QANTARA: BRIDGE-FLOW TRAINING FOR MULTI-PARADIGM JEPA INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Joint-Embedding Predictive Architectures (JEPAs) underpin a growing family of latent world models for control from raw pixels. Published JEPA world models for pixel-based control (PLDM, DINO-WM, LeWM, V-JEPA-2-AC) each commit at training time to a single inference paradigm, trajectory optimisation in a learned dynamics model, so a checkpoint trained to plan cannot be read off as a policy without retraining. Deferring that choice to inference would let the constraints that bind at deployment, per-step latency and whether a goal observation can be supplied, pick the path. We present **Qantara**, an end-to-end JEPA of ~ 21M parameters whose joint training objective pairs a Brownian-bridge interpolant between consecutive clean latents on the state axis with noise-to-data flow matching on the action axis. One checkpoint serves three inference paradigms without retraining: latent planning, behaviour-cloning action sampling, and a video–inverse composition that predicts the next latent without action conditioning, then recovers the action bridging the two. Training samples the two noise levels along the edges of the (action-noise, state-noise) square, one edge per inference dispatch, so each path is trained where it is queried. Replacing that sampler with uniform sampling over the square at matched compute drops Push-T planning from 90.1 to 53.3 success rate (SR). On the LeWM control suite at a matched 10-epoch budget, Qantara averages 91.2 SR over the four environments and reaches 93.7 on OGBench-Cube, $+7.7$ over DINO-WM and $+17.0$ over LeWM, each measured against the stronger of that baseline's published score and our retrain of it. From the same weights the two goal-blind paths, which receive no goal image, reach 82.1–83.2 SR on Push-T and 70.8–72.8 on Cube at ~ 15–$65\times$ lower inference cost, and fall to near-noise on Reacher, whose released training corpus comes from a random policy.

1 INTRODUCTION

A robot deployed for visuomotor control faces inference-time constraints (per-step latency, the size of the action search space, whether a goal observation can be supplied) that the training run cannot anticipate. The same learned predictor is useful in different ways under different constraints: rolled out as a forward dynamics simulator under a planner when the search budget allows, or read off as a behaviour-cloning policy when latency is tight. Published JEPA world models trained from pixels for control commit to one of these inference paradigms at training time, and a checkpoint trained for one cannot serve the others without retraining.

Two clusters of prior work tile the relevant design space. JEPA world models commit to a single inference paradigm, trajectory optimisation: PLDM (Sobal et al., 2025), DINO-WM (Zhou et al., 2024), and LeWM (Maes et al., 2026; Balestriero & LeCun, 2025) train sub-billion action-conditional latent dynamics and plan by minimising a goal-conditioned latent distance, and V-JEPA-2-AC (Assran et al., 2025) post-trains a 300M action-conditional predictor on a frozen 1B video JEPA and plans the same way, by cross-entropy-method energy minimisation in a receding-horizon loop. Each of them needs a goal image and a rollout budget at every control step. Multi-paradigm checkpoints exist in a second cluster that predicts the future observation rather than a latent dynamics of its own, at up to billions of parameters: PAD, UVA, UWM, DUST, and Cosmos Policy (Guo et al., 2024; Li et al., 2025; Zhu et al., 2025; Won et al., 2025; Kim et al., 2026) denoise observations and actions jointly and serve forward dynamics, inverse dynamics, and behaviour policy from one set of weights. The

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.

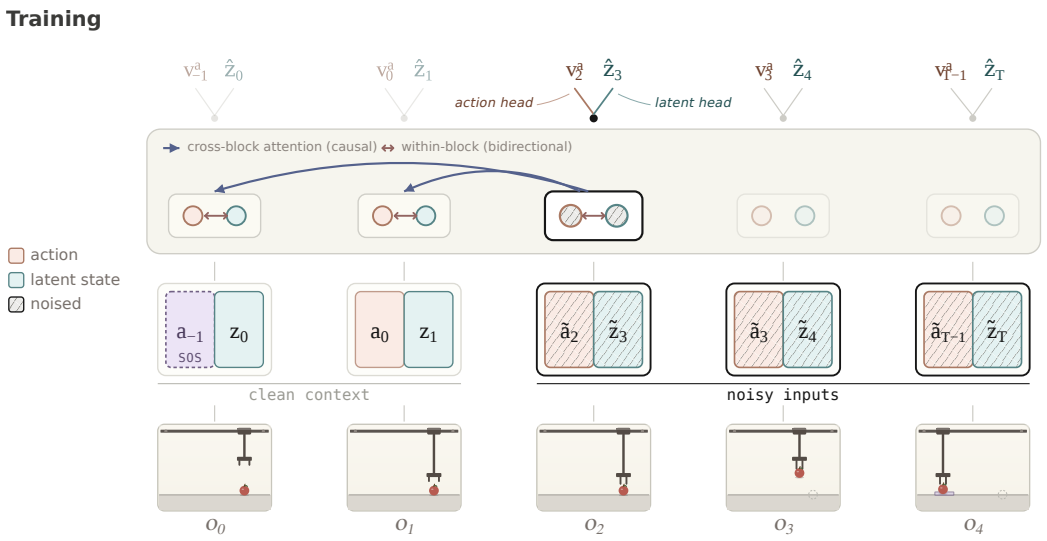

Figure 1: **Qantara training scheme.** Block-causal predictor on (a, z) tokens; a clean prefix conditions the noisy continuation, arrows mark where attention is causal (across blocks) versus bidirectional (within a block), and per block the action head emits an action velocity v^a while the state head emits a latent residual that yields $\hat{z}$.

(sub-billion JEPA, multi-paradigm) cell is sparsely occupied: Nguyen et al. (2026) reach it with a second training stage fitted on frozen LeWM latents.

The obstacle is that planning, behaviour cloning, and inverse dynamics each query the predictor at a different combination of clean and noisy inputs on the action and state axes, so an objective tuned for one combination under-trains the others. The open question is whether a single sub-billion JEPA can support all three at deployment without a cost to any of them.

We present **Qantara**, an end-to-end pixel JEPA world model whose joint training objective pairs a Brownian-bridge interpolant on the state axis between consecutive clean latents with noise-to-data flow matching on the action axis (Lyu et al., 2024; Liu et al., 2023; Lipman et al., 2023; Albergo et al., 2023). Per-token noise levels (τ^a, τ^z) are sampled along the four edges of the $[0, 1]^2$ noise square (one edge per inference dispatch) plus a fifth diagonal mode that co-regularises the shared trunk. The same training objective populates the four corners that planning, behaviour cloning, and inverse dynamics query at deployment. From a single $\sim$21M-parameter checkpoint, Qantara serves three inference paradigms without retraining: goal-conditioned latent planning, behaviour-cloning sampling, and a video–inverse composition that first predicts the next latent without action conditioning and then recovers the action that bridges the two.

We make three contributions. (1) We present Qantara, a sub-billion JEPA world model that defers the planning-vs-imitation choice to inference (§2). (2) On the LeWM-suite (Maes et al., 2026) at a matched 10-epoch budget, Qantara averages 91.2 SR and reaches 93.7 on OGBench-Cube, $+7.7$ over DINO-WM as re-evaluated by Maes et al. (2026); from the same checkpoint, the behaviour-cloning and video–inverse dispatches read off at $\sim$ 15–65$\times$ lower inference cost than Qantara's CEM planner (§3.2, §3.3). (3) We identify a corner co-regularisation effect: dropping any of the `policy`, `inverse`, or `joint` modes destabilises Push-T training on all three inference paths, even the planner, whose edge shares at most one corner with theirs (§3.4).

2 Method

Qantara is a joint model of latents and their generating actions, trained offline from trajectories $\{(o_t, a_t)\}_{t=0}^{T}$ of pixel observations o_t and actions $a_t \in \mathbb{R}^{d_a}$, without rewards or task labels. Figure 1 sketches the predictor and Figure 2 the sampling design (right).

2.1 LEARNING THE WORLD-ACTION MODEL

Model architecture. Qantara consists of an encoder E_ϕ that maps each frame o_t to a d-dimensional latent $z_t = E_\phi(o_t)$, and a predictor f_θ that models the joint distribution p_θ of actions and future latents along a trajectory. We organise the trajectory into a sequence of *blocks* indexed by $t \in \{0, \dots, T\}$, where each block bundles latent z_t with the action that produced it; for block 0 a learnable start-of-sequence token stands in for the non-existent prior action. The predictor factors p_θ autoregressively across blocks but jointly within each block: at step t, the pair $(a_t,\ z_{t+1})$ of action and resulting latent (which together populate block $t+1$) is predicted jointly given the prefix $(z_{\le t},\ a_{<t})$. For any clean (un-noised) prefix length $t' \in \{0, \dots, T-1\}$, the chain rule then gives

$$p_\theta(a_{t':T-1},\ z_{t'+1:T} \mid z_{\le t'},\ a_{<t'}) \ = \ \prod_{t=t'}^{T-1} p_\theta(a_t,\ z_{t+1} \mid z_{\le t},\ a_{<t}), \tag{1}$$

and f_θ parameterises each per-block conditional with a transformer whose attention is block-causal across blocks (realising Eq. 1) and bidirectional within a block, so the action and state tokens of a block are denoised jointly under the bridge-flow objective detailed below.

