluation methodology. The authors are responsible for validating AI-assisted suggestions and for the accuracy and integrity of the manuscript, including methods, proofs, experimental results, and conclusions.

ETHICS STATEMENT

This work investigates inference acceleration for pretrained robot policies through simulation and real-world manipulation experiments. We acknowledge the sources of the pretrained models and benchmarks used in our study. Ethical considerations include the responsible use of these resources and physical safety in real-world robotic experimentation and deployment.

REPRODUCIBILITY STATEMENT

Section 5 describes the experimental protocols and evaluation metrics. Appendices A– E describe the implementation setup, real-world evaluation procedures, and model-specific sampling configurations. Appendix D reports additional ablations.

REFERENCES

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.

Homanga Bharadhwaj, Debidatta Dwibedi, Abhinav Gupta, Shubham Tulsiani, Carl Doersch, Ted Xiao, Dhruv Shah, Fei Xia, Dorsa Sadigh, and Sean Kirmani. Gen2act: Human video generation in novel scenarios enables generalizable robot manipulation. *arXiv preprint arXiv:2409.16283*, 2024.

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026.

Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, et al. Gr-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv preprint arXiv:2410.06158*, 2024.

Ronghan Chen, Yandan Yang, Zuojin Tang, Dongjie Huo, Tong Lin, Haoning Wu, Haoyun Liu, Yuzhi Chen, Lulu Zheng, Botai Yuan, et al. Abot-m0. 5: Unified mobility-and-manipulation world action model. *arXiv preprint arXiv:2607.00678*, 2026.

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.

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36:9156–9172, 2023.

Yao Feng, Chendong Xiang, Xinyi Mao, Hengkai Tan, Zuyue Zhang, Shuhe Huang, Kaiwen Zheng, Haitian Liu, Hang Su, and Jun Zhu. Vidarc: Embodied video diffusion model for closed-loop control. *arXiv preprint arXiv:2512.17661*, 2025.

Jun Guo, Qiwei Li, Peiyan Li, Zilong Chen, Nan Sun, Yifei Su, Heyun Wang, Yuan Zhang, Xinghang Li, and Huaping Liu. Unified 4d world action modeling from video priors with asynchronous denoising. *arXiv preprint arXiv:2604.26694*, 2026.

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. *arXiv preprint arXiv:2412.14803*, 2024.

Physical Intelligence, Bo Ai, Ali Amin, R Aniceto, A Balakrishna, G Balke, K Black, G Bokinsky, S Cao, T Charbonnier, et al. π0. 7: a steerable generalist robotic foundation model with emergent capabilities, 2026. *URL https://arxiv. org/abs/2604.15483*.

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

Jiajun Li, Tiecheng Guo, Yifan Ye, Rongyu Zhang, Xiaowei Chi, Qianpu Sun, Ying Li, Yunfan Lou, Yan Huang, Zhihe Lu, et al. Efficient-wam: A 1b-parameter world-action model with low-cost future imagination. *arXiv preprint arXiv:2606.10040*, 2026a.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026b.

Ziang Li, Dongzhou Cheng, Yibin Wang, Shiyue Wang, Xiaoyang Xu, Lingxuan Weng, Juan Wang, and Jiaqi Wang. Light-wam: Efficient world action models with state-fusion action decoding. *arXiv preprint arXiv:2606.08242*, 2026c.

Junbang Liang, Pavel Tokmakov, Ruoshi Liu, Sruthi Sudhakar, Paarth Shah, Rares Ambrus, and Carl Vondrick. Video generators are robot policies. *arXiv preprint arXiv:2508.00795*, 2025.

Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7353–7363. IEEE, 2025a.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with taylorseers. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15853–15863. IEEE, 2025b.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, 22 (4):730–751, 2025a.

Jianrong Lu, Zhiyu Zhu, and Junhui Hou. Parasolver: A hierarchical parallel integral solver for diffusion models. In *International Conference on Learning Representations*, volume 2025, pp. 41678–41704, 2025b.

Jing Lyu, Shuanghao Bai, Runze Xiao, Zhenyu Liao, Wenxing Tan, Zihan Tang, Ruochuan Shi, Cheng Peng, Yuheng Ji, Yihao Wang, et al. Dido: Distilling interaction-centric dynamics into one-step denoising for world action models. *arXiv preprint arXiv:2609.15570*, 2026.

Zhengyao Lyu, Chenyang Si, Junhao Song, Zhenyu Yang, Yu Qiao, Ziwei Liu, and Kwan-Yee K Wong. Fastercache: Training-free video diffusion model acceleration with high quality. In *International Conference on Learning Representations*, volume 2025, pp. 33132–33156, 2025.

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.

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. *arXiv preprint arXiv:2405.07503*, 2024.

Andy Shih, Suneel Belkhale, Stefano Ermon, Dorsa Sadigh, and Nima Anari. Parallel sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36:4263–4276, 2023.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Zhiwei Tang, Jiasheng Tang, Hao Luo, Fan Wang, and Tsung-Hui Chang. Accelerating parallel sampling of diffusion models. In *Forty-first International Conference on Machine Learning*, 2024.

GigaWorld Team, Angen Ye, Angyuan Ma, Boyuan Wang, Chaojun Ni, Fangzheng Ye, Guan Huang, Guo Li, Guosheng Zhao, Haodong Yan, et al. Gigaworld-policy-0.5: A faster and stronger wam empowered by autoresearch. *arXiv preprint arXiv:2607.13960*, 2026a.

MotuBrain Team, Chendong Xiang, Fan Bao, Haitian Liu, Hengkai Tan, Hongzhe Bi, James Li, Jiabao Liu, Jingrui Pang, Kiro Jing, et al. Motubrain: An advanced world action model for robot control. *arXiv preprint arXiv:2604.27792*, 2026b.

Zhendong Wang, Zhaoshuo Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, et al. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. *arXiv preprint arXiv:2410.21257*, 2024.

John Won, Kyungmin Lee, Huiwon Jang, Dongyoung Kim, and Jinwoo Shin. Dual-stream diffusion for world-model augmented vision-language-action model. *arXiv preprint arXiv:2510.27607*, 2025.

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*, volume 2024, pp. 10641–10662, 2024.

Liudi Yang, Yang Bai, George Eskandar, Fengyi Shen, Mohammad Altillawi, Dong Chen, Ziyuan Liu, and Abhinav Valada. Covar: Co-generation of video and action for robotic manipulation via multi-modal diffusion. *arXiv preprint arXiv:2512.16023*, 2025.

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.

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.

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.

Yuyang Zhang, Wenyao Zhang, Zekun Qi, He Zhang, Haitao Lin, Jingbo Zhang, Yao Mu, Xiaokang Yang, Wenjun Zeng, and Xin Jin. Imagewam: Do world action models really need video generation, or just image editing? *arXiv preprint arXiv:2606.19531*, 2026.

Pengfei Zhou, Liliang Chen, Shengcong Chen, Di Chen, Wenzhi Zhao, Rongjun Jin, Guanghui Ren, and Jianlan Luo. Act2goal: From world model to general goal-conditioned policy. *arXiv preprint arXiv:2512.23541*, 2025.

Siyuan Zhou, Yilun Du, Jiaben Chen, Yandong Li, Dit-Yan Yeung, and Chuang Gan. Robodreamer: Learning compositional world models for robot imagination. *arXiv preprint arXiv:2404.12377*, 2024.

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. *arXiv preprint arXiv:2504.02792*, 2025.

APPENDIX

A IMPLEMENTATION DETAILS.

We implement DASH in PyTorch on multiple NVIDIA L40S GPUs, using a persistent, one-process-per-GPU producer–consumer execution scheme with PyTorch Distributed and NCCL. For parallel Picard iteration, we use a default window size of $p = 5$ and a total terminal-action contribution budget of $\varepsilon_A = 1.0$ for cumulative verification.

