Qantara: Bridge-Flow Training for Multi-Paradigm JEPA Control

Ruslan Rakhimov* **George Bredis** **Yuriy Maksyuta** **Daniil Gavrilov**
T-Tech

Abstract

Joint-Embedding Predictive Architectures (JEPAs) underpin a growing family of latent world models for control from raw pixels, but every existing JEPA world model commits at training time to a single inference paradigm: either trajectory optimisation in a learned dynamics model, or direct behaviour cloning. A single checkpoint that serves both would defer this choice to inference, when deployment constraints (rollout cost, observation accessibility) determine which path wins. We present **Qantara**, an end-to-end JEPA 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. The same checkpoint serves three inference paradigms without retraining: latent planning, behaviour-cloning action sampling, and inverse dynamics, which we query through a video–inverse composition that first predicts the next latent without action conditioning, then extracts the action. Training concentrates mass on the edges of the (action-time, state-time) noise square, where inference queries the predictor: replacing it with uniform interior sampling drops Push-T planning from 90.1 to 53.3 SR at matched compute. On the LeWM control suite, Qantara reaches a 91.2 SR three-train-seed average and sets new SOTA on OGBench-Cube ($+7.7$ SR over DINO-WM, $+19.7$ over LeWM). From the same weights, the behaviour-cloning and video–inverse paths reach 82–83 SR on Push-T and 71–73 SR on Cube. These results move JEPA world models from single-paradigm planners to multi-paradigm controllers.

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. Every JEPA world model trained from pixels for control today commits 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. Sub-billion JEPA world models commit to a single inference paradigm: PLDM [36], DINO-WM [42], and LeWM [29, 4] train action-conditional latent dynamics and plan via trajectory optimisation, while V-JEPA-2-AC [3] post-trains a 300M action head on a 1B video JEPA for direct behaviour cloning. Multi-paradigm checkpoints exist in a second cluster that operates in pixel space at billion-class scale: PAD, UVA, UWM, DUST, and Cosmos Policy [16, 22, 43, 39, 21] denoise observations and actions jointly and serve forward dynamics, inverse dynamics, and behaviour policy from one set of weights. The (sub-billion JEPA, multi-paradigm) cell is empty.

*Correspondence: `ru.rakhimov@t-tech.dev`. Project page: `https://corl-team.github.io/qantara`

Accepted at ICML 2026 Workshop on Decision-Making from Offline Datasets to Online Adaptation: Black-Box Optimization to Reinforcement Learning.

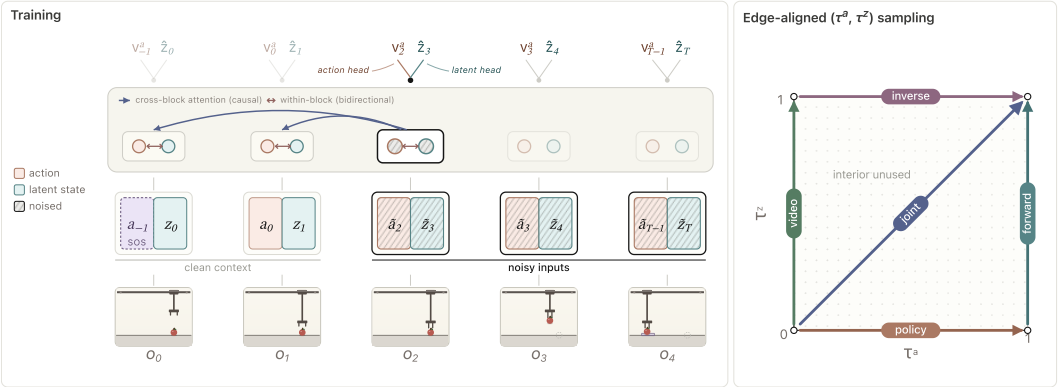

Figure 1: **Qantara training and edge-aligned** (τ^a, τ^z) **sampling.** *Left:* block-causal predictor on (a, z) tokens; a clean prefix conditions the noisy continuation and per block the action head emits an action velocity v^a while the state head emits a latent residual that yields $\hat{z}$. *Right:* 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, concentrating capacity on the regions queried at inference.

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. Single-paradigm training optimises one such combination and leaves the rest weak. Whether a single sub-billion JEPA can support all three at deployment without paying a cost on any of them is the open question.

In this paper, 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 [28, 24, 23, 2]. 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 by pairing a Brownian bridge on the state axis with edge-aligned (τ^a, τ^z) sampling on the action axis (§2). (2) On the LeWM-suite [29], Qantara averages 91.2 SR and sets a new SOTA on OGBench-Cube (+7.7 SR over DINO-WM); from the same checkpoint, the behaviour-cloning and video–inverse dispatches read off at $\sim$15–65$\times$ lower inference cost (§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 even on inference paths whose noise-square corner the dropped mode does not cover (§3.4).