Joint bridge-flow matching. At training time the predictor reads a sequence split into a random observed context and a noised continuation it must denoise:

$$\big[\ \underbrace{a_{-1}\, z_0 \quad a_0\, z_1 \quad \cdots \quad a_{t-1}\, z_t}_{\text{context (observed)}} \ \ \underbrace{\tilde{a}_t\, \tilde{z}_{t+1} \quad \cdots \quad \tilde{a}_{T-1}\, \tilde{z}_T}_{\text{noised (to denoise)}}\ \big]. \tag{2}$$

Each action and state token carries its own scalar $\tau \in [0, 1]$, indexed by the token's time (τ^a_t for action token a_t, τ^z_t for state token z_t); $\tau = 1$ is the clean endpoint and $\tau = 0$ the source endpoint. Each τ is fed to the predictor as a sinusoidal time-encoding through per-token AdaLN modulation (Peebles & Xie, 2022; Perez et al., 2017). The two axes use different interpolants because inference exposes a different anchor at $\tau = 0$ on each. Actions have no anchor: at every rollout step the planner queries the predictor for an action it does not yet possess, so the $\tau^a_t = 0$ source must be task-agnostic and we adopt the standard noise-to-data flow-matching interpolant (Lipman et al., 2023; Liu et al., 2022; Albergo et al., 2023; Gao et al., 2026),

$$\tilde{a}_t \ = \ \tau^a_t\, a_t \ + \ (1 - \tau^a_t)\, \varepsilon_a, \qquad \varepsilon_a \sim \mathcal{N}(0, I). \tag{3}$$

The latent, by contrast, always has the previous clean z_t as a free anchor; pairing it with the unknown target z_{t+1} as endpoints of a Brownian-bridge interpolant (Albergo et al., 2023; Liu et al., 2023; Lyu et al., 2024) gives

$$\tilde{z}_{t+1} \ = \ (1 - \tau^z_{t+1})\, z_t \ + \ \tau^z_{t+1}\, z_{t+1} \ + \ \gamma \sqrt{\tau^z_{t+1}(1 - \tau^z_{t+1})}\, \varepsilon_z, \qquad \varepsilon_z \sim \mathcal{N}(0, I), \tag{4}$$

whose $\tau^z_{t+1} = 0$ source is exactly the previous latent the planner feeds in at deployment. Because this source already matches the deployment input, the model does not have to learn a map from arbitrary Gaussian noise back to the latent manifold; it only has to model the on-manifold transition $z_{t+1} - z_t$ between consecutive rollout steps.

Training objective. For each noisy target block, the transformer produces a hidden vector per token, contextualised over earlier blocks (block-causal attention) and the same-block partner (within-block bidirectional attention):

$$(h^a_t,\ h^z_{t+1}) \ = \ f_\theta\big(\tilde{a}_{\le t},\ \tilde{z}_{\le t+1}\big). \tag{5}$$

The state head reads h^z_{t+1} and emits a residual on the previous clean latent; the action head reads h^a_t and emits an action velocity,

$$\hat{z}_{t+1} \ = \ z_t + \mathrm{head}_z\big(h^z_{t+1}\big), \qquad v^a_t \ = \ \mathrm{head}_a\big(h^a_t\big). \tag{6}$$

The state-head residual form (Eq. 6) pairs with zero-initialised final-layer weights, making the predictor the identity at initialisation ($\hat{z}_{t+1} = z_t$); we ablate the residual choice in §3.4. The state-axis bridge loss regresses the clean endpoint and the action-axis flow-matching loss the velocity,

$$\mathcal{L}^z_{\mathrm{B}} = \mathbb{E}\,\|\hat{z}_{t+1} - z_{t+1}\|_2^2, \qquad \mathcal{L}^a_{\mathrm{FM}} = \mathbb{E}\,\|v^a_t - (a_t - \varepsilon_a)\|_2^2, \tag{7}$$

where the expectation is over data transitions (z_t, a_t, z_{t+1}), flow times (τ^a_t, τ^z_{t+1}) sampled per §2.3, and independent Gaussian noises $\varepsilon_a, \varepsilon_z \sim \mathcal{N}(0, I)$. Optimising $\mathcal{L}^z_{\mathrm{B}}$ alone admits a collapse: if the encoder outputs the same latent for every frame, the bridge target is trivially matched and the action head sees no state information to condition on. We add SIGReg (Balestriero & LeCun, 2025), the Sketched-Isotropic-Gaussian regulariser, applied to the batch of clean latents $\{z_0, \ldots, z_T\}$ to drive their distribution toward an isotropic Gaussian; the total objective is

$$\mathcal{L} \;=\; \lambda_z\, \mathcal{L}^z_{\mathrm{B}} \;+\; \lambda_a\, \mathcal{L}^a_{\mathrm{FM}} \;+\; \lambda_{\mathrm{SIG}}\, \mathrm{SIGReg}\big(\{z_0, \ldots, z_T\}\big), \tag{8}$$

with $\lambda_z = 3$, $\lambda_a = 1$, and $\lambda_{\mathrm{SIG}} = 0.09$. All components (encoder, predictor, and the action and state heads) are trained jointly from scratch. We adopt LeWM's no-heuristics training recipe (Maes et al., 2026): no stop-gradient on encoder outputs, no EMA target encoder, and no pretrained representations, with SIGReg as the sole anti-collapse mechanism.

2.2 Inference: three paradigms from a single checkpoint

Every control step has the same setup: the encoder produces a clean latent that extends the context prefix $(z_{\le t},\, a_{<t})$, and we need to produce the next action a_t to execute. Because a_t lives in the joint target block (a_t, z_{t+1}), the same checkpoint admits three qualitatively different ways to obtain it, depending on whether an encoded goal vector is supplied at inference.

Geometrically, each query fixes one of (τ^a_t, τ^z_{t+1}) at 0 or 1 and sweeps the other from 0 to 1, tracing a single side of the $[0,1]^2$ noise-time square (Figure 2). We call these four sides the *edges* of the square and refer to each dispatch by the edge it traverses (`policy`, `forward`, `video`, or `inverse`); training concentrates capacity along them (§2.3).

Behaviour-cloning sampling. We sample a_t directly from the BC conditional $p_\theta(a_t \mid z_{0:t},\, \tilde{z}_{t+1} = z_t)$ along the `policy` edge ($\tau^z = 0$, $\tau^a\colon 0 \to 1$), where the predictor acts as a state-conditional behaviour policy. With the state slot of block $t{+}1$ held at z_t, the action slot starts as Gaussian noise and K Euler steps on the velocity head sweep it to a clean sample,

$$\tilde{a}^{(k+1)} \;=\; \tilde{a}^{(k)} \;+\; \tfrac{1}{K}\, v^a\big(\tilde{a}^{(k)},\, \tau^a_t = k/K\big), \qquad \tilde{a}^{(0)} \sim \mathcal{N}(0, I), \quad k = 0, \ldots, K{-}1, \tag{9}$$

giving $a_t = \tilde{a}^{(K)}$. At deployment we run $K = 2$ Euler steps over a context window of length $T = 4$ matching training.

Video–inverse composition. The BC conditional $p_\theta(a_t \mid z_{0:t})$ also factorises through the next latent, as in UniPi's video-as-policy paradigm (Du et al., 2023); here the "video" factor is the predicted JEPA latent $\hat{z}_{t+1}$ (not a pixel frame), and no decoder runs at inference. By the chain rule,

$$\underbrace{p_\theta(a_t \mid z_{0:t})}_{\text{BC dispatch}} \;=\; \int \underbrace{p_\theta(z_{t+1} \mid z_{0:t})}_{\texttt{video}\ \text{edge}}\, \underbrace{p_\theta(a_t \mid z_{0:t},\, z_{t+1})}_{\texttt{inverse}\ \text{edge}}\, \mathrm{d}z_{t+1}, \tag{10}$$

which we approximate by point-evaluating at the conditional mean $\hat{z}_{t+1}$ of the video factor. The two factors traverse two edges of the (τ^a, τ^z) square. First, the `video` edge ($\tau^a = 0, \tau^z\colon 0 \to 1$): with the action slot filled by pure Gaussian noise (carrying no information about a_t), the predictor reduces to an action-free latent dynamics model and reads off $\hat{z}_{t+1} \approx \mathbb{E}[z_{t+1} \mid z_{0:t},\, \tilde{z}_{t+1} = z_t]$, the action-marginal next latent under the training-time action prior. We follow the JEPA world model convention in calling this the `video` edge (Zhou et al., 2024; Assran et al., 2025; Maes et al., 2026); we operate in latent space and never decode pixels. Second, the `inverse` edge ($\tau^z = 1, \tau^a\colon 0 \to 1$): with both transition endpoints clean (z_t on the prefix and $\hat{z}_{t+1}$ on the state slot, action slot from Gaussian noise), K Euler steps recover $a_t \sim p_\theta(a_t \mid z_{0:t},\, \tilde{z}_{t+1} = \hat{z}_{t+1})$, the inverse-dynamics action that bridges z_t to $\hat{z}_{t+1}$. The composition gains 1–2 SR over BC sampling on Push-T and Cube, and ties within ± 1.5 SR on Two-Room and Reacher (§3.3).

Latent planning. Given a goal observation o_g encoded to $z_g = E_\phi(o_g)$, we search for an action sequence $a^*_{0:H-1}$ minimising the terminal cost $\mathcal{C}(\hat{z}_H) = \|\hat{z}_H - z_g\|_2^2$, following LeWM (Maes et al., 2026). Each rollout step queries the `forward` edge ($\tau^a = 1, \tau^z\colon 0 \to 1$): with the candidate clean action a_t on the action slot and the previous latent z_t on the state slot, the predictor acts as a forward dynamics model and reads off $\hat{z}_{t+1} \approx \mathbb{E}[z_{t+1} \mid z_{0:t},\, a_t,\, \tilde{z}_{t+1} = z_t]$ from the residual head

(Eq. 6). We iterate the predictor recursion for K denoising steps along $\tau^z : 0 \to 1$ to sharpen the $\hat{z}_{t+1}$ estimate ($K = 4$; flat sweep over $\{1, 2, 4, 8\}$ in App. G). The search uses the Cross-Entropy Method (Rubinstein & Kroese, 2004) with N Gaussian-proposal candidates and elite-set refit; rollouts accumulate prediction error with horizon, so we apply receding-horizon Model Predictive Control, executing the elite-mean plan before replanning (App. F).

2.3 Edge-aligned (τ^a, τ^z) sampling

Sampling (τ^a_t, τ^z_{t+1}) uniformly on $[0, 1]^2$ during training would spend most capacity on the interior, where no inference dispatch ever queries. We concentrate training on the four edges instead, plus a fifth `joint` mode along the diagonal ($\tau^a = \tau^z$, sweeping $0 \to 1$ together). The joint diagonal is never queried at inference, but removing it destabilises training (§3.4).

Each batch row is replicated across all five modes (Figure 2); within each mode the swept axis follows a per-block monotone-decreasing τ-chain (Xie et al., 2024; Liu et al., 2026) and a random clean prefix keeps every cold-start rollout horizon in-distribution (Po et al., 2025; Zhu et al., 2026), both detailed in App. C.