B REAL-WORLD EVALUATION DETAILS

Setup and Protocol. Real-world experiments use a bimanual AgileX PiPER platform with two 6-DoF PiPER arms, each equipped with a two-finger gripper, as shown in Figure 5. Visual observations come from three RealSense D435i cameras: one mounted on each wrist for eye-in-hand views and one head camera for a third-person view. We evaluate three tabletop tasks of increasing difficulty, *Place a Cube into a Plate*, *Transfer a Bottle to a Target Tray* and *Fold a Towel*, which progress from a small rigid object with a large placement tolerance, to a tall object that must be transported and released upright, to a deformable object. For each task we collect 100 demonstrations and fine-tune each policy for 20k steps. All inference configurations of a given model, namely standard sequential sampling, TeaCache, ParaSolver and DASH, share the same fine-tuned weights and differ only in the sampling procedure, so differences in task outcome isolate the effect of the sampler.

Each configuration is evaluated over 20 trials per task. At the start of every trial the manipulated object is placed at a randomized initial position within the reachable workspace, and the target container is placed within the workspace at the same time; the policy then runs closed-loop from the current observations until it releases the object and retracts, or until a fixed time limit is reached, in which case the trial is counted as a failure. A trial is judged successful by a human operator only if the task-specific criteria below are satisfied at the end of the trial; partial completion is counted as a failure. We report the success rate as the proportion of successful trials, together with inference latency per action chunk measured on the same hardware, and speedup relative to each model's sequential inference on the same task.

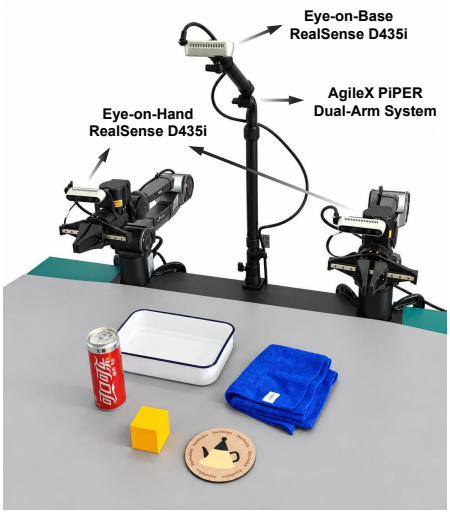

Figure 5: Real-world experimental setup.

Place a Cube into a Plate. The robot must grasp a small cube from its randomized initial position, lift it clear of the table, transport it to a plate and release it. The trial is counted as a success only when the cube rests entirely within the rim of the plate after the gripper opens and retracts, and remains there without rolling out. Failures include missing or dropping the cube, placing it on the rim or outside the plate, and displacing the plate during placement. This task requires a stable grasp on a small rigid object followed by an accurate placement, and serves as the baseline test of whether an accelerated sampler preserves basic pick-and-place precision.

Transfer a Bottle to a Target Tray. The robot must grasp an upright bottle, lift it, transport it to a target tray and set it down. The trial is counted as a success only when the bottle stands upright with its base fully inside the tray boundary after the gripper releases, and remains standing. A bottle that tips over, whether inside or outside the tray, or that is released with its base partially outside the tray, counts as a failure, as does dropping the bottle during transport. Compared with the cube, the bottle is taller and has a higher center of mass, so it is sensitive to gripper orientation during transport and to the timing and smoothness of the release; the task therefore tests whether the accelerated sampler maintains consistent action chunks over a longer motion, since inconsistencies between consecutive chunks manifest directly as tipping at release.

Fold a Towel. A towel is laid flat on the table with a randomized position and orientation, and the robot must fold it once in half by grasping it along one edge and bringing that edge over to the opposite edge. The trial is counted as a success only when, after the gripper releases, the towel lies folded with the two halves overlapping, the moved edge approximately aligned with the opposite edge, and no corner left unfolded or hanging off the folded shape. Failures include missing the grasp, dragging or crumpling the towel instead of folding it, an incomplete fold that leaves a large portion of the towel exposed, and a fold that opens again after release. This is the most demanding of the three tasks: it involves a deformable object whose configuration changes throughout the motion, so the policy must react to the evolving visual state across the entire trajectory, and it is the task on which any degradation in closed-loop responsiveness or in the coherence of successive action chunks is most likely to appear.

C Model Details

This section describes the models evaluated in our simulation and real-world experiments, focusing on their architectures, visual–action coupling, and sampling configurations.

Motus. Motus (Bi et al., 2026) is an 8B unified latent action world model comprising a Wan2.2-5B video backbone, a 2.13B vision–language model, a 641.5M action expert, and a 253.5M understanding expert. It is trained with rectified flow matching and supports separate video and action noise coordinates, τ^v and τ^a. In our experiments' joint generation mode, each denoising evaluation predicts both video and action velocity fields. Our RoboTwin 2.0 configuration uses 10 Euler integration steps, jointly refining future video latents and the action chunk. The generated video latents provide an internal future representation and need not be decoded to RGB for action execution.

LingBot-VA. LingBot-VA (Li et al., 2026b) is a 5.3B causal video–action model with a dual-stream Mixture-of-Transformers architecture. Its Wan2.2-5B video stream and narrower action stream have hidden widths of 3072 and 768, respectively, and share a depth of 30 layers. The model generates chunks autoregressively while retaining historical context in key–value caches. Within each chunk, the video stream first predicts future latent visual states through flow matching; the action stream then generates actions through inverse dynamics conditioned on these predictions and the preceding observation–action history. Our RoboTwin 2.0 configuration uses 25 Euler integration steps for video prediction and 50 for action generation, with guidance scales of 5.0 and 1.0, respectively.

X-WAM. X-WAM (Guo et al., 2026) is a joint video–action flow-matching model built on the Wan2.2-TI2V-5B backbone. Video latents, proprioceptive states, and actions are processed in a shared token sequence with modality-specific noise schedules. Our RoboTwin 2.0 configuration specifies 50 denoising steps for video and 10 each for actions and future proprioceptive states. Under asynchronous denoising, actions and proprioceptive states reach their clean endpoints after the first 10 joint denoising evaluations, while video remains partially denoised. Actions can therefore be dispatched before the remaining video steps are completed; the released policy implementation supports early stopping at this point. The model predicts $H = 8$ future frames and proprioceptive states together with an action chunk of $K = 32$, corresponding to four actions per future frame. The auxiliary depth branch is disabled in our policy evaluation.

ImageWAM-4B. ImageWAM (Zhang et al., 2026) uses image-editing representations of a future endpoint frame to condition action generation. The 4B variant pairs a FLUX.2 [klein] 4B image-editing backbone with a 642M action DiT. At inference, an editing-branch forward pass at a fixed timestep $\tau^\star$ produces layer-wise key–value caches, without running a full image-denoising trajectory or decoding the target image. These caches remain fixed while the action expert denoises the action chunk through flow matching. The visual branch contributes one context-extraction pass, while refinement occurs in the action branch. Our RoboTwin 2.0 configuration uses 10 action denoising steps, an input resolution of 288×256, and an action chunk length of 16.

Efficient-WAM. Efficient-WAM (Li et al., 2026a) is an approximately 1B-parameter world–action model that couples a compact video expert with a dedicated action expert through a Mixture-of-Transformers architecture. The video expert is initialized by selecting layers and channels from Wan2.2-5B and further trained with teacher-guided distillation. Action tokens attend to visual features through layer-wise cross-modal attention. The framework supports a multiscale visual layout that retains detailed current observations while representing predicted futures with fewer tokens.

It also supports asymmetric denoising: the video branch is refreshed at selected iterations, and its cached features condition subsequent action updates. Our real-world experiments use 10 action denoising steps per sampling trajectory.

Cosmos Policy. Cosmos Policy (Kim et al., 2026) adapts Cosmos-Predict2-2B to robot control through latent frame injection, without adding a separate action expert. Normalized action chunks, proprioceptive states, and values are replicated and reshaped into latent frames alongside image latents, allowing all modalities to be modeled by the same diffusion transformer. The model uses an EDM-based denoising formulation. In direct-policy mode, it jointly denoises the action chunk, future observations and proprioception, and future-state value, conditioned on the current observations, robot state, and task instruction. The action chunk is recovered by averaging the replicated action entries and reversing normalization, without VAE decoding. Future-state and value predictions are not required for action selection in this mode; they support the model's separate planning mode. Our real-world experiments use 10 diffusion denoising steps per action chunk.