2 Method

We present **Qantara**, 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. The section describes the world-action model: predictor architecture and the joint bridge-flow training objective pairing a Brownian-bridge interpolant on the state axis with flow matching on the action axis (§2.1); the three inference paradigms the trained predictor drives from a single set of weights (§2.2); and the edge-aligned (τ^a, τ^z) sampling that concentrates training capacity on the regions inference queries (§2.3). Figure 1 sketches the predictor (left) and 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 $t \in \{0, \ldots, T\}$, 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 $t' \in \{0, \ldots, T-1\}$, 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; intra-block bidirectionality bakes in no fixed conditioning order between the two, leaving the inference-time denoising order as a sampling choice.

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:

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

Each action and state token carries its own scalar $\tau \in [0, 1]$, indexed by the token's time (τ_t^a for action token a_t, τ_t^z 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 [32, 33]. 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 $\tau_t^a = 0$ source must be task-agnostic and we adopt the standard noise-to-data flow-matching interpolant [23, 26, 2, 12],

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

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 [2, 24, 28] gives

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

whose $\tau_{t+1}^z = 0$ 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):

$$(h_t^a,\, h_{t+1}^z) \;=\; f_\theta\big(\tilde{a}_{\le t},\, \tilde{z}_{\le t+1}\big). \tag{5}$$

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

$$\hat{z}_{t+1} = z_t + \mathrm{head}_z\big(h_{t+1}^z\big), \tag{6}$$

$$v_t^a = \mathrm{head}_a\big(h_t^a\big). \tag{7}$$

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 and action-axis flow-matching loss are

$$\mathcal{L}_{\mathrm{B}}^z = \mathbb{E}\, \|\hat{z}_{t+1} - z_{t+1}\|_2^2, \tag{8}$$

$$\mathcal{L}_{\mathrm{FM}}^a = \mathbb{E}\, \|v_t^a - (a_t - \varepsilon_a)\|_2^2, \tag{9}$$

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 [4], 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

$$\mathcal{L} \;=\; \lambda_z\, \mathcal{L}^z_{\mathrm{B}} \;+\; \lambda_a\, \mathcal{L}^a_{\mathrm{FM}} \;+\; \lambda_{\mathrm{SIG}}\, \mathrm{SIGReg}\big(\{z_0, \ldots, z_T\}\big), \tag{10}$$

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 [29]: 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. Without such a vector, the predictor reproduces the behaviour policy implicit in the training data: *behaviour-cloning sampling* denoises a_t directly from the action head with the prefix held clean, and *video–inverse composition* reaches the same target through two queries: predict the next latent under an action prior, then recover the action that produces the transition. With an encoded goal vector supplied, *latent planning* repurposes the predictor as a forward dynamics simulator and recovers an action sequence whose predicted terminal latent lies close to the goal.

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 1). 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 $p_\theta(a_t \,|\, z_{0:t},\, \tilde{z}_{t+1} = z_t)$ 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,

$$\tilde{a}^{(k+1)} \;=\; \tilde{a}^{(k)} \;+\; \tfrac{1}{K}\, v^a\big(\tilde{a}^{(k)},\, \tau^a_t = k/K\big), \qquad \tilde{a}^{(0)} \sim \mathcal{N}(0, I), \quad k = 0, \ldots, K{-}1, \tag{11}$$

giving $a_t = \tilde{a}^{(K)}$. At deployment we run $K = 2$ Euler steps over a context window of length $T = 4$ matching training. The joint diagonal ($\tau^a = \tau^z$, denoising both slots in lockstep) is a natural alternative but slightly underperforms the policy edge.

Video–inverse composition. The BC conditional $p_\theta(a_t \,|\, z_{0:t})$ also factorises through the next latent, as in UniPi's video-as-policy paradigm [11]; 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,

$$\underbrace{p_\theta(a_t \,|\, z_{0:t})}_{\text{BC dispatch}} \;=\; \int \underbrace{p_\theta(z_{t+1} \,|\, z_{0:t})}_{\texttt{video}\ \text{edge}}\, \underbrace{p_\theta(a_t \,|\, z_{0:t},\, z_{t+1})}_{\texttt{inverse}\ \text{edge}}\, \mathrm{d}z_{t+1}, \tag{12}$$