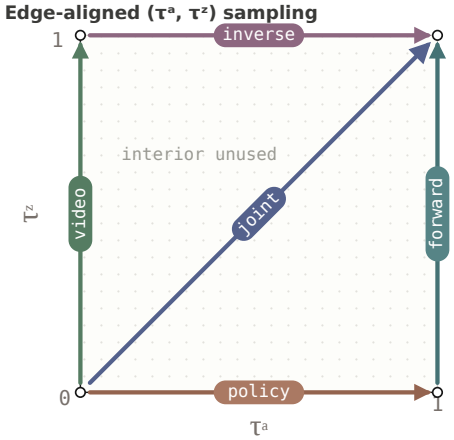

Figure 2: **Edge-aligned (τ^a, τ^z) sampling.** Per-token (τ^a, τ^z) are drawn under five 1D modes: the four edges of $[0, 1]^2$ (`forward`, `policy`, `inverse`, `video`) and the `joint` diagonal; the 2D interior is never sampled, so capacity stays on the regions queried at inference.

3 Experiments

3.1 Setup

We evaluate on the *LeWM-suite* (Maes et al., 2026), the four-environment subset on which LeWM places PLDM, DINO-WM, and itself in a single end-to-end pixel-JEPA pipeline. The subset supplies apples-to-apples baseline numbers, and its four envs cross 2D vs. 3D observations, navigation vs. manipulation vs. motor control, and four classes of data-collection oracle (noisy heuristic, human demonstration, scripted expert, uniform random), so no single data distribution dominates the suite average. The only broader pixel JEPA-world-model benchmark, DINO-WM's six-environment evaluation (Zhou et al., 2024), adds envs either in categories this suite already covers or in a deformable-dynamics class separate from the multi-paradigm question. The four envs are **Two-Room** (Sobal et al., 2025) (2D navigation), **Push-T** (Chi et al., 2023) on the DINO-WM-released expert dataset (Zhou et al., 2024) (contact-rich 2D manipulation), **OGBench-Cube** (Park et al., 2024) (3D pick-and-place; OGBench's single-cube `cube-single` task, App. A), and **Reacher-Hard** from the DeepMind Control suite (Tassa et al., 2018) (precision motor control); per-env trajectory counts, lengths, and oracles are in App. A. The three inference modes place different demands on data quality: BC sampling cannot exceed the demonstrating policy and video–inverse composition needs near-expert coverage, while CEM planning needs only goal-spanning data. The near-expert corpora meet all three on Two-Room, Push-T and Cube; Reacher's released corpus is random instead (App. A), and it behaves as this predicts, planning still reaching 80.9 SR while the two imitation paths fall to or below the random-policy floor (§3.3).

Models are trained for 10 epochs per env. Each (recipe, env) cell aggregates 50 episodes $\times$ 5 eval seeds $\times$ 3 train seeds = 750 episodes; we report mean and population std across the three train-seed means, matching Maes et al. (2026)'s 3-train-seed convention with 5$\times$ more eval episodes per train seed.

3.2 Goal-conditioned planning on the LeWM suite

We compare against the three end-to-end pixel-based world-model baselines reported on this suite by Maes et al. (2026): PLDM (Sobal et al., 2025), DINO-WM (Zhou et al., 2024) (no-proprio variant:

Table 1: **LeWM-suite results.** Per-environment success rate; mean ± population std across 3 train seeds. *Bold = best within our reproduction pipeline* (rows below the midrule); the top three rows are numbers reported by Maes et al. (2026), run under their pipeline, which trains every model for 10 epochs and evaluates all methods under a fixed 50-step episode budget, stricter than the original DINO-WM evaluation (DINO-WM reports 90.0 on Push-T under its own protocol). We retrain both LeWM and PLDM end-to-end in our pipeline, so the bottom block shares data, training scaffold and eval budget across all three rows. Both reproductions trail their published Reacher numbers by almost the same margin (-21.7 and -22.9 SR) while landing within 5.6 SR elsewhere, locating that gap outside either method (see below). PLDM's loss recipe (one of its two published ones) is chosen per environment on a single train seed (App. B).

method	Push-T	Two-Room	Cube	Reacher
PLDM (Sobal et al., 2025)	78.0 ± 5.0	97.0 ± 1.2	65.0 ± 2.8	78.0 ± 5.4
DINO-WM (Zhou et al., 2024)	74.0 ± 4.5	100.0 ± 0.0	86.0 ± 4.7	79.0 ± 5.1
LeWM (Maes et al., 2026) (paper)	96.0 ± 4.0	87.0 ± 2.5	74.0 ± 3.0	86.0 ± 5.0
PLDM (reproduction)	72.4 ± 1.8	96.9 ± 1.0	64.5 ± 1.9	55.1 ± 1.5
LeWM (reproduction)	$\mathbf{92.3} \pm 2.0$	89.9 ± 1.4	76.7 ± 1.3	64.3 ± 1.7
Qantara (ours)	90.1 ± 1.1	$\mathbf{100.0} \pm 0.0$	$\mathbf{93.7} \pm 0.7$	$\mathbf{80.9} \pm 1.8$

pixel-only inputs, matching the proprio-free setting of the other baselines), and LeWM (Maes et al., 2026) itself. Our training and evaluation pipeline extends LeWM's code, so we also retrain both LeWM and PLDM end-to-end in this pipeline. DINO-WM numbers are taken from Maes et al. (2026).

Per-environment, Qantara ties the Two-Room ceiling with DINO-WM and exceeds the DINO-WM row on Push-T (90.1 vs 74.0 re-evaluated) and Reacher (80.9 vs 79.0), trailing LeWM-paper by 5.9 and 5.1 on those two. An independent evaluation of LeWM's released checkpoint under its default planner and the published 50-episode size scores 89.3 ± 6.4 on Push-T and 68.0 ± 9.2 on Reacher (Nguyen et al., 2026), close to our LeWM reproduction (92.3 and 64.3). Retraining PLDM here shows the same pattern: within 5.6 SR of its published Push-T, Two-Room and Cube numbers but -22.9 on Reacher.

Two measurements place the Reacher gap in the training budget: a random policy scores 11.2 under our protocol against the published 10.9 (Maes et al., 2026), and at 20 epochs the LeWM reproduction rises from 64.3 to 80.0 ± 0.9 while Qantara stays flat (79.7 ± 1.8); PLDM recovers to 78.4 ± 3.3 (App. J). The Reacher column is therefore a matched-10-epoch statement, and the $+16.6$ Reacher margin below does not persist at the longer budget.

Head-to-head against the LeWM reproduction at this budget, Qantara wins on three of four envs (Two-Room $+10.1$, Cube $+17.0$, Reacher $+16.6$, the last of which the 20-epoch control above erases) and trails by 2.2 SR on Push-T. The Cube margin survives two controls trained in the same pipeline over three train seeds (App. B): DINO-WM retrained here reaches 80.0 ± 1.5 ($+13.7$ for Qantara), and a LeWM predictor grown to 20.75M parameters, against Qantara's 21.55M, reaches 74.7 ± 1.6, below the stock 76.7. On the harder two-cube variant (Cube-Double, App. A) the planners fall to 49.6 ± 1.0 for Qantara against 36.9 ± 0.7 for LeWM and 36.3 ± 0.5 for DINO-WM, all three trained in our pipeline.

3.3 The same checkpoint serves three inference paradigms

Tab. 2 reports all three paths on Push-T, Two-Room and Cube, queried from the same Qantara checkpoints as Tab. 1 at the inference defaults of §2.2 (CEM $K = 4$; BC and video–inverse $K = 2$) without per-paradigm retraining.

The goal-blind dispatches succeed when the demonstration distribution supplies the missing goal information and the env's success tolerance absorbs the imitation drift. In every env the eval goal is a later state of the dataset trajectory an episode starts from (App. A). On Push-T that trajectory heads for the task goal fixed across the corpus, which a goal-blind sampler internalises in its weights rather than reading it off pixels (masking the drawn marker at evaluation changes BC and video–inverse by at most 2 SR, within seed noise). BC and video–inverse reach 82.1 and

Table 2: **One Qantara checkpoint, three inference paradigms.** Mean SR $\pm$ std across 3 train seeds. CEM is goal-conditioned (it plans against the encoded goal z_g); BC and video–inverse are goal-blind imitation paths that receive no goal input and run at 17–27 ms per env-step versus CEM's 0.4–1.2 s. Reacher omitted: the goal-blind dispatches drop to near-noise there (6.1 / 4.8 SR for BC / video–inverse, see the text); its CEM result is in Tab. 1. The GCBC column (goal-conditioned behaviour cloning, Ghosh et al., 2021) is the suite authors' own number, run on DINOv2 patch embeddings rather than retrained in our pipeline, so it is a cross-pipeline reference and not a matched comparison.

	Qantara (single checkpoint)			GCBC
environment	CEM ($K=4$)	BC ($K=2$, goal-blind)	video–inverse ($K=2$, goal-blind)	
Push-T	90.1 ± 1.1	82.1 ± 1.0	83.2 ± 0.6	75
Two-Room	100.0 ± 0.0	69.1 ± 0.4	69.3 ± 0.8	100
Cube	93.7 ± 0.7	70.8 ± 0.3	72.8 ± 0.6	84

83.2 SR, above the 75 the suite authors report for goal-aware GCBC (see the caption), and match the 1-mode `policy` specialist trained in the same pipeline (81.3, Tab. 3); CEM uses z_g to recover the full 90.1 SR. On Two-Room and Cube the per-episode goal is randomised and never rendered, so BC and video–inverse reach it by reproducing the expert's continuation. On Reacher the per-episode target qpos requires precise joint matching that the goal-blind dispatches cannot reliably deliver, so BC and video–inverse drop to near-noise (6.1 / 4.8 SR; omitted from Tab. 2). The goal-blind paths are therefore the fast dispatch wherever the task goal is fixed across demonstrations (Push-T) or the success tolerance absorbs imitation drift (Two-Room, Cube); on precision targets such as Reacher, planning is the reliable dispatch. On Cube-Double the ordering flips: the goal-blind paths reach 54.3 ± 1.4 and 54.4 ± 2.3, above the planner's 49.6, so the best dispatch depends on the task.

At the inference defaults BC and video–inverse run $\sim$15–65$\times$ faster than CEM (App. I); decoded rollouts under all three dispatches stay coherent from the same checkpoint (App. H).