D Additional Ablation Studies

How do video changes and error propagation affect residual verification? We investigate whether residual verification benefits from accounting for video-state changes and error propagation through subsequent denoising steps. Checking action changes alone may miss the influence of evolving video states, while local residual estimates may not capture their eventual impact on the final action chunk. For *w/o video change*, we set $\gamma_i^v = 0$ in Eq. (9) while retaining the original action weight γ_i^a. For *w/o error propagation*, we replace the scoring weights with $[\gamma_i^v, \gamma_i^a] = [0, 1]K_i$, retaining the local action-residual estimate but omitting subsequent propagation. *Full verification (ours)* uses the complete terminal-action contribution score. All variants share the same AVE initialization, sampling schedule, window width, Picard updates, and cumulative budget rule. We compare task success rate and inference latency to assess each factor's contribution to the performance–latency trade-off. As illustrated in Table 5, removing the video-change term reduces latency by only 3.9% but lowers SR by 6.8 percentage points relative to full verification. Omitting error propagation reduces latency by 28.1% but incurs a larger SR drop of 13.2 percentage points. These results support retaining both components to maintain task performance, with full verification still achieving a $1.93\times$ speedup over sequential inference.

Table 5: **Ablation of residual verification components** on RoboTwin 2.0 under the randomized setting.

Method	SR ↑	Latency ↓	Speedup ↑
Motus (Bi et al., 2026)	86.8%	1833 ms	$1.00\times$
w/o video change	81.5%	912 ms	$2.01\times$
w/o error propagation	75.1%	**682 ms**	$\mathbf{2.69}\times$
Full verification (ours)	**88.3%**	949 ms	$1.93\times$

Effect of Cumulative Budgeting on Window Advancement. Our cumulative budgeting rule allows later denoising positions to use allowances left unused by earlier positions. This ablation tests whether such sharing reduces inference latency while maintaining task success. We compare *per-position budgeting*, which requires $\eta_i \le \varepsilon_A/T$ at every accepted position, with *cumulative budgeting (ours)*, which requires $C_b + \sum_{\ell=b+1}^{i} \eta_\ell \le \varepsilon_A i/T$ at every accepted position i. Here, b is the current anchor index, and C_b records the cumulative contribution of previously accepted positions. Both rules accept a contiguous prefix, stopping before the first violation. We keep the full contribution score, total budget ε_A, AVE initialization, window width, and Picard updates fixed, and report SR, inference latency, and speedup. Table 6 shows that cumulative budgeting reduces latency by 17.8% and improves SR by 5.6 percentage points over per-position budgeting, achieving a $1.93\times$ speedup over sequential inference. These results support sharing unused allowances across positions to improve the performance–latency trade-off under the same total contribution budget.

Table 6: **Ablation of budgeting strategies** on RoboTwin 2.0 under the randomized setting.

Method	SR ↑	Latency ↓	Speedup ↑
Motus (Bi et al., 2026)	86.8%	1833 ms	$1.00\times$
Per-position budget	82.7%	1154 ms	$1.59\times$
Cumulative budget (ours)	**88.3%**	**949 ms**	$\mathbf{1.93}\times$

E Computational Overhead and Parallel Efficiency

Algorithm inefficiency. We define algorithm inefficiency as the ratio of mean total model-state evaluations (NFE) per action chunk between DASH and sequential sampling. We evaluate 10-step

sampling configurations across 50 RoboTwin 2.0 (Chen et al., 2025) tasks, matching input observations, initial noise, and schedules between methods for each configuration. Each evaluated denoising state is counted separately, including evaluations within parallel batches, repeated refinement evaluations, and additional anchor evaluations; reused predictions incur no additional NFE. State updates, contribution scoring, and communication are excluded from this count. Averaged over the evaluated configurations and tasks, DASH requires 16.1 evaluations per chunk, including 3.75 anchor evaluations, compared with 10 for sequential sampling. This yields an algorithm inefficiency of 1.61, corresponding to a 61% increase in model-state evaluations. This metric quantifies the additional evaluation workload, while the execution gains from parallel hardware are assessed below.

Hardware efficiency. We assess multi-GPU execution gains using the default window size $p = 5$ on one and five NVIDIA L40S GPUs. Let $L_{\mathrm{DASH}}(p, G)$ denote the mean end-to-end policy inference latency per action chunk with window size p and G GPUs. We define the execution gain as $H(p, G) = L_{\mathrm{DASH}}(p, 1)/L_{\mathrm{DASH}}(p, G)$, holding the sampling configuration and model-evaluation workload fixed. Inference latency decreases from 2706 to $949\,\mathrm{ms}$, yielding $H(5, 5) = 2.85$. We define net speedup over sequential sampling as $S(p, G) = L_{\mathrm{baseline}}/L_{\mathrm{DASH}}(p, G)$, where L_{baseline} is the latency of the single-GPU, 10-evaluation sequential sampler measured on paired inputs. With $L_{\mathrm{baseline}} = 1833\,\mathrm{ms}$, this gives $S(5, 5) = 1.93$. All timings use the same measurement boundaries and include observation processing, conditioning computations, denoising, action postprocessing, and required communication. Thus, the $2.85\times$ gain measures multi-GPU acceleration of the same DASH workload, whereas the $1.93\times$ speedup measures the improvement over sequential sampling after accounting for additional evaluations and inference overhead.

F Limitations

DASH requires additional model evaluations to reduce sequential execution depth. Its latency gains therefore depend on sufficient parallel computing resources: when parallel execution cannot compensate for the extra evaluations and communication overhead, the method may provide little speedup or run slower than sequential sampling. Our acceptance criterion also depends on sensitivity estimates calibrated from a finite set of state pairs. If these estimates underestimate the sensitivities encountered during sampling, satisfying the contribution budget does not ensure that the actual action deviation from the sequential reference remains below that budget. Applying the method to new models, sampling schedules, or input distributions therefore requires validating the sensitivity estimates and, when necessary, recalibrating them.

G Qualitative Results and Visualization

We examine how inference acceleration affects predicted videos and action trajectories using Motus (Bi et al., 2026). We compare sequential sampling (Fresh, the reference), TeaCache (Liu et al., 2025a) (threshold $= 0.2$), ParaSolver (Lu et al., 2025b) ($\tau = 0.1$), and **DASH** ($\varepsilon_A = 1.0\varepsilon_0$) under matched input observations and initial noise. Predicted video frames are aligned by future time index. For action visualization, we select the two arm joints with the largest baseline command ranges across both arms and the gripper with the larger baseline range. Each range is computed over the current observed state and the predicted action chunk. Selection depends only on the Fresh baseline, and identical dimensions are displayed for all methods.

We present four tasks: `"handover block"` (Figure 6), `"move pillbottle pad"` (Figure 7), `"lift pot"` (Figure 8), and `"move stapler pad"` (Figure 9). Across these examples, all three acceleration methods produce future video frames visually similar to Fresh, although visual similarity alone does not establish agreement in the predicted actions. The selected arm-joint trajectories largely overlap, whereas differences are more apparent in the gripper commands. In the displayed gripper trajectories, **DASH** generally follows the sequential reference more closely than TeaCache and ParaSolver. These observations provide qualitative evidence of reference-action fidelity in the displayed dimensions, rather than establishing agreement across the entire action chunk or improved task success. This behavior is consistent with our design goal of guiding parallel sampling by the estimated impact of sampling inconsistencies on the action output.