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 $\hat{z}_{t+1} \approx \mathbb{E}[z_{t+1} \,|\, z_{0:t},\, \tilde{z}_{t+1} = z_t]$, the action-marginal next latent under the training-time action prior. We follow the JEPA world model convention in calling this the `video` edge [42, 3, 29]; 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 $a_t \sim p_\theta(a_t \,|\, z_{0:t},\, \tilde{z}_{t+1} = \hat{z}_{t+1})$, 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 [29]. Each rollout step queries the `forward` edge ($\tau^a = 1, \tau^z : 0 \to 1$): 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 $\hat{z}_{t+1} \approx \mathbb{E}[z_{t+1} \,|\, z_{0:t},\, a_t,\, \tilde{z}_{t+1} = z_t]$ 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 $\{1, 2, 4\}$ in App. G). The search uses the Cross-Entropy Method [35] 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 1). Within each mode, the swept axis follows a per-block monotone-decreasing τ-chain: the cumulative product of i.i.d. uniforms over the T target blocks, so later blocks are noisier than earlier ones [41, 25]. A random prefix length $t' \sim \mathcal{U}\{1, \ldots, T\}$ per row pins the first t' blocks at $\tau = 1$, keeping every cold-start rollout horizon in-distribution [34, 44].

3 Experiments

3.1 Setup

Qantara trains a single checkpoint for goal-conditioned latent planning, behaviour-cloning sampling, and video–inverse composition (§2.1–§2.3). We evaluate on the *LeWM-suite* [29], the four-environment subset on which LeWM places PLDM, DINO-WM, and itself in a single end-to-end pixel-JEPA pipeline. We re-use this subset because it provides apples-to-apples baseline numbers (all four envs run by LeWM in one pipeline) and covers the diversity axes the multi-paradigm capability claim turns on. The only broader pixel JEPA-world-model benchmark, DINO-WM's six-environment evaluation [42], shares Push-T and Reacher with our suite, adds two further navigation envs (Maze and Wall) in the category Two-Room already represents, and adds two deformable-manipulation envs (Rope and Granular) in a dynamics class separate from the rigid-body envs that bear on the multi-paradigm question. Within the suite the four envs cross 2D vs. 3D observations, navigation vs. manipulation vs. motor control, and three classes of data-collection oracle (noisy heuristic, scripted expert, RL-trained policy), so no single data distribution dominates the suite average. The four envs are **Two-Room** [36] (2D navigation), **Push-T** [9] on the DINO-WM-released expert dataset [42] (contact-rich 2D manipulation), **OGBench-Cube** [31] (3D pick-and-place), and **Reacher-Hard** from the DeepMind Control suite [37] (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 trajectory coverage so its imagined futures land near goals. CEM planning is the most forgiving: in principle suboptimal but goal-spanning data suffices, though purely random exploration rarely reaches success regions in sparse-reward goal-reaching. The near-expert datasets above meet what all three modes need.

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. [29]'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. [29]: PLDM [36], DINO-WM [42] (no-proprio variant: pixel-only inputs, matching the proprio-free setting of the other baselines), and LeWM [29] itself. Our training and evaluation

Table 1: **LeWM-suite results.** Per-environment success rate; mean $\pm$ 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. [29], run under their evaluation pipeline. Qantara sets new SOTA on OGBench-Cube ($+7.7$ over DINO-WM, $+19.7$ over LeWM-paper) and reaches the Two-Room ceiling; our LeWM reproduction also trails the published LeWM numbers on Push-T and Reacher, indicating a protocol-level gap rather than a method-level one.

method	Push-T	Two-Room	Cube	Reacher
PLDM [36]	78.0 ± 5.0	97.0 ± 1.2	65.0 ± 2.8	78.0 ± 5.4
DINO-WM [42]	74.0 ± 4.5	100.0 ± 0.0	86.0 ± 4.7	79.0 ± 5.1
LeWM [29] (paper)	96.0 ± 4.0	87.0 ± 2.5	74.0 ± 3.0	86.0 ± 5.0
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	$\mathbf{100.0} \pm 0.0$	$\mathbf{93.7} \pm 0.7$	$\mathbf{80.9} \pm 1.8$

pipeline extends LeWM's code, so we additionally retrain LeWM end-to-end in this pipeline; this matches data, training scaffold, and eval-seed budget across the LeWM and Qantara rows, so the gap reflects method-level differences alone. PLDM and DINO-WM numbers are taken from Maes et al. [29].