A joint video–action model under the same protocol. We train UVA (Li et al., 2025) on our Push-T corpus at a matched optimiser-step budget (248,268–248,336 steps against the released recipe's 247,050) and at its published inference cadence, then query its action head through the same evaluation code, episode budget and five evaluation seeds as every row above. Over three training seeds it reaches 14.4 ± 1.8 SR, against 82.1 for our goal-blind BC path and 1.6 ± 2.0 for a random policy. The UVA checkpoint fits its training data: run open-loop on frames from its own training corpus, it imitates the expert's next action chunk to 0.0156 mean absolute error in a two-unit-wide action space (18–21$\times$ better than reversed- and shuffled-frame controls), so the gap arises in closed-loop control. On one UVA training seed, replanning at every step instead of at its published cadence recovers only $+5.6$ SR (paired 95% CI $[+1.1, +10.1]$), about 8% of the gap.

3.4 Which design choices matter

Edge-aligned sampling beats uniform coverage. Uniform sampling on $[0, 1]^2$, the independent per-modality noise levels UWM trains with (Zhu et al., 2025), places measure zero on the four edges that inference queries (§2.2), and the cost falls hardest on the action-sensitive cells. Replacing the five edge-aligned modes (§2.3) with five copies of a uniform sampler at matched compute, with the bridge and all else fixed, regresses Push-T CEM by **36.8 SR** (90.1 $\rightarrow$ 53.3), Push-T video–inverse by 33.5 SR (83.2 $\rightarrow$ 49.7), and Reacher CEM by 23.4 SR (80.9 $\rightarrow$ 57.5); Cube CEM regresses modestly by 6.8 SR (93.7 $\rightarrow$ 86.9) and Push-T BC stays within seed noise (82.1 $\rightarrow$ 81.6) (Table 3, second row).

Mode-set ablation: trunk co-regularisation across modes. At a fixed batch of examples and optimiser-step count, we ablate the 5-mode design (§2.3) and evaluate all three inference paths per cell (Tab. 3; 11-recipe sweep). Per-mode coverage predicts that dropping a mode breaks only the inference path querying its corner; three findings on Push-T overturn this.

Of the four compute-matched drop-one rows, only `drop-video` preserves all three Push-T paths, staying within 2.8 SR of the 5-mode reference: the $(0, 0)$ corner where video–inverse starts denoising is still trained as the lower endpoint of the `joint` diagonal and of the `policy` edge, so the third path

Table 3: **Mode-set + sampler ablation at the default recipe**, multi-seed (3 train $\times$ 5 eval $\times$ 50 episodes). Inference defaults from §2.2: CEM $K=4$, BC and video–inverse $K=2$. †: train-seed collapse (pop-std $>$ 15 SR). Two-Room omitted (CEM reaches 100 SR at the reference, Tab. 2; no headroom for ablation). BC and video–inverse omit Cube and Reacher, where both goal-blind paths are insensitive to the mode set (Cube, 66–73 SR in every evaluated row but 1-mode {`forward`}) or collapse to near-noise (Reacher). The 1- and 2-mode rows double as the specialist checkpoints, each trained on that locus alone at the same recipe and step count. A row with M modes trains on an effective batch of $M \cdot B$, so rows with equal M are compute-matched.

	CEM			BC	video–inverse
modes (sampler)	Push-T	Cube	Reacher	Push-T	Push-T
5-mode (reference)	**90.1** $\pm$ 1.1	93.7 $\pm$ 0.7	**80.9** $\pm$ 1.8	82.1 $\pm$ 1.0	**83.2** $\pm$ 0.6
5-mode uniform	53.3 $\pm$ 9.6	86.9 $\pm$ 1.0	57.5 $\pm$ 6.9	81.6 $\pm$ 1.0	49.7 $\pm$ 1.3
4-mode drop-{`policy`}	27.2 $\pm$ 32.5†	89.9 $\pm$ 4.5	82.3 $\pm$ 4.0	20.4 $\pm$ 21.3†	21.7 $\pm$ 22.9†
4-mode drop-{`inverse`}	69.1 $\pm$ 23.7†	89.9 $\pm$ 0.9	78.9 $\pm$ 2.2	65.5 $\pm$ 25.6†	35.3 $\pm$ 15.7†
4-mode drop-{`video`}	87.3 $\pm$ 1.7	93.7 $\pm$ 0.2	80.1 $\pm$ 1.7	81.6 $\pm$ 1.2	81.6 $\pm$ 0.3
4-mode drop-{`joint`}	61.7 $\pm$ 41.1†	91.2 $\pm$ 2.3	76.7 $\pm$ 2.2	56.7 $\pm$ 38.1†	56.8 $\pm$ 37.3†
3-mode {`forward, policy, inverse`}	76.0 $\pm$ 21.8†	**93.9** $\pm$ 1.0	78.5 $\pm$ 0.2	68.4 $\pm$ 19.5†	65.5 $\pm$ 20.3†
2-mode {`forward, policy`}	88.5 $\pm$ 1.4	91.9 $\pm$ 0.2	76.4 $\pm$ 0.6	**82.3** $\pm$ 0.8	7.5 $\pm$ 1.5
2-mode {`video, inverse`}	10.4 $\pm$ 7.7	67.9 $\pm$ 4.8	26.3 $\pm$ 2.2	20.5 $\pm$ 9.2	37.6 $\pm$ 22.4†
1-mode {`forward`}	62.0 $\pm$ 33.9†	76.3 $\pm$ 7.6	56.0 $\pm$ 23.4†	2.4 $\pm$ 0.0	2.0 $\pm$ 0.0
1-mode {`policy`}	2.1 $\pm$ 0.7	50.3 $\pm$ 1.5	10.4 $\pm$ 1.0	81.3 $\pm$ 1.3	4.4 $\pm$ 1.0
1-mode uniform	40.5 $\pm$ 23.3†	84.9 $\pm$ 1.5	51.2 $\pm$ 7.4	48.5 $\pm$ 27.5†	27.9 $\pm$ 15.8†

survives without a dedicated mode. Dropping `policy`, `inverse`, or `joint` instead destabilises training across all three paths, CEM pop-std exceeding 15 SR with 1–2 of three seeds collapsing below 40 SR (sharpest: `drop-policy` 27.2$\pm$32.5 vs `drop-video` 87.3$\pm$1.7); the $(1, 0)$ corner is the only point where `policy`'s $\tau^z = 0$ edge and CEM's $\tau^a = 1$ query meet, yet removing `policy` still destabilises CEM, while Cube and Reacher hold within ~ 5 SR of the reference across the four drops. Neither tested 2-mode subset is self-sufficient: {`forward`, `policy`} collapses on video–inverse (7.5 SR) and {`video`, `inverse`} underperforms that same target (37.6 vs 83.2).

All five modes are therefore kept for *Push-T training stability*; `drop-video` already preserves the third path. On the collapsed seeds, final $\mathcal{L}_{\mathrm{B}}^{z}$ ends 2–40$\times$ above the reference and separates collapsed from healthy drop-one cells without overlap, so the instability is a training-time failure, not an eval-only artefact.

Output residual + zero-init is load-bearing. Ablating the state-head residual at fixed bridge interpolant, so the predictor is no longer the identity at initialisation (Eq. 6), regresses suite average by 5.6 SR and Push-T by 15.2 SR, with seed-level training instability (App. E).

Defaults robust to K; off-canonical γ collapses. Sweeping the predictor recursion step count $K \in \{1, 2, 4, 8\}$ leaves the suite average within seed noise (89.8–91.2 SR), and the $\gamma = \sqrt{2}$ Schrödinger-bridge canonical (Liu et al., 2023) ties our $\gamma = 1$ stochastic-interpolant canonical (Albergo et al., 2023) at 90.5 vs. 90.8 SR; off-canonical $\gamma \in \{0, 0.5, 2\}$ collapse 1–2 of 3 train seeds. We retain $\gamma = 1$, and drop the upstream LeWM predictor dropout ($p = 0.1 \to 0$; $+3.4$ SR suite average at an earlier recipe). Full sweeps: App. K.

4 Related work

World models for control. JEPA-style world models predict environment dynamics in a learned embedding space: PLDM (Sobal et al., 2025) trains end-to-end with VICReg, DINO-WM (Zhou et al., 2024) freezes DINOv2, LeWM (Maes et al., 2026; Balestriero & LeCun, 2025) pairs next-embedding prediction with an isotropic-Gaussian regulariser, and V-JEPA-2-AC (Assran et al., 2025) post-trains a 300M action-conditional predictor on a frozen 1B video JEPA for model-predictive control. Recent extensions push the recipe to dexterous manipulation (Goswami et al., 2025a), one-shot imitation (Goswami et al., 2025b), masked latent interventions (Nam et al., 2026), value-aligned latent shaping (Destrade et al., 2026), and JEPA world model design-choice studies (Terver et al., 2025). On this

suite Gao & Xu (2026) cut CEM solve time with action-prefix rollout and Nguyen et al. (2026) amortise the search into a head fitted on frozen LeWM latents, reporting higher search-free success than ours on Cube and Reacher. Adjacent world models include TD-MPC2 (Hansen et al., 2023; 2026), DreamerV4 (Hafner et al., 2025), and NWM (Bar et al., 2025); TD-MPC2 pairs a planner with a policy prior, the rest serve one paradigm. Qantara serves planning, behaviour cloning, and inverse dynamics from one checkpoint under a single objective.

Joint state-action denoising at billion scale. A second cluster trains one transformer to denoise observations and actions jointly and serves forward dynamics, inverse dynamics, and behaviour policy from one checkpoint, predicting the future observation rather than latent dynamics of their own: PAD (Guo et al., 2024) (joint-denoising DiT), UVA (Li et al., 2025) (decoupled video and action heads), UWM (Zhu et al., 2025) (per-modality timesteps), DUST (Won et al., 2025) (dual-stream MMDiT), WorldVLA / RynnVLA-002 (Cen et al., 2025b;a) (autoregressive token transformer), Cosmos Policy (Kim et al., 2026) (post-trained 2B video model), and DreamZero (Ye et al., 2026), a 14B autoregressive video-diffusion backbone whose flow-matching objective shares one denoising timestep across the video and action modalities. GR-1 (Wu et al., 2024) is the GPT-style ancestor. Parameter counts span 150M to 14B; all reconstruct the observation except DUST, which generates frozen-VLM embeddings of it, and none learns latent dynamics of its own. Trajectory-diffusion and per-token-noise antecedents (Janner et al., 2022; Ajay et al., 2022; Chen et al., 2024) are reviewed in App. M. Table 8 makes the capability structure of this cluster explicit: goal-conditioned test-time planning is the row no member of it exposes (UVA only re-ranks another policy's proposals), because trajectory optimisation queries the predictor hundreds of times per control step, which a compact latent space makes cheap. We also train the cluster's UVA on our Push-T corpus and evaluate it under our own protocol (§3.3).