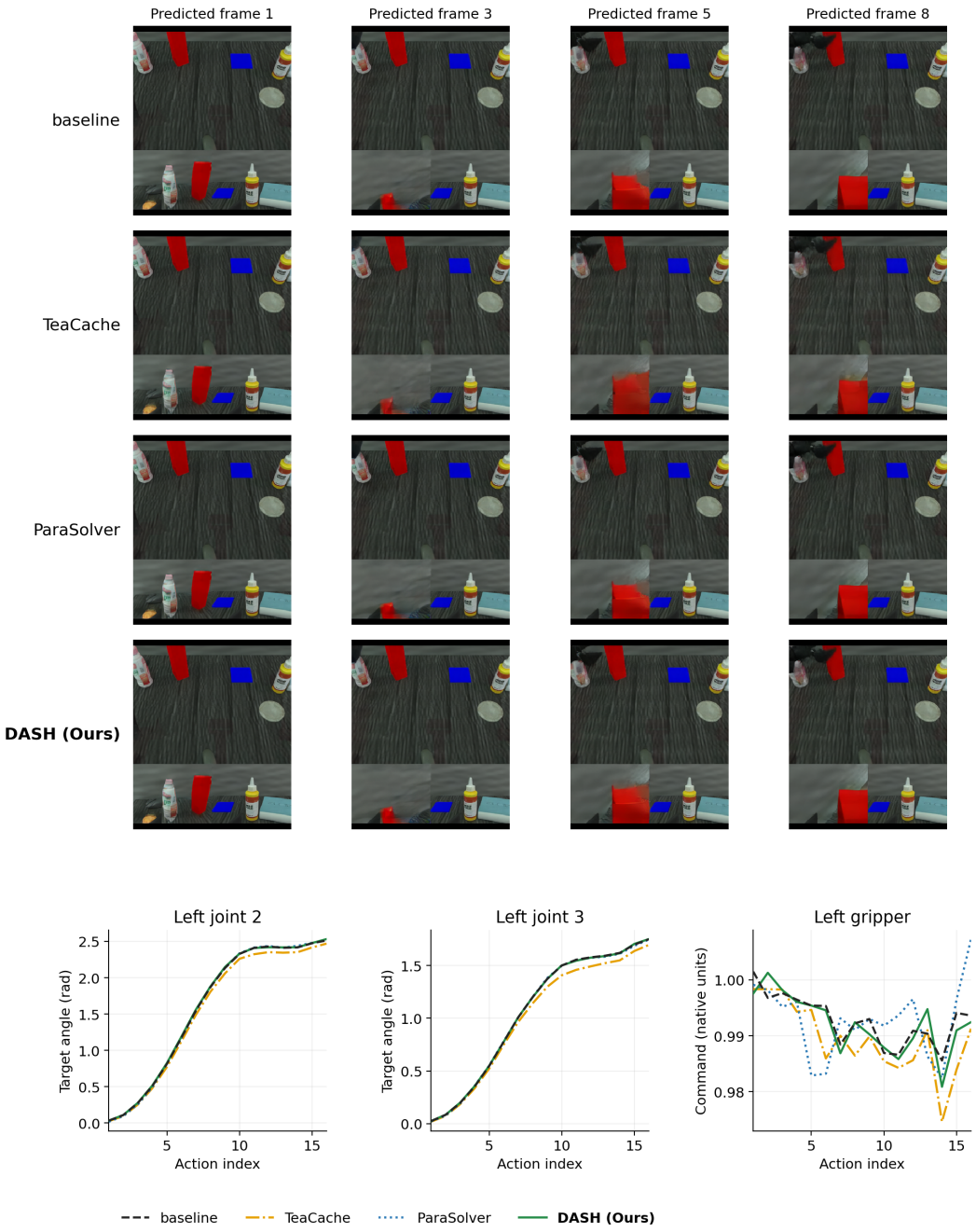

Figure 6: **Qualitative results** on the "`handover block`".

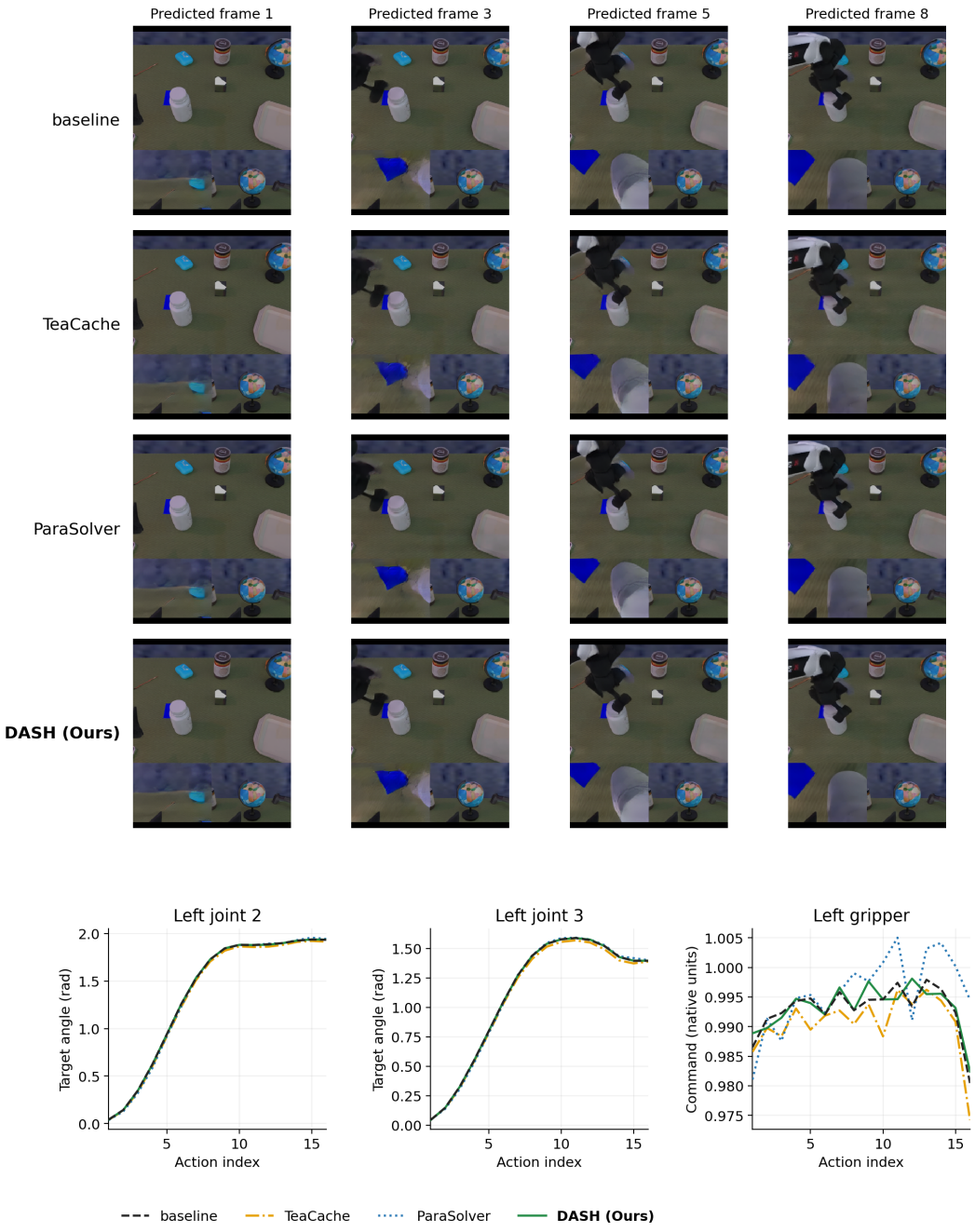

Figure 7: **Qualitative results** on the "`move pillbottle pad`".