Per-environment, Two-Room ties at ceiling with DINO-WM (both 100.0; $+13.0$ over LeWM-paper). On Push-T Qantara reaches 90.1, beating DINO-WM by $+16.1$ but trailing LeWM-paper's 96.0 by -5.9; on Reacher 80.9 beats DINO-WM by $+1.9$ but trails LeWM-paper's 86.0 by -5.1. Our LeWM reproduction at the published recipe also trails the paper on the same two envs (Push-T -3.7 SR, Reacher -21.7 SR; Table 1 bottom block), so the gap to published numbers symmetrically depresses both LeWM and Qantara cells under our protocol. Head-to-head against the LeWM reproduction, Qantara wins on three of four envs (Two-Room $+10.1$, Cube $+17.0$, Reacher $+16.6$) and trails by 2.2 SR on Push-T. The Cube cell carries the strongest signal: 93.7 ± 0.7 SR ($+7.7$ over DINO-WM, $+19.7$ over LeWM-paper) on a manipulation env where DINO-WM was the prior SOTA, with a margin that survives the protocol drift visible on the other cells.

3.3 The same checkpoint serves three inference paradigms

A single Qantara checkpoint serves all three inference paradigms with the same trained weights (§2.2). When the encoded goal z_g is supplied, latent planning rolls the predictor forward and searches actions whose terminal latent ends near z_g; when no goal vector is supplied, behaviour-cloning sampling and video–inverse composition produce a_t from the current observation and the prefix alone. Tab. 2 reports all three paths on the LeWM-suite, 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. Prior single-paradigm JEPA world models each commit to one of these paths at training time.

Table 2: **One Qantara checkpoint, three inference paradigms.** Mean SR $\pm$ std across 3 train seeds. Reacher omitted: its `qpos_match` success criterion is precision-bound and the goal-blind dispatches drop to near-noise (6.1 / 4.8 SR for BC / video–inverse); the CEM result is in Tab. 1. GCBC: goal-conditioned behaviour cloning [13], a goal-aware imitation baseline that anchors the Push-T 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

The goal-blind dispatches succeed when the current observation supplies enough information to recover an action that lands inside the env's success tolerance. On Push-T the canvas renders the target T pose alongside the current state, so a goal-blind sampler can read goal information directly off pixels: BC and video–inverse reach 82.1 and 83.2 SR, surpassing the goal-aware GCBC

reference (75) despite seeing the same input, while CEM uses z_g to recover the full 90.1 SR. On Two-Room and Cube the per-episode goal is set in the env state but is not rendered visually; the eval protocol initialises each episode within the goal-reaching demonstration window (App. A), so BC and video–inverse reach the goal by reproducing the expert's continuation whenever the env's success tolerance absorbs the imitation drift, achieving 69.1 / 70.8 SR (BC) and 69.3 / 72.8 SR (video–inverse) on Two-Room / Cube. 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 CEM path remains well-posed across all four envs through the $\|\hat{z}_H - z_g\|^2$ terminal cost. Routing z_g into the BC dispatch is the natural extension, deferred to follow-up. The goal-blind paths therefore carry the multi-paradigm claim only where the current observation exposes the goal or the env's success tolerance absorbs imitation drift (Push-T, Two-Room, Cube); on precision targets such as Reacher, planning remains the only reliable dispatch.

At the inference defaults, BC and video–inverse run at 17–24 ms per env-step versus CEM's 0.4–1.2 s, a $\sim$15–65$\times$ speedup (CEM cost scales with `n_steps`, set per env: Push-T uses 30, the others 10); decoded rollouts under all three dispatches stay coherent from the same checkpoint (App. H).

3.4 Which design choices are load-bearing

We ablate the recipe along three axes: the (τ^a, τ^z) sampling design (edges over uniform, §3.4; the 5-mode set, §3.4), the state-head residual+zero-init (§3.4), and robustness to the remaining knobs (γ, K; §3.4). The mode-set ablation gives our most surprising result: dropping any single mode destabilises Push-T training even on inference paths that do not query the dropped mode's region of the noise square.

Edge-aligned sampling beats uniform coverage. Uniform sampling on $[0, 1]^2$ 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 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. Holding compute per step fixed, we ablate the 5-mode design (§2.3) and evaluate all three inference paths per cell (Tab. 3; 10-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.

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: both paths are goal-blind (Tab. 2); on Cube they track the demo distribution insensitively to mode-set perturbations ($\sim$70 SR across all rows), and on Reacher they collapse to near-noise. Only the 5-mode reference and `drop-video` are collapse-free across all three Push-T inference paths.