Position. The *(JEPA-latent, multi-paradigm)* cell of the (representation $\times$ paradigm-count) plane the two clusters tile is sparse (Nguyen et al., 2026). Qantara reaches it from one joint objective and a single $\sim$21M-parameter checkpoint, up to three orders of magnitude smaller than the second cluster, trading raw capacity for low-latency deployment (17–27 ms per goal-blind step, §3.3).

5 Conclusion

Qantara trains a single $\sim$21M-parameter JEPA world model and serves three inference paradigms from one set of weights. On the LeWM-suite the planning path averages 91.2 SR and lifts OGBench-Cube to 93.7 SR, $+7.7$ over the suite's DINO-WM row (Tab. 1); from the same checkpoint the behaviour-cloning and video–inverse paths reach 82.1 and 83.2 SR on Push-T at ~ 15–$65\times$ lower inference cost than the planner (§3.3, Tab. 2).

The design follows a single rule: training should match how the model is queried at inference. The three dispatches read the (τ^a, τ^z) square only on its edges and corners, so training samples there; the planner exposes the previous clean latent at $\tau^z = 0$, so the state axis is a Brownian bridge to that anchor. On the most action-sensitive environment, uniform $[0, 1]^2$ sampling at matched compute regresses Push-T CEM by 36.8 SR (§3.4), and ablating the state-head residual + zero-init pair, a separate choice, regresses it by 15.2 SR (§3.4); replacing the bridge interpolant with a Gaussian source costs 12.7 SR on Reacher with the residual off (App. E). The mode-set ablation then produced a result we did not expect: dropping any single `policy`, `inverse`, or `joint` mode destabilises Push-T training across *all three* inference paths, including the planner, whose edge meets the dropped mode's at a single corner at most (§3.4, Tab. 3), so the five modes co-regularise the shared trunk beyond their per-corner coverage role.

Limitations. The capability claim is bounded on three axes: single-block denoising per control step, with chunked multi-block inference (Chen et al., 2024; Po et al., 2025) the direct extension; simulation-only evaluation on near-expert offline data, leaving real-robot transfer and natural-video pre-training (Assran et al., 2025) open; and a single $\sim$21M-parameter scale.

REPRODUCIBILITY STATEMENT

Unless labelled otherwise, every success rate is a mean over 50 episodes $\times$ 5 evaluation seeds $\times$ 3 training seeds, reported with the population standard deviation across the three training-seed means; the protocol, including the fixed 50-step episode budget, is stated in §3.1 and Tab. 1, and every cell that rests on a single training seed is labelled as such where it appears. Environments, datasets and their collection oracles are described in App. A; optimiser and training-time hyperparameters in App. B; the planner's cross-entropy-method and receding-horizon settings, which govern every goal-conditioned number, in App. F. Every baseline we compare against under a matched protocol (LeWM and PLDM on all four environments, DINO-WM on Cube and Cube-Double) was retrained in our own pipeline rather than transcribed, so the data, training scaffold and evaluation budget are shared across those rows (Tab. 1, §3.2). The training and evaluation code, every configuration file behind the reported runs, and the pinned environment lockfile are included as anonymised supplementary material.

AI USE STATEMENT

We used AI assistants in two ways. For research execution, AI coding assistants helped write training and evaluation code and make figures; the authors set the research direction, reviewed and tested all code, and recomputed every reported number from the stored run outputs. For writing, AI assistants helped edit and polish author-written text. AI tools were not used to generate data, and the method and its derivations are the authors' own. The authors take full responsibility for the content of this paper.

REFERENCES

A. Ajay, Yilun Du, Abhi Gupta, J. Tenenbaum, T. Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *International Conference on Learning Representations*, 2022. doi: 10.48550/arxiv.2211.15657. URL https://doi.org/10.48550/arxiv.2211.15657. Decision Diffuser; arXiv:2211.15657.

M. Albergo, N. Boffi, and E. Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of machine learning research*, 2023. doi: 10.48550/arxiv.2303.08797. URL https://doi.org/10.48550/arxiv.2303.08797.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Randall Balestriero and Yann LeCun. LeJEPA: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025.

Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15791–15801, 2025.

Jun Cen, Siteng Huang, Yuqian Yuan, Kehan Li, Hangjie Yuan, Chaohui Yu, Bohan Hou, Yuming Jiang, Jiayan Guo, Xin Li, Hao Luo, Fan Wang, Deli Zhao, and Hao Chen. RynnVLA-002: A unified vision-language-action and world model. *arXiv preprint arXiv:2511.17502*, 2025a.

Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, Deli Zhao, and Hao Chen. WorldVLA: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025b.

Boyuan Chen, Diego Marti Monso, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *NeurIPS*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/2aee1c4159e48407d68fe16ae8e6e49e-Abstract-Conference.html.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

Matthieu Destrade, Oumayma Bounou, Quentin Le Lidec, Jean Ponce, and Yann LeCun. Value-guided action planning with JEPA world models. *arXiv preprint arXiv:2601.00844*, 2026.

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.

Dechen Gao, Boqi Zhao, Andrew Lee, Ian Chuang, Hanchu Zhou, Hang Wang, Zhe Zhao, Junshan Zhang, and Iman Soltani. VITA: Vision-to-action flow matching policy. In *ICLR*, 2026. URL https://openreview.net/forum?id=BTe5VLBjPg.

Yuntian Gao and Xiangyu Xu. Fast LeWorldModel. *arXiv preprint arXiv:2606.26217*, 2026.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Manon Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. In *ICLR*, 2021. URL https://openreview.net/forum?id=rALA0Xo6yNJ.

Raktim Gautam Goswami, Amir Bar, David Fan, Tsung-Yen Yang, Gaoyue Zhou, Prashanth Krishnamurthy, Michael Rabbat, Farshad Khorrami, and Yann LeCun. World models for learning dexterous hand-object interactions from human videos. *arXiv preprint arXiv:2512.13644*, 2025a.

Raktim Gautam Goswami, Prashanth Krishnamurthy, Yann LeCun, and Farshad Khorrami. OSVI-WM: One-shot visual imitation for unseen tasks using world-model-guided trajectory generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b.

Yanjiang Guo, Yucheng Hu, Jianke Zhang, Yen-Jen Wang, Xiaoyu Chen, Chaochao Lu, and Jianyu Chen. Prediction with action: Visual policy learning via joint denoising process. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. *International Conference on Learning Representations*, 2023. doi: 10.48550/arxiv.2310.16828. URL https://doi.org/10.48550/arxiv.2310.16828.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Learning massively multitask world models for continuous control. In *ICLR*, 2026. URL https://openreview.net/forum?id=MPabX9LEds.

Michael Janner, Yilun Du, J. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. *International Conference on Machine Learning*, 2022. doi: 10.48550/arxiv.2205.09991. URL https://doi.org/10.48550/arxiv.2205.09991. arXiv:2205.09991.

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. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.

Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Guan-Horng Liu, Arash Vahdat, De-An Huang, Evangelos A. Theodorou, Weili Nie, and Anima Anandkumar. I^2SB: Image-to-image Schrödinger bridge. *International Conference on Machine Learning*, 2023. doi: 10.48550/arxiv.2302.05872. URL https://doi.org/10.48550/arxiv.2302.05872.

Kunhao Liu, Wenbo Hu, Jiale Xu, Ying Shan, and Shijian Lu. Rolling forcing: Autoregressive long video diffusion in real time. In *International Conference on Learning Representations (ICLR)*, 2026.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *International Conference on Learning Representations*, 2022. doi: 10.48550/arxiv.2209.03003. URL https://doi.org/10.48550/arxiv.2209.03003.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Zonglin Lyu, Ming Li, Jianbo Jiao, and Chen Chen. Frame interpolation with consecutive Brownian bridge diffusion. In *Proceedings of the 32nd ACM International Conference on Multimedia (MM '24)*, 2024. doi: 10.1145/3664647.3680961.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorld-Model: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

Heejeong Nam, Quentin Le Lidec, Lucas Maes, Yann LeCun, and Randall Balestriero. Causal-JEPA: Learning world models through object-level latent masking. *International Conference on Machine Learning*, 2026.

Hoang Nguyen, Xiaohao Xu, and Xiaonan Huang. Latent geometry beyond search: Amortizing planning in world models. *arXiv preprint arXiv:2605.08732*, 2026.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. *International Conference on Learning Representations*, 2024. doi: 10.48550/arxiv.2410.20092. URL https://doi.org/10.48550/arxiv.2410.20092.

William S. Peebles and Saining Xie. Scalable diffusion models with transformers. *IEEE International Conference on Computer Vision*, 2022. doi: 10.1109/iccv51070.2023.00387. URL https://doi.org/10.1109/iccv51070.2023.00387.

Ethan Perez, Florian Strub, H. D. Vries, Vincent Dumoulin, and Aaron C. Courville. FiLM: Visual reasoning with a general conditioning layer. *AAAI Conference on Artificial Intelligence*, 2017. doi: 10.1609/aaai.v32i1.11671. URL https://doi.org/10.1609/aaai.v32i1.11671.

Ryan Po, Yotam Nitzan, Richard Zhang, Berlin Chen, Tri Dao, Eli Shechtman, Gordon Wetzstein, and Xun Huang. Long-context state-space video world models. In *ICCV*, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Po_Long-Context_State-Space_Video_World_Models_ICCV_2025_paper.html.

Reuven Y. Rubinstein and Dirk P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*. Information Science and Statistics. Springer, 2004.