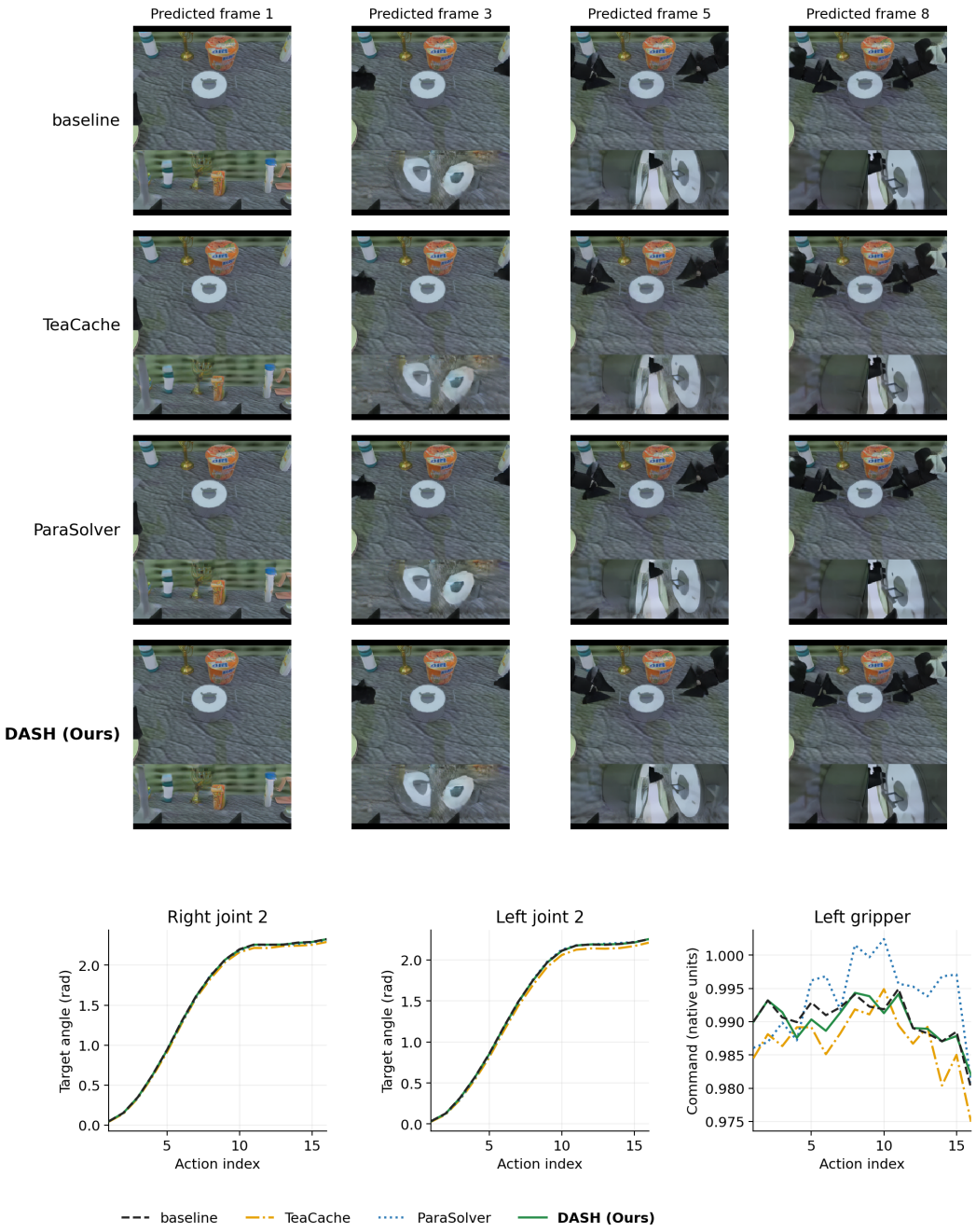

Figure 8: **Qualitative results** on the "`lift pot`".

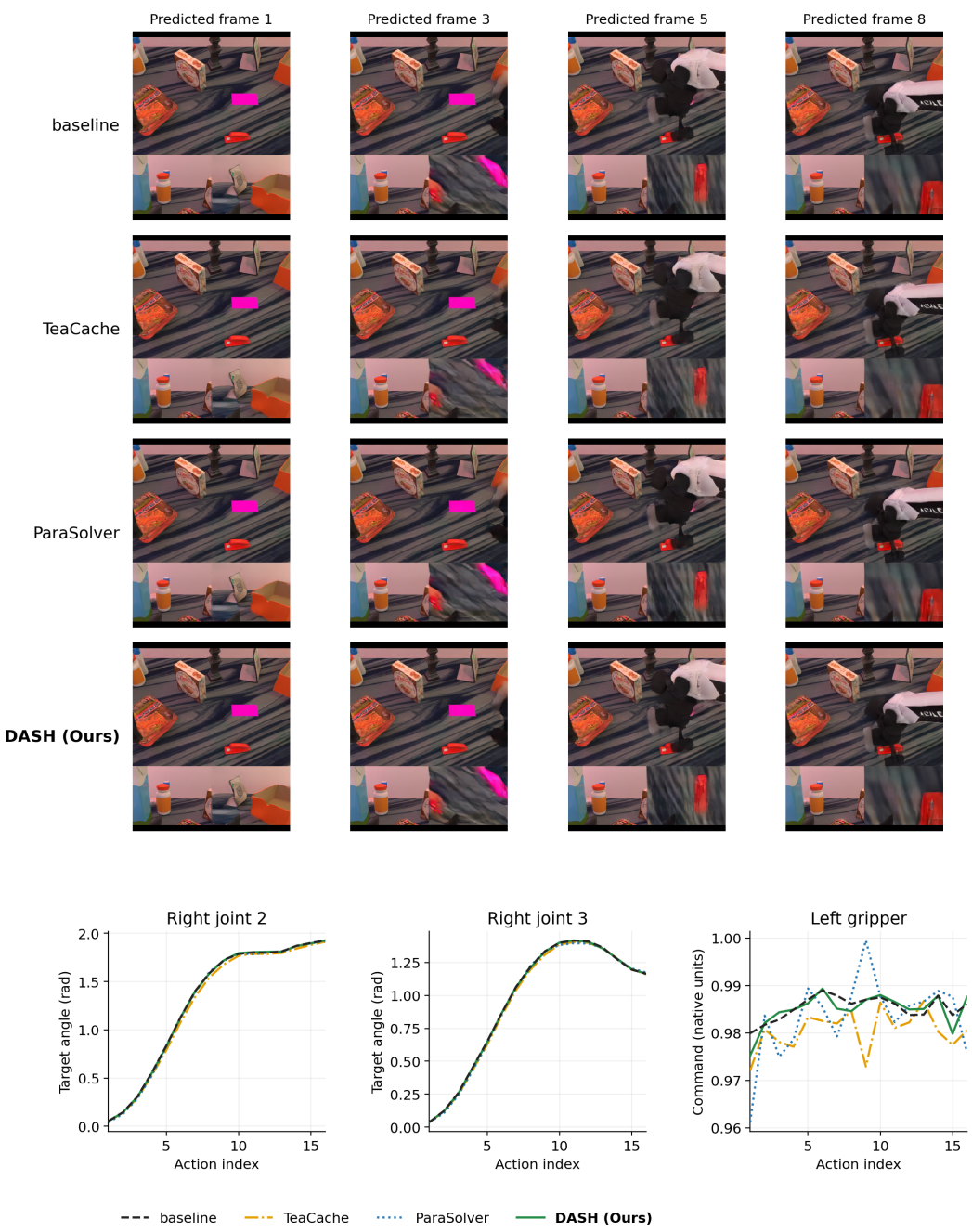

Figure 9: **Qualitative results** on the "`move stapler pad`".

H Norms and Notation

We follow the notation of Sections 3 and 4. For each sampling instance, the initial state, conditioning c, and sampling schedule are fixed. Unless stated otherwise, the analysis assumes exact arithmetic.

Let $\mathbf{X} = (\mathbf{V}, \mathbf{A})$ denote the joint state, with modality index $m \in \{v, a\}$. Write $f_i(\mathbf{X}) = f_\theta(\mathbf{X}, \boldsymbol{\tau}_i, c)$ and $\Delta\tau_{i-1}^m = \tau_i^m - \tau_{i-1}^m$. The sequential Euler map is

$$[G_i(\mathbf{X}, c)]^m = \mathbf{X}^m + \Delta\tau_{i-1}^m f_{i-1}^m(\mathbf{X}), \qquad i = 1, \dots, T. \tag{14}$$

The sequential reference satisfies $\mathbf{X}_0^* = \mathbf{X}_0$ and $\mathbf{X}_i^* = G_i(\mathbf{X}_{i-1}^*, c)$.

Let D_v be the number of scalar elements in the future-video latent and let $\mathbf{A} \in \mathbb{R}^{H_a \times D_a}$ be the action chunk. We use the modality norms

$$\|\mathbf{V}\|_v = \left(\frac{1}{D_v} \sum_{e=1}^{D_v} V_e^2 \right)^{1/2}, \qquad \|\mathbf{A}\|_a = \max_{\substack{1 \le t \le H_a \\ 1 \le d \le D_a}} \frac{|A_{t,d}|}{\sigma_d}, \tag{15}$$

where $\sigma_d > 0$ are fixed per-dimension scales shared by calibration and sampling. These are a scaled Euclidean norm and a weighted maximum norm, respectively.

For a joint quantity $\mathbf{Y} = (\mathbf{Y}^v, \mathbf{Y}^a)$, define its norm vector as

$$\overline{\mathbf{Y}} = \begin{pmatrix} \|\mathbf{Y}^v\|_v \\ \|\mathbf{Y}^a\|_a \end{pmatrix} \in \mathbb{R}^2_{\ge 0}. \tag{16}$$

All vector inequalities below are interpreted componentwise. The modality norms satisfy the triangle inequality and absolute homogeneity. Multiplication by an entrywise nonnegative matrix or row vector preserves componentwise inequalities.

I Fixed Point and Exact Recovery of Picard Iteration

Let $\Delta_i(\mathbf{X}, c) = G_i(\mathbf{X}, c) - \mathbf{X}$. For $\mathbf{Z} = (\mathbf{X}_1, \dots, \mathbf{X}_T)$, recall the Picard operator in Eq. (2),

$$[\mathcal{P}(\mathbf{Z})]_j = \mathbf{X}_0 + \sum_{i=1}^{j} \Delta_i(\mathbf{X}_{i-1}, c).$$

We write $\mathbf{Z}^{(k)} = (\mathbf{X}_1^{(k)}, \dots, \mathbf{X}_T^{(k)})$ and $\mathbf{Z}^* = (\mathbf{X}_1^*, \dots, \mathbf{X}_T^*)$.

Proposition 1 (Fixed point equals the sequential reference) *A trajectory $\mathbf{Z}$ satisfies $\mathbf{Z} = \mathcal{P}(\mathbf{Z})$ if and only if $\mathbf{Z} = \mathbf{Z}^*$. Thus, $\mathbf{Z}^*$ is the unique fixed point of $\mathcal{P}$.*

Proof. Summing the sequential increments gives

$$\mathbf{X}_j^* = \mathbf{X}_0 + \sum_{i=1}^{j} \Delta_i(\mathbf{X}_{i-1}^*, c),$$

so $\mathbf{Z}^*$ is a fixed point.

Conversely, suppose $\mathbf{Z} = \mathcal{P}(\mathbf{Z})$. Subtracting adjacent cumulative equations, with $[\mathcal{P}(\mathbf{Z})]_0 = \mathbf{X}_0$, yields

$$\mathbf{X}_j - \mathbf{X}_{j-1} = \Delta_j(\mathbf{X}_{j-1}, c), \qquad \mathbf{X}_j = G_j(\mathbf{X}_{j-1}, c). \tag{17}$$

The recurrence and fixed initial state uniquely determine $\mathbf{X}_j = \mathbf{X}_j^*$ for every $j = 1, \dots, T$. □

Proposition 2 (Exact recovery in at most T rounds) *Let $\mathbf{X}_1^{(0)}, \dots, \mathbf{X}_T^{(0)}$ be arbitrary initial estimates and keep $\mathbf{X}_0^{(k)} = \mathbf{X}_0$ for all k. Under the full-trajectory Picard iteration Eq. (3),*

$$\mathbf{X}_j^{(k)} = \mathbf{X}_j^* \qquad \textit{for } 0 \le j \le \min(k, T). \tag{18}$$

Consequently, $\mathbf{Z}^{(T)} = \mathbf{Z}^*$.

Proof. We use induction on k. At $k = 0$, the claim holds because $\mathbf{X}_0^{(0)} = \mathbf{X}_0^*$. Suppose it holds at round k and consider $1 \le j \le \min(k+1, T)$. Every predecessor used to update position j has index $i - 1 \le j - 1 \le k$, so $\mathbf{X}_{i-1}^{(k)} = \mathbf{X}_{i-1}^*$ for all $i \le j$. Therefore,

$$\begin{aligned} \mathbf{X}_j^{(k+1)} &= \mathbf{X}_0 + \sum_{i=1}^{j} \Delta_i(\mathbf{X}_{i-1}^{(k)}, c) \\ &= \mathbf{X}_0 + \sum_{i=1}^{j} \Delta_i(\mathbf{X}_{i-1}^*, c) = \mathbf{X}_j^*. \end{aligned}$$

The initial state remains fixed, completing the induction. □

J Residual Identity and Progress Guarantee

Consider a refinement round with fixed anchor b and window width $w = \min(p, T-b)$. The window states are updated by Eq. (6), and their local residuals are defined by Eq. (7).

Lemma 3 (Residual identity) *For every $i = b+1, \dots, b+w$ and $m \in \{v, a\}$,*

$$\mathbf{d}_i^{m,(k+1)} = \Delta\tau_{i-1}^m \left[f_{i-1}^m(\mathbf{X}_{i-1}^{(k)}) - f_{i-1}^m(\mathbf{X}_{i-1}^{(k+1)}) \right]. \tag{19}$$

In particular, $\mathbf{d}_{b+1}^{(k+1)} = \mathbf{0}$.

Proof. Subtracting adjacent Picard updates gives

$$\mathbf{X}_i^{m,(k+1)} = \mathbf{X}_{i-1}^{m,(k+1)} + \Delta\tau_{i-1}^m f_{i-1}^m(\mathbf{X}_{i-1}^{(k)}). \tag{20}$$

For $i = b+1$, the same equality follows from $\mathbf{X}_b^{(k+1)} = \mathbf{X}_b^{(k)}$. Subtracting $[G_i(\mathbf{X}_{i-1}^{(k+1)}, c)]^m$ and using Eq. (14) yields Eq. (19). At the first position, both velocity arguments equal the fixed anchor, so the residual vanishes. □

Lemma 4 (Progress and termination) *Let $T \geq 1$ and $p \geq 1$ be integers, $\varepsilon_A \geq 0$, and let the weights $\gamma_i^v, \gamma_i^a \geq 0$ in Eq. (9) be arbitrary. Suppose a round starts at $b < T$ with $C_b \leq \varepsilon_A b/T$. The stride selected by Eq. (12) satisfies $s \geq 1$, and acceptance preserves $C_{b+s} \leq \varepsilon_A(b+s)/T$. Starting from $b = 0$ and $C_0 = 0$, sampling reaches T within at most T rounds, and a completed run satisfies $C_T \leq \varepsilon_A$.*

Proof. The fixed anchor gives $\mathbf{X}_b^{(k+1)} - \mathbf{X}_b^{(k)} = \mathbf{0}$, and hence $\eta_{b+1}^{(k+1)} = 0$ by Eq. (9). Thus,

$$C_b + \eta_{b+1}^{(k+1)} = C_b \leq \varepsilon_A \frac{b}{T} \leq \varepsilon_A \frac{b+1}{T}.$$

The first candidate passes, so $s \geq 1$.

The acceptance rule stops before the first violating prefix. Consequently,

$$C_{b+s} = C_b + \sum_{i=b+1}^{b+s} \eta_i^{(k+1)} \leq \varepsilon_A \frac{b+s}{T}.$$

The budget invariant holds initially and is preserved after every round. Since the anchor advances by at least one position per round, at most T rounds are required. At termination, the invariant gives $C_T \leq \varepsilon_A$. □

K Terminal-Action Deviation Bound

We now relate the accumulated contribution score to the terminal action deviation from the sequential Euler reference.