Vlad Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Stress-testing offline reward-free reinforcement learning: A case for planning with latent dynamics models. In *7th Robot Learning Workshop: Towards Robots with Human-Level Abilities*, 2025. URL https://openreview.net/forum?id=jON7H6A9UU.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, and Martin A. Riedmiller. DeepMind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Basile Terver, Tsung-Yen Yang, Jean Ponce, Adrien Bardes, and Yann LeCun. What drives success in physical planning with joint-embedding predictive world models? *Trans. Mach. Learn. Res.*, 2025. doi: 10.48550/arxiv.2512.24497. URL https://doi.org/10.48550/arxiv.2512.24497.

Christopher Wewer, Bart Pogodzinski, Bernt Schiele, and Jan Eric Lenssen. Spatial reasoning with denoising models. In *ICML*, 2025.

John Won, Kyungmin Lee, Huiwon Jang, Dongyoung Kim, and Jinwoo Shin. DUST: 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 *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

Amber Xie, Oleh Rybkin, Dorsa Sadigh, and Chelsea Finn. Latent diffusion planning for imitation learning. In *ICML*, 2025.

Desai Xie, Zhan Xu, Yicong Hong, Hao Tan, Difan Liu, Feng Liu, Arie E. Kaufman, and Yang Zhou. Progressive autoregressive video diffusion models. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2024. doi: 10.1109/cvprw67362.2025.00628. URL https://doi.org/10.1109/cvprw67362.2025.00628.

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*, 2026.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. *International Conference on Machine Learning*, 2024. doi: 10.48550/arxiv.2411.04983. URL https://doi.org/10.48550/arxiv.2411.04983.

Chuning Zhu, Raymond Yu, Siyuan Feng, B. Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. *Robotics*, 2025. doi: 10.48550/arxiv.2504.02792. URL https://doi.org/10.48550/arxiv.2504.02792.

Hongzhou Zhu, Min Zhao, Guande He, Hang Su, Chongxuan Li, and Jun Zhu. Causal forcing: Autoregressive diffusion distillation done right for high-quality real-time interactive video generation. *International Conference on Machine Learning*, 2026.

APPENDIX

A ENVIRONMENTS AND DATASETS

All four envs use continuous action spaces and are goal-conditioned on a future state sampled from the same trajectory. **Two-Room** (Sobal et al., 2025) is 2D continuous navigation through a wall-with-door (10k trajectories, average 92 steps, generated by a noisy door-then-target heuristic). **Push-T** (Chi et al., 2023) is contact-rich 2D manipulation reorienting a T-block; we use the DINO-WM-released dataset (Zhou et al., 2024) of 20k human-designed expert demonstrations averaging 196 steps. **OGBench-Cube** (Park et al., 2024) refers to OGBench's cube-single task (one cube, 10k expert trajectories of 200 steps); Cube in this paper always denotes this single-cube variant. Trajectories are generated by the OGBench scripted data-collection policy. **Cube-Double** is the two-cube variant of the same environment; following the recipe behind the single-cube corpus, we generate 10k trajectories of 200 steps at 224 px with the same scripted oracle and collection mode, and plan and evaluate with the Cube settings unchanged. **Reacher-Hard** from the DeepMind Control suite (Tassa et al., 2018) requires precise two-joint alignment, 10k trajectories of 200 steps. We use the Reacher corpus that the LeWM codebase and `stable-worldmodel` ship, which is collected by a uniform random policy rather than by the Soft Actor-Critic policy the LeWM appendix describes. Measured on the released file: the action marginal is uniform to within 0.1% over ten bins, mean finger-to-target distance is flat across the episode (0.203 at step 0, 0.204 at step 200), and only 3.8% of steps fall inside the success tolerance. Two observations indicate the published Reacher numbers rest on this same corpus rather than on a Soft Actor-Critic one: it is the only Reacher dataset either shipped configuration names, and the random-policy control agrees across the two pipelines (11.2 ± 4.8 here against a published 10.9 ± 5.3), which a goal distribution drawn from a converged reaching policy would not reproduce.

B Optimisation and training-time hyperparameters

We train with AdamW (Loshchilov & Hutter, 2019): weight decay 10^{-3}, gradient clip 1.0, batch size 128, bf16 mixed precision, peak learning rate 3×10^{-4} under a linear-warmup cosine-annealing schedule (1% warmup, cosine decay to 0 over the remaining steps). All five training modes (§2.3) are enabled by default, so the effective batch is $5\cdot B=640$.

One environment trains in roughly 2–3 hours on a single H100 for the $\sim$21M-parameter model (10 epochs, $\approx$50k optimiser steps at $B=128$). The LeWM baseline row in Tab. 1 was retrained in this same pipeline with the same data, $B=128$ source rows per step and optimiser-step count, as was the PLDM reproduction row (16 training runs: two loss recipes on train seed 11 to select one per environment, then the winner on seeds 22 and 33). The DINO-WM row and the top-block PLDM row are as published by Maes et al. (2026). The two Cube controls of §3.2 are trained in this pipeline for 10 epochs over three train seeds: DINO-WM with the recipe shipped in the suite's codebase (frozen DINOv2-S/14 encoder, depth-6 predictor, batch size 32, pixel-only inputs as in the suite's DINO-WM row), and LeWM with its predictor grown from depth 6 and MLP width 2048 to depth 7 and width 2400 (20.75M parameters excluding the decoder, against 18.04M for stock LeWM and 21.55M for Qantara). LeWM and DINO-WM are retrained the same way on Cube-Double. The *default recipe* fixes the $\gamma=1$ Brownian bridge on the state axis, loss balance $\lambda_z=3$, $\lambda_a=1$, and the inference defaults of §2.2 (CEM $K=4$, BC and video–inverse $K=2$); 10 epochs correspond to $\approx$50k optimiser steps at $B=128$.

C Monotone τ-chain on the noisy suffix

For each row, the per-block scalar τ on the noisy suffix is built as a cumulative product of i.i.d. uniforms,

$$\tau[t] \;=\; \prod_{i\le t} u_i, \qquad u_i \sim \mathcal{U}[0,1], \tag{11}$$

monotone-decreasing in the block index. The first noisy block (immediately after the clean prefix) carries an unbiased $\tau\sim\mathcal{U}[0,1]$; deeper blocks are stochastically noisier as the product accumulates additional uniforms below 1. Together with the per-row clean-prefix length, the schedule places training mass on every (τ^a,τ^z) pair an autoregressive rollout can reach. Every noisy block $s\ge t$ of Eq. 2 forms its bridge input (Eq. 4) from the clean dataset pair (z_s, z_{s+1}), and the state head's residual (Eq. 6) is taken on the same clean z_s; for $s>t$ the clean z_s enters only through the corrupted input and the residual and is never supplied as a context token.

Empirical role. Replacing the cumulative-product chain with i.i.d. per-block draws leaves every per-environment SR within ±2 SR of the cumprod chain at the 5-mode reference recipe (single-seed), but destabilises Push-T training under reduced mode sets: at the 3-mode {`forward`, `inverse`, `policy`} variant of the prior $\lambda_z=1$ reference recipe, the multi-seed Push-T mean drops from 84.4 ± 0.9 (cumprod) to 75.1 ± 16.6 (i.i.d.; one of three train seeds collapses Push-T to 53.2 SR), with the other three environments unchanged within seed noise. The cumprod chain matches the rolling-denoise structure an autoregressive rollout induces; removing the inductive bias is invisible at the over-parameterised 5-mode design but breaks Push-T under sparser training-mode coverage. We retain the chain as a safety net.

D Brownian γ sweep, full table

Table 4 gives the full Brownian-bridge γ sweep summarised in §3.4.

E Output residual parameterisation

The state-head residual form $\hat z_{t+1}=z_t+\mathrm{head}_z(h^z_{t+1})$ (Eq. 6) pairs with zero-initialised final-layer weights so the predictor is the identity at initialisation. We ablate the residual by having the head emit $\hat z_{t+1}$ directly with default Linear initialisation, holding the Brownian-bridge input interpolant

Table 4: **Brownian-bridge γ sweep at the default recipe** (multi-seed, $K=1$). $\gamma=1$ is the Albergo et al. stochastic-interpolant canonical (Albergo et al., 2023); $\gamma=\sqrt{2}$ is the Schrödinger-bridge canonical (Liu et al., 2023); the two tie on suite average. The off-canonical $\gamma\in\{0, 0.5, 2\}$ collapse 1–2 of 3 Push-T seeds and $\gamma=2$ additionally collapses one Two-Room seed. The $\gamma=1$ row is the training-time evaluation of the Table 1 checkpoints; their re-evaluation in Table 6 reads 89.1 on Push-T. ‡mean masks seed-level collapse.

γ	Push-T	Two-Room	Cube	Reacher
$0^{\ddagger}$	31.5 ± 35.5	99.6 ± 0.0	92.5 ± 0.7	78.3 ± 2.7
$0.5^{\ddagger}$	60.1 ± 39.7	99.9 ± 0.2	93.6 ± 0.6	78.8 ± 1.0
1 (Albergo SI canonical, default)	$\mathbf{89.3} \pm 1.4$	$\mathbf{100.0} \pm 0.0$	$\mathbf{93.3} \pm 0.7$	80.5 ± 3.0
$\sqrt{2}$ (Schrödinger-bridge canonical)	89.3 ± 0.2	$\mathbf{100.0} \pm 0.0$	91.9 ± 1.7	$\mathbf{80.8} \pm 2.6$
$2^{\ddagger}$	53.6 ± 36.6	71.5 ± 40.4	$\mathbf{93.6} \pm 0.3$	79.6 ± 3.1

$\tilde{z}_{t+1} = (1-\tau)\, z_t + \tau\, z_{t+1} + \gamma\sqrt{\tau(1-\tau)}\, \varepsilon_z$ and every other recipe knob fixed at the default recipe (Table 5, row 1 vs row 2).

To complete the factorial we also ablate the input-side bridge with the residual already removed: the source endpoint becomes Gaussian noise rather than z_t, so $\tilde{z}_{t+1} = \tau\, z_{t+1} + (1-\tau)\, \varepsilon_z$ (Table 5, row 3). Comparing rows 2 and 3 isolates the bridge contribution at residual = off; the fourth corner (bridge off, residual on) is structurally unreachable in our implementation since the residual head is gated on the bridge being active. Restoring the bridge interpolant on top of (residual off) yields a +12.7 SR lift on Reacher (precision motor control), a within-noise change on Cube, saturation at 100 SR on Two-Room, and a Push-T comparison confounded by the row-2 seed-level instability noted above. The z_t-anchored source endpoint is therefore most informative on Reacher, while the output-side residual stabilises Push-T and Cube training.