Accepted trajectory. Let $\mathbf{X}_i$ denote the state at acceptance, which remains fixed thereafter. When position i is accepted, its predecessor is either the anchor or a position accepted in the same round. Thus, the updated predecessor is also final. For the accepting round $k+1$ of position i, define $\mathbf{d}_i = \mathbf{d}_i^{(k+1)}$ and $\boldsymbol{\delta}_{i-1} = \mathbf{X}_{i-1}^{(k+1)} - \mathbf{X}_{i-1}^{(k)}$. Then

$$\begin{aligned} \mathbf{X}_i &= G_i(\mathbf{X}_{i-1}, c) + \mathbf{d}_i, \\ \eta_i &= [\gamma_i^v, \gamma_i^a]\, \overline{\boldsymbol{\delta}}_{i-1}, \qquad i = 1, \dots, T. \end{aligned} \tag{21}$$

Since each position is accepted once, Eq. (13) gives $C_T = \sum_{i=1}^T \eta_i$.

Assumption 1 (Modality-wise sensitivity bounds) *For each $i = 1, \dots, T$, there exists an entry-wise nonnegative matrix $\mathbf{L}_i = (L_i^{mn})_{m,n\in\{v,a\}}$ such that*

$$\begin{aligned}\|f_{i-1}^m(\mathbf{X}) - f_{i-1}^m(\mathbf{X}')\|_m \le L_i^{mv}\|\mathbf{V}-\mathbf{V}'\|_v \\ + L_i^{ma}\|\mathbf{A}-\mathbf{A}'\|_a, \qquad m \in \{v,a\},\end{aligned} \tag{22}$$

for any pair of states among the refinement states at position $i-1$, the accepted state, and the sequential reference state. The calibrated matrices satisfy $\widehat{\mathbf{L}}_i \ge \mathbf{L}_i$ componentwise.

The off-diagonal entries capture cross-modal sensitivity: L_i^{av} bounds the effect of video-state changes on the action velocity, and L_i^{va} the converse.

Lemma 5 (Local residual bound) *Under Assumption 1, every accepted position satisfies*

$$\overline{\mathbf{d}}_i \le \mathbf{K}_i \overline{\boldsymbol{\delta}}_{i-1}, \qquad i = 1, \dots, T, \tag{23}$$

where $\mathbf{K}_i$ is defined in Eq. (10).

Proof. Apply Lemma 3 in the accepting round of position i. Absolute homogeneity and Assumption 1 give

$$\begin{aligned}\|\mathbf{d}_i^m\|_m &= |\Delta\tau_{i-1}^m|\,\|f_{i-1}^m(\mathbf{X}_{i-1}^{(k)}) - f_{i-1}^m(\mathbf{X}_{i-1}^{(k+1)})\|_m \\ &\le |\Delta\tau_{i-1}^m| \sum_{n\in\{v,a\}} L_i^{mn}\|\boldsymbol{\delta}_{i-1}^n\|_n \\ &\le [\mathbf{K}_i\overline{\boldsymbol{\delta}}_{i-1}]_m.\end{aligned}$$

The last inequality uses $\widehat{\mathbf{L}}_i \ge \mathbf{L}_i$. Collecting the two modality bounds proves the claim. □

Lemma 6 (One-step error propagation) *Let $\mathbf{e}_i = \mathbf{X}_i - \mathbf{X}_i^*$. Under Assumption 1, $\mathbf{e}_0 = \mathbf{0}$ and*

$$\overline{\mathbf{e}}_i \le (\mathbf{I}_2 + \mathbf{K}_i)\overline{\mathbf{e}}_{i-1} + \overline{\mathbf{d}}_i, \qquad i = 1, \dots, T. \tag{24}$$

Proof. The initial states coincide, so $\mathbf{e}_0 = \mathbf{0}$. Subtracting the sequential reference update from Eq. (21) gives

$$\mathbf{e}_i^m = \mathbf{e}_{i-1}^m + \Delta\tau_{i-1}^m\left[f_{i-1}^m(\mathbf{X}_{i-1}) - f_{i-1}^m(\mathbf{X}_{i-1}^*)\right] + \mathbf{d}_i^m. \tag{25}$$

Taking modality norms and applying the triangle inequality and Assumption 1 yields

$$\|\mathbf{e}_i^m\|_m \le \|\mathbf{e}_{i-1}^m\|_m + [\mathbf{K}_i\overline{\mathbf{e}}_{i-1}]_m + \|\mathbf{d}_i^m\|_m.$$

The two component inequalities give Eq. (24). □

Theorem 7 (Terminal-action deviation) *Under Assumption 1, a completed run of DASH satisfies*

$$\|\mathbf{A}_T - \mathbf{A}_T^*\|_a \le \sum_{i=1}^{T}\eta_i = C_T \le \varepsilon_A, \tag{26}$$

where $\mathbf{A}_T^*$ is the action chunk produced by sequential Euler sampling with the same initial state, conditioning, and schedule.

Proof. Starting from $\overline{\mathbf{e}}_0 = \mathbf{0}$, repeated application of Lemma 6 gives

$$\overline{\mathbf{e}}_j \le \sum_{i=1}^{j}\left[\prod_{\ell=i+1}^{j}(\mathbf{I}_2 + \mathbf{K}_\ell)\right]\overline{\mathbf{d}}_i, \qquad j = 1, \dots, T. \tag{27}$$

The product is ordered as $(\mathbf{I}_2 + \mathbf{K}_j)\cdots(\mathbf{I}_2 + \mathbf{K}_{i+1})$, with the empty product equal to $\mathbf{I}_2$. Each substitution preserves the inequality because all propagation matrices are entrywise nonnegative.

Setting $j = T$, selecting the action component, and applying Lemma 5 yields

$$\begin{aligned}\|\mathbf{A}_T - \mathbf{A}_T^*\|_a &\le \sum_{i=1}^{T} [0,1] \left[\prod_{\ell=i+1}^{T} (\mathbf{I}_2 + \mathbf{K}_\ell)\right] \mathbf{K}_i \overline{\boldsymbol{\delta}}_{i-1} \\ &= \sum_{i=1}^{T} [\gamma_i^v, \gamma_i^a]\, \overline{\boldsymbol{\delta}}_{i-1} \\ &= \sum_{i=1}^{T} \eta_i = C_T \le \varepsilon_A.\end{aligned} \tag{28}$$

The equalities use the weight definition Eq. (11) and the scores in Eq. (21). The final inequality follows from Lemma 4. □

Scope of the guarantee. Theorem 7 bounds the action chunk relative to the discrete sequential Euler reference with the same initial state, conditioning and schedule; video inconsistencies enter only through their propagated effect on the action via the off-diagonal entries of $\mathbf{K}_\ell$, and no separate video budget or closed-loop success guarantee is implied. The calibrated sensitivities enter the analysis solely through Assumption 1: termination and $C_T \le \varepsilon_A$ (Lemma 4) hold for any nonnegative weights, whereas the inequality $\|\mathbf{A}_T - \mathbf{A}_T^*\|_a \le C_T$ requires $\widehat{\mathbf{L}}_i \ge \mathbf{L}_i$ on the visited states. Since $\widehat{\mathbf{L}}_i$ is estimated from finitely many calibration pairs (Section L), this domination is not certified, and C_T is accordingly used as an empirical indicator of the deviation; finite-precision effects add the rounding term in Eq. (32), which is not controlled by ε_A.

L Sensitivity Calibration and Numerical Considerations

Empirical sensitivity calibration. We estimate the modality-wise sensitivities from an offline calibration set. For a pair of states $\mathbf{X}$ and $\mathbf{X}'$ with the same conditioning and noise levels, differing only in modality n, define

$$\rho_i^{mn}(\mathbf{X}, \mathbf{X}') = \frac{\|f_{i-1}^m(\mathbf{X}) - f_{i-1}^m(\mathbf{X}')\|_m}{\|\mathbf{X}^n - \mathbf{X}'^n\|_n}, \qquad m, n \in \{v, a\}, \tag{29}$$

for pairs with a nonzero denominator. The entries of $\widehat{\mathbf{L}}_i$ are estimated by aggregating these ratios over calibration pairs. The model parameters, sampling schedule, and modality norms remain fixed during calibration and subsequent use of the weights.

Precomputation of the weights. Given $\widehat{\mathbf{L}}_i$ and the schedule, we construct $\mathbf{K}_i$ using Eq. (10). Define $\mathbf{r}_i = [0,1](\mathbf{I}_2 + \mathbf{K}_T) \cdots (\mathbf{I}_2 + \mathbf{K}_{i+1})$, with an empty product at $i = T$. The weights are computed by the backward recursion

$$\begin{aligned}\mathbf{r}_T &= [0,1], \\ \mathbf{r}_{i-1} &= \mathbf{r}_i(\mathbf{I}_2 + \mathbf{K}_i), \quad i = T, \dots, 2, \\ [\gamma_i^v, \gamma_i^a] &= \mathbf{r}_i \mathbf{K}_i, \qquad\quad i = 1, \dots, T.\end{aligned} \tag{30}$$

This requires $O(T)$ products of a two-dimensional row vector and a 2×2 matrix. The precomputed weights are reused during sampling, so scoring requires no additional model evaluations.

Finite-precision considerations. Regard the stored states as real-valued quantities and retain the mathematical residual definition $\mathbf{d}_i = \mathbf{X}_i - G_i(\mathbf{X}_{i-1}, c)$. At the accepting round of position i, write

$$\mathbf{d}_i^m = \Delta\tau_{i-1}^m \left[f_{i-1}^m(\mathbf{X}_{i-1}^{(k)}) - f_{i-1}^m(\mathbf{X}_{i-1}^{(k+1)})\right] + \boldsymbol{\rho}_i^m. \tag{31}$$

The perturbation $\boldsymbol{\rho}_i$ collects the rounding errors of the implemented update relative to the exact residual identity. If Assumption 1 holds and $\overline{\boldsymbol{\rho}}_i \le \overline{\boldsymbol{\varrho}}_i$ for valid nonnegative perturbation bounds, then $\overline{\mathbf{d}}_i \le \mathbf{K}_i \overline{\boldsymbol{\delta}}_{i-1} + \overline{\boldsymbol{\varrho}}_i$. The propagation argument consequently gives

$$\|\mathbf{A}_T - \mathbf{A}_T^*\|_a \le C_T + \sum_{i=1}^{T} \mathbf{r}_i \overline{\boldsymbol{\varrho}}_i. \tag{32}$$

where C_T is the sum of scores evaluated on the stored states.

M EXTENSION TO DIFFUSION-BASED WAMS

Consider a deterministic single-step sampling map

$$[G_i(\mathbf{X}, c)]^m = a_i^m \mathbf{X}^m + b_i^m h_{i-1}^m(\mathbf{X}), \qquad m \in \{v, a\}, \tag{33}$$

where a_i^m and b_i^m are schedule-dependent scalars. The prediction h_{i-1}^m depends on the joint state, the noise levels at position $i-1$, and the fixed conditioning, with the latter arguments suppressed. This form includes deterministic DDIM updates Song et al. (2020a) and Euler discretizations of probability-flow ODEs for standard VP and VE diffusion processes Song et al. (2020b).

Define $\Delta_i(\mathbf{X}, c) = G_i(\mathbf{X}, c) - \mathbf{X}$. The corresponding windowed Picard update is

$$\mathbf{X}_{b+j}^{(k+1)} = \mathbf{X}_b + \sum_{\ell=b+1}^{b+j} \Delta_\ell(\mathbf{X}_{\ell-1}^{(k)}, c), \qquad j = 1, \dots, w, \tag{34}$$

with the anchor fixed. The full-trajectory fixed-point and exact-recovery results in Propositions 1 and 2 remain valid because their proofs depend only on G_i and its increment.

Let $\widehat{\mathbf{L}}_i$ denote the modality-wise sensitivity estimates for h_{i-1}. Define

$$\begin{aligned} \mathbf{K}_i &= \operatorname{diag}(|b_i^v|, |b_i^a|)\widehat{\mathbf{L}}_i, \\ \mathbf{R}_i &= \operatorname{diag}(|a_i^v - 1|, |a_i^a - 1|) + \mathbf{K}_i, \\ \mathbf{P}_i &= \operatorname{diag}(|a_i^v|, |a_i^a|) + \mathbf{K}_i. \end{aligned} \tag{35}$$

The matrices $\mathbf{R}_i$ and $\mathbf{P}_i$ control local residuals and downstream error propagation, respectively. The terminal-action weights are

$$[\gamma_i^v, \gamma_i^a] = [0, 1]\mathbf{P}_T \cdots \mathbf{P}_{i+1}\mathbf{R}_i, \tag{36}$$

where the product over subsequent steps is the identity at $i = T$. At acceptance, the score is $\eta_i = [\gamma_i^v, \gamma_i^a]\overline{\boldsymbol{\delta}}_{i-1}$.

Proposition 8 (Terminal-action bound for diffusion sampling) *Suppose that $\widehat{\mathbf{L}}_i$ satisfies Assumption 1 with f_{i-1} replaced by h_{i-1}. Consider a completed run using Eq. (34), the weights in Eq. (36), and the cumulative acceptance rule Eq. (12). Then*

$$\|\mathbf{A}_T - \mathbf{A}_T^*\|_a \le \sum_{i=1}^{T} \eta_i = C_T \le \varepsilon_A, \tag{37}$$

where $\mathbf{X}_i^* = G_i(\mathbf{X}_{i-1}^*, c)$ is the sequential reference with the same initial state, conditioning, and schedule.

Proof. Subtracting adjacent Picard updates gives $\mathbf{X}_i^{(k+1)} - \mathbf{X}_{i-1}^{(k+1)} = \Delta_i(\mathbf{X}_{i-1}^{(k)}, c)$. Therefore, the residual satisfies

$$\mathbf{d}_i^{(k+1)} = \Delta_i(\mathbf{X}_{i-1}^{(k)}, c) - \Delta_i(\mathbf{X}_{i-1}^{(k+1)}, c).$$

Expanding the increment yields

$$\begin{aligned} \mathbf{d}_i^{m,(k+1)} = (a_i^m - 1)\left(\mathbf{X}_{i-1}^{m,(k)} - \mathbf{X}_{i-1}^{m,(k+1)}\right) \\ + b_i^m\left[h_{i-1}^m(\mathbf{X}_{i-1}^{(k)}) - h_{i-1}^m(\mathbf{X}_{i-1}^{(k+1)})\right]. \end{aligned} \tag{38}$$

Taking modality norms and applying the sensitivity bounds gives $\overline{\mathbf{d}}_i \le \mathbf{R}_i\overline{\boldsymbol{\delta}}_{i-1}$, which replaces Lemma 5. Likewise, for $\mathbf{e}_i = \mathbf{X}_i - \mathbf{X}_i^*$, subtracting the sequential update from the accepted update gives $\mathbf{e}_i^m = a_i^m\mathbf{e}_{i-1}^m + b_i^m[h_{i-1}^m(\mathbf{X}_{i-1}) - h_{i-1}^m(\mathbf{X}_{i-1}^*)] + \mathbf{d}_i^m$, hence $\overline{\mathbf{e}}_i \le \mathbf{P}_i\overline{\mathbf{e}}_{i-1} + \overline{\mathbf{d}}_i$, which replaces Lemma 6. All matrices are entrywise nonnegative, so unrolling as in the proof of Theorem 7 yields $\|\mathbf{A}_T - \mathbf{A}_T^*\|_a \le \sum_{i=1}^T \eta_i$, and Lemma 4 gives $\sum_i \eta_i = C_T \le \varepsilon_A$. □

Flow-based Euler sampling is recovered by setting $a_i^m = 1$, $b_i^m = \Delta\tau_{i-1}^m$, and $h_{i-1} = f_{i-1}$. Then $\mathbf{R}_i = \mathbf{K}_i$ and $\mathbf{P}_i = \mathbf{I}_2 + \mathbf{K}_i$, recovering Eq. (11).