Table 5: **State-head and bridge structural ablations, multi-seed, all rows at $K=1$.** The output-side residual head (row 1 vs row 2) is load-bearing: removing it costs 5.6 SR on suite average and 15.2 SR on Push-T with one of three train seeds collapsing. The input-side bridge interpolant (row 2 vs row 3) is selectively load-bearing on Reacher: restoring it yields $+12.7$ SR there, within-noise elsewhere. The two priors are complementary, not redundant.

bridge	residual	Push-T	Two-Room	Cube	Reacher
on	on (zero-init; default)	$\mathbf{89.3} \pm 1.4$	100.0 ± 0.0	$\mathbf{93.3} \pm 0.7$	$\mathbf{80.5} \pm 3.0$
on	off (default Linear init)	74.1 ± 12.3	100.0 ± 0.0	86.3 ± 1.3	80.4 ± 2.4
off	off (default Linear init)	86.5 ± 0.5	100.0 ± 0.0	87.1 ± 0.8	67.7 ± 1.0

F CEM AND MPC CONFIGURATION

We inherit the CEM and receding-horizon MPC setup from LeWM (Maes et al., 2026), which in turn follows DINO-WM (Zhou et al., 2024), and plan every method trained in our pipeline with these same settings. CEM uses $N=300$ Gaussian candidates per iteration, 30 refit iterations on Push-T and 10 elsewhere, top-30 elites, and an initial sampling variance of 1. The planning horizon is $H=5$; under a frame-skip of 5 (action-block repetition), each planned action commands 5 environment steps, so one plan covers 25 environment steps. We use a receding horizon equal to the planning horizon: the entire elite-mean plan is executed before replanning. All other settings are identical across the four environments.

G CEM DENOISING-STEP SWEEP

We sweep the state-axis predictor recursion step count $K \in \{1, 2, 4, 8\}$ on the trained Qantara checkpoints (§3.2; Table 6). On the `forward` edge the recursion starts from the bridge source $\tilde{z}^{(0)}_{t+1} = z_t$ and makes K predictor calls at $\tau^z_k = k/K$, $k=0, \ldots, K-1$. Call k reads $\tilde{z}^{(k)}_{t+1}$ and returns

the endpoint estimate $\hat{z}^{(k)}_{t+1} = z_t + \mathrm{head}_z(h^z_{t+1})$ (Eq. 6), which is re-projected onto the bridge with fresh noise $\xi_k \sim \mathcal{N}(0, I)$,

$$\tilde{z}^{(k+1)}_{t+1} \;=\; (1 - \tau^z_{k+1})\, z_t \;+\; \tau^z_{k+1}\, \hat{z}^{(k)}_{t+1} \;+\; \gamma \sqrt{\tau^z_{k+1}(1 - \tau^z_{k+1})}\, \xi_k, \tag{12}$$

so each call's input lies on the training-time bridge marginal. The noise term vanishes at $\tau^z_K = 1$, giving $\hat{z}_{t+1} = \tilde{z}^{(K)}_{t+1} = \hat{z}^{(K-1)}_{t+1}$; at $K = 1$ the recursion is the deterministic single call. The `video` factor of the video–inverse composition is this single call with the action slot at Gaussian noise. The state head is trained by endpoint regression under the bridge corruption (Eq. 7), and the bridge enters inference through Eq. (12): each call returns an endpoint estimate, which the recursion maps back onto the bridge at the next τ^z, as endpoint-predicting diffusion samplers re-noise their clean estimate. We report the same sweep for the $\gamma = \sqrt{2}$ checkpoints of Table 4 for comparison.

Table 6: **CEM predictor recursion step count K at the trained Qantara checkpoints, multi-seed.** $\gamma = 1$, $K = 4$ is the default (Table 1). $K = 8$ regresses on Push-T at both γ values; $\gamma = \sqrt{2}$ ties $\gamma = 1$ on suite average and is dominated at $K = 4$.

γ	K	Push-T	Two-Room	Cube	Reacher
1	1	89.1 ± 1.3	100.0 ± 0.0	93.3 ± 0.7	80.5 ± 3.0
1	2	88.0 ± 0.9	100.0 ± 0.0	93.3 ± 2.1	80.1 ± 1.9
1	**4** (default)	**90.1** ± 1.1	100.0 ± 0.0	**93.7** ± 0.7	**80.9** ± 1.8
1	8	87.2 ± 1.4	100.0 ± 0.0	93.2 ± 0.6	78.7 ± 1.9
$\sqrt{2}$	1	89.3 ± 0.2	100.0 ± 0.0	91.9 ± 1.6	80.8 ± 2.6
$\sqrt{2}$	2	90.4 ± 2.0	100.0 ± 0.0	93.1 ± 2.2	79.3 ± 0.5
$\sqrt{2}$	4	89.7 ± 1.5	100.0 ± 0.0	92.3 ± 2.8	80.0 ± 2.3
$\sqrt{2}$	8	88.9 ± 1.3	100.0 ± 0.0	92.0 ± 2.7	77.3 ± 3.0

H MULTI-PARADIGM DECODED ROLLOUTS

Starting from one context frame at $t = 0$, we roll the predictor forward six steps under three action sources (Fig. 3). The *demo* row replays the demonstration actions recorded for the held-out trajectory and exercises the forward-dynamics primitive CEM scores inside its candidate-search loop; the *BC* row autoregressively chains BC action sampling with the same forward dynamics; the *video–inv* row chains video–inverse action extraction with forward dynamics. All three rows share the starting latent and the forward-dynamics call, differing only in how the action sequence is chosen; the $t = 0$ tile (left of the dashed separator) is the shared encoded context decoded once, and $t = 1 - 6$ are decoded predicted latents. Decoded trajectories remain coherent across the planning horizon; finer details (T-block angle on Push-T, end-effector orientation on Cube, agent position in Two-Room and Reacher) drift after a few steps, matching the characterisation of decoded JEPA latents in Maes et al. (2026, Fig. 7). The decoder behind these figures is a convolutional head trained alongside the model on stop-gradient predicted latents with a pixel MSE loss: no gradient flows from the decoder into the encoder or the predictor, every parameter count in the paper excludes the decoder, and no dispatch runs the decoder. The training times of App. B include the decoder, which the LeWM reproduction trains as well.

I PER-DISPATCH WALL-CLOCK

Table 7 is the measurement behind the ~ 15–$65\times$ figure quoted in §3.3. CEM latent planning runs $N_{\mathrm{samples}} \times H \times K$ predictor passes per refit iteration ($300 \times 5 \times 4 = 6000$ at the headline settings, with 30 iterations per plan on Push-T and 10 elsewhere), while the two goal-blind dispatches need $N_a = 2$ Euler steps each. The evaluation calls `get_action` once per environment step, and a dispatch computes only when its action buffer is empty: CEM once per plan, every 25 environment steps (App. F), and the goal-blind dispatches once per 5-step frame-skip block. The means in Table 7 therefore amortise each dispatch over its own replanning cadence, and one CEM planning call is the worst-case decision latency. The measured ratios span $14.7\times$ (Reacher video–inverse) to $65.4\times$ (Push-T behaviour cloning).

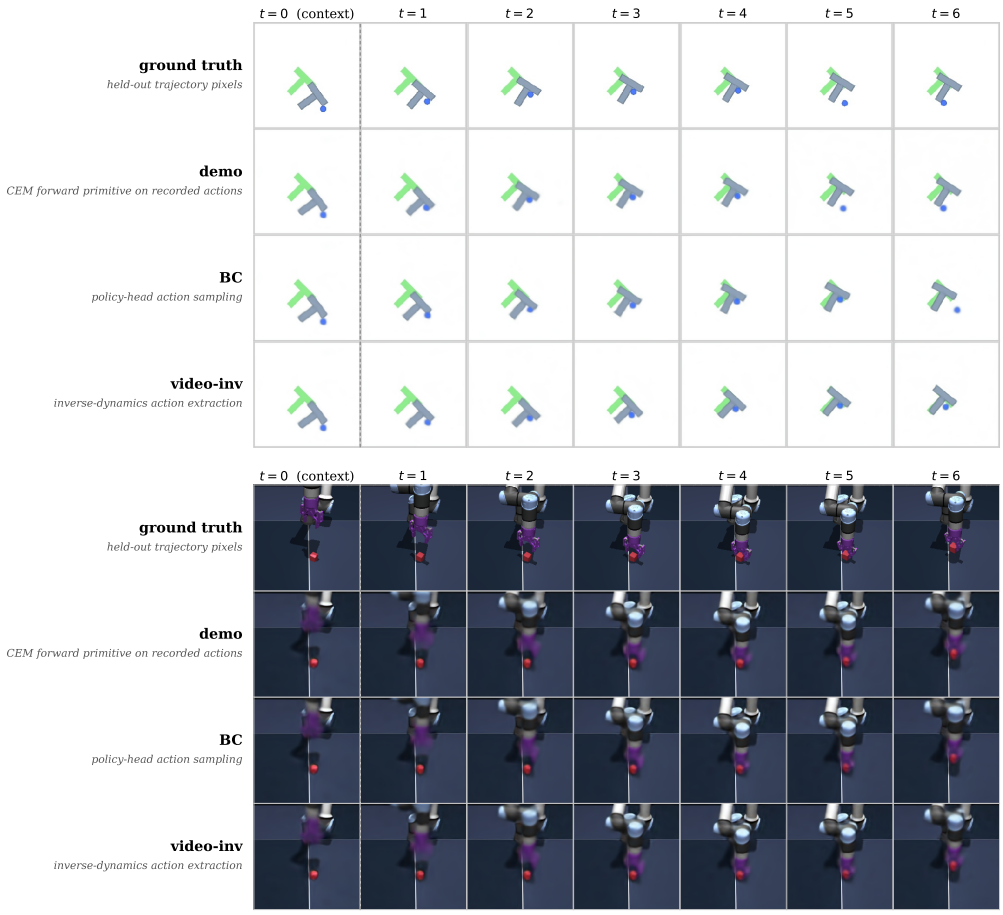

Figure 3: **Same checkpoint, three action sources, three coherent decoded rollouts** (Push-T top, Cube bottom; Two-Room and Reacher continued in Fig. 3). Columns are timesteps $t = 0 - 6$; the dashed separator marks the boundary between the encoded context ($t = 0$) and the open-loop predictor rollout ($t = 1 - 6$). Top row: ground-truth pixels (decoder not used). The next three rows decode the predictor's open-loop latent rollout under *demo* (CEM forward primitive on recorded actions), *BC* (policy-head action sampling), and *video–inv* (inverse-dynamics action extraction) action sources; all three rows share the starting latent and the forward-dynamics call.

Table 7: **Mean `get_action` wall-clock per env-step, seconds.** One training seed (s11), 5 episodes per environment, evaluation seed 42, `num_envs` = 5, on a single H100, at the headline solver settings (CEM $N_{\text{samples}} = 300$, $n_{\text{steps}} = 30$ on Push-T and 10 elsewhere, $K = 4$, horizon 5; BC and video–inverse $N_a = 2$). Times include CUDA synchronisation. Unlike the success rates in this paper, these are single-train-seed timings and are reported as an order-of-magnitude envelope rather than a three-seed mean.

env	CEM latent planning	BC $\tau^z = 0$ Euler	video–inverse	CEM/BC	CEM/vid–inv
Push-T	1.166 ± 0.003	0.018 ± 0.000	0.019 ± 0.001	$65.4\times$	$60.1\times$
Two-Room	0.398 ± 0.001	0.017 ± 0.000	0.019 ± 0.000	$22.9\times$	$20.9\times$
Cube	0.404 ± 0.001	0.022 ± 0.000	0.025 ± 0.001	$18.1\times$	$16.4\times$
Reacher	0.403 ± 0.003	0.024 ± 0.004	0.027 ± 0.003	$16.7\times$	$14.7\times$

J REACHER TRAINING-BUDGET CONTROLS

Two measurements place the Reacher gap in the training budget rather than in the benchmark or the methods. First, the evaluation instance is consistent with the published one: a random policy

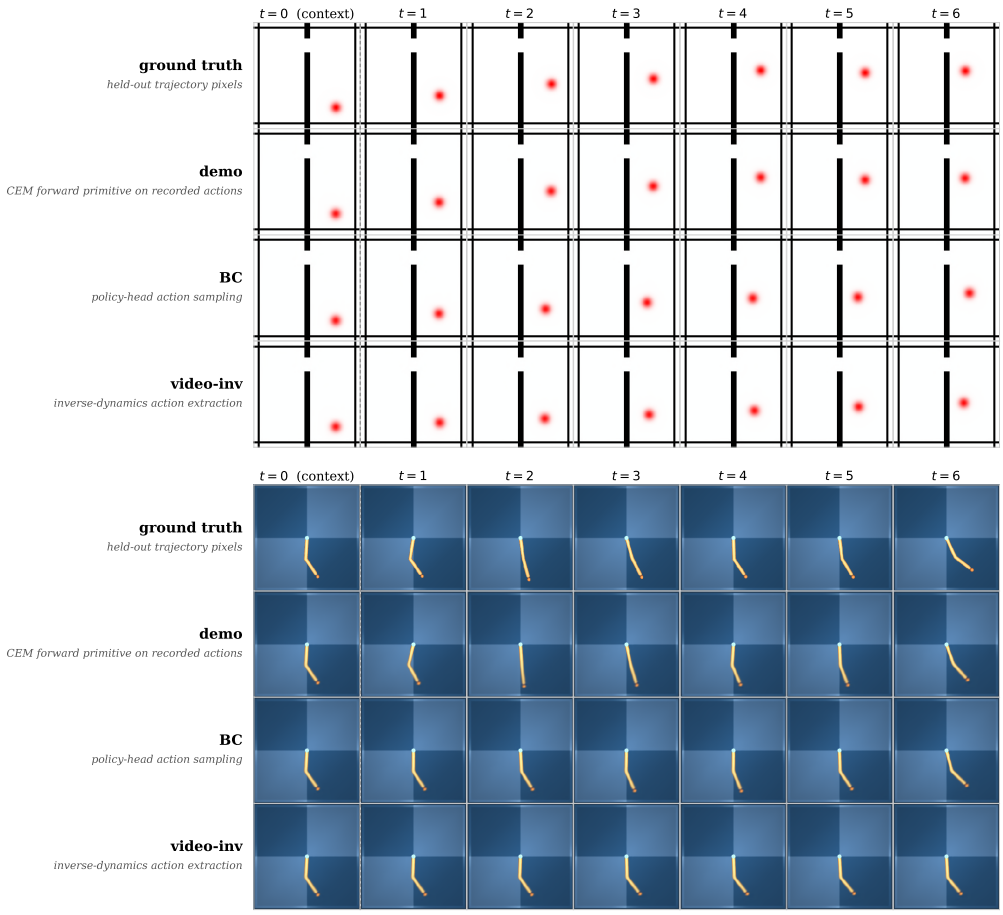

Figure 3: (Continued.) Two-Room (top) and Reacher (bottom) decoded rollouts under the same three action sources; layout and column semantics as in Fig. 3.

scores 11.2 (mean over 5 eval seeds) under our protocol against the 10.9 reported for the same control by Maes et al. (2026), as expected if both draw start–goal pairs from the same corpus, and unlike what pairs drawn from a converged reaching policy would give (App. A). Second, the deficit closes at a larger training budget: retraining the LeWM reproduction for 20 epochs over 5 train seeds lifts Reacher from 64.3 to 80.0 ± 0.9, which is 1.2 published standard deviations below the 86.0 row, while Qantara stays flat (79.7 ± 1.8) and the two models become indistinguishable there. On a single train seed the same doubling moves the other three environments by under 1 SR, so Reacher is where the 10-epoch budget of Table 1 binds. The Reacher column of Table 1 is therefore a matched-10-epoch statement for every method in the bottom block, and the $+16.6$ Reacher margin of §3.2 does not persist at the longer budget. PLDM recovers at three train seeds: Reacher $55.1 \rightarrow 78.4 \pm 3.3$ over seed-matched 10-epoch cells; at one seed, Push-T, Two-Room and Cube move -4.4, -0.4, $+2.0$ SR.

K Robustness to the remaining knobs

Sweeping the state-axis predictor recursion step count $K \in \{1, 2, 4, 8\}$ (§2.2) on the trained Qantara checkpoints leaves the suite average within seed noise (89.8–91.2 SR; App. G); the action-axis Euler step count moves Push-T BC SR within 2.9 SR across $K \in \{1, 2, 4, 8\}$, peaking at $K = 4$ with $+1.1$ SR over the $K = 2$ default we deploy for latency. The $\gamma = \sqrt{2}$ Schrödinger-bridge canonical (Liu et al., 2023) ties our $\gamma = 1$ stochastic-interpolant canonical (Albergo et al., 2023) on suite average (90.5 vs. 90.8 SR, within seed noise), while off-canonical $\gamma \in \{0, 0.5, 2\}$ are unstable: 1–2 of 3 train seeds collapse on Push-T, and $\gamma = 2$ additionally collapses Two-Room on one seed (App. D). We retain $\gamma = 1$. Finally, the upstream LeWM recipe ships predictor dropout $p = 0.1$; ablating to $p = 0$ at

an earlier reference recipe lifts the suite average from 84.9 to 88.3 SR (3 train seeds), and we keep $p=0$ as our default.

L CAPABILITY STRUCTURE OF THE JOINT VIDEO–ACTION CLUSTER

Table 8 lists the capabilities shown in each paper of the joint video–action cluster (§4).

Table 8: **Capability structure of the joint video–action cluster.** Parameter counts as stated in each paper; "n/s": not stated; "not shown": not demonstrated in the paper. †: ranks a best-of-N action set by a learned value function, optimising no test-time goal. §: re-ranks 100 action sequences proposed by a separate pretrained policy with a hand-coded block-to-target reward on predicted frames, optimising no action sequence against a goal observation. ‡: fully fine-tuned from the 2B Cosmos-Predict2 video model.

capability	UVA	PAD	UWM	DUST	Cosmos Policy	Qantara
BC action sampling	yes	yes	yes	yes	yes	yes
inverse dynamics	yes	no	yes	not shown	no	yes
video / next-obs prediction	yes	yes	yes	yes	yes	yes (latent)
goal-conditioned test-time planning	no[§]	no	no	no	no[†]	**yes**
prediction space	pixel	pixel	pixel	VLM embed.	pixel	JEPA latent
parameters	0.5B	0.66B	n/s (DiT-B)	n/s	2B[‡]	**0.021B**

M TRAJECTORY-DIFFUSION AND FORCING ANTECEDENTS

Earlier trajectory-diffusion baselines (Diffuser (Janner et al., 2022), Decision Diffuser (Ajay et al., 2022)) operate on raw state-action vectors; Decision Diffuser shares the state-diffusion-plus-inverse-dynamics factorisation with our video–inverse path, but uses classifier-free return guidance over multi-step horizons on D4RL/Kuka, whereas Qantara operates on pixel-encoded JEPA latents at single-step horizon. Latent Diffusion Planning (Xie et al., 2025) applies the same factorisation in a learned image latent space, with a diffusion planner over future latents and a separate inverse dynamics model on latent pairs; in Qantara both factors are edges of one checkpoint. Diffusion Forcing (Chen et al., 2024) introduces per-token noise levels and uses one sequence model over action, reward and observation tokens both as a policy and, with reward guidance over a longer lookahead, as a planner; Qantara instead denoises pixel-encoded JEPA latents, plans by CEM against an encoded goal observation, and trains on the edges and diagonal of the (τ^a, τ^z) square rather than on i.i.d. per-token noise levels. Follow-ups add clean-prefix conditioning (Zhu et al., 2026; Po et al., 2025). Matching the training distribution of noise levels to the one inference visits is also the principle behind the Uniform-$\bar{t}$ sampler of Spatial Reasoning Models (Wewer et al., 2025), which replaces Diffusion Forcing's i.i.d. per-token levels, whose mean concentrates around 0.5; Qantara applies the principle to the edges and corners of the (τ^a, τ^z) square that the three dispatches query.