	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 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†

- *Only `drop-video` preserves all three Push-T paths.* CEM, BC, and video–inverse stay within 2.8 SR of the 5-mode reference; the $(0, 0)$ corner where video–inverse starts denoising is still trained as the `joint` diagonal's lower endpoint, so the third path survives without a dedicated mode.
- *Dropping `policy`, `inverse`, or `joint` destabilises Push-T training across all three paths.* CEM pop-std exceeds 15 SR with 1–2 of three seeds collapsing to < 40 SR (sharpest: `drop-policy` 27.2 ± 32.5 vs `drop-video` 87.3 ± 1.7). The five modes co-regularise the shared trunk: `policy` stabilises CEM training (not just BC inference), `joint` stabilises CEM and BC (not just its diagonal). 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. Cube and Reacher hold steady across the four mode-LOO drops (within ~ 5 SR of the 5-mode reference); the 1-mode, 2-mode {`video`, `inverse`}, and uniform-sampler cells degrade them too (Tab. 3).
- *Neither 2-mode subset is self-sufficient.* {`forward`, `policy`} matches the reference on CEM and BC but video–inverse collapses (Push-T 7.5 SR); {`video`, `inverse`}, the symmetric minimal candidate for the third path, underperforms its target (Push-T 37.6 vs reference 83.2).

The case for keeping all five modes is therefore *Push-T training stability*, not preserving the third path (`drop-video` also preserves it). On the collapsed seeds, final $\mathcal{L}_{\mathrm{B}}^{z}$ plateaus at $\sim 10\times$ the reference and is rank-monotone with eval SR across all nine mode-LOO Push-T cells, so the instability is a training-time failure, not an eval-only artefact.

Output residual + zero-init is required everywhere. The state-head residual form (Eq. 6) pairs with zero-initialised final-layer weights so the predictor is the identity at initialisation ($\hat{z}_{t+1} = z_t$), keeping every iterate of the K-step CEM rollout on the training-time bridge marginal. Ablating only the residual at fixed bridge interpolant (head emits $\hat{z}_{t+1}$ directly with default Linear init) regresses suite average by 5.6 SR; Push-T drops by 15.2 SR with seed-level training instability (one of three seeds at 57.2 SR; App. E).

Defaults robust to K; off-canonical γ collapses. 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 [24] ties our $\gamma = 1$ stochastic-interpolant canonical [2] 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 our 5-mode reference recipe lifts the suite average from 84.9 to 88.3 SR (3 train seeds), and we keep $p = 0$ as our default.

4 Related work

Methods for visuomotor control via world models differ on three axes: future-state representation (JEPA latent / pixel-video), parameter scale ($\sim$ 10M to 7B), and inference paradigms served per checkpoint (one / many). Qantara occupies the JEPA-latent $\times$ sub-billion $\times$ multi-paradigm cell.

World models for control. JEPA-style world models predict environment dynamics in a learned embedding space: PLDM [36] trains end-to-end with VICReg, DINO-WM [42] freezes DINOv2, LeWM [29, 4] pairs next-embedding prediction with an isotropic-Gaussian regulariser, and V-JEPA-2-AC [3] post-trains a 300M action head on a 1B video JEPA. Recent extensions push the recipe to dexterous manipulation [14], one-shot imitation [15], masked latent interventions [30], value-aligned latent shaping [10], and JEPA world model design-choice studies [38]. Adjacent world models include TD-MPC2 [18, 19], DreamerV4 [17], and NWM [5]. Each of these models is trained for a single inference paradigm, whereas Qantara serves planning, behaviour cloning, and inverse dynamics from one checkpoint.

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, all in **pixel space**: PAD [16] (joint-denoising DiT), UVA [22] (decoupled video and action heads), UWM [43] (per-modality timesteps), DUST [39] (dual-stream MMDiT), WorldVLA / RynnVLA-002 [7, 6] (autoregressive token transformer), and Cosmos Policy [21] (post-trained 2B video model). GR-1 [40] is the GPT-style ancestor. Parameter counts span 150M to 7B; none predicts in an embedding space. Earlier trajectory-diffusion baselines (Diffuser [20], Decision Diffuser [1]) 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. Diffusion Forcing [8] introduces per-token noise levels; follow-ups add clean-prefix conditioning [44, 34].

Position. The two clusters above tile a (representation $\times$ paradigm-count) plane. *(JEPA-latent, single-paradigm)* = PLDM, DINO-WM, LeWM, V-JEPA-2-AC; *(pixel-video, multi-paradigm)* = PAD, UVA, UWM, DUST, WorldVLA, RynnVLA-002, Cosmos Policy at 150M to 7B. The *(JEPA-latent, multi-paradigm)* cell is empty in prior work; Qantara occupies it with a single ~ 21M-parameter checkpoint that serves latent planning, behaviour cloning, and inverse dynamics, one to two orders of magnitude smaller than the pixel-video multi-paradigm cluster. The closest neighbour is DINO-WM (same representation, single paradigm). Relative to the billion-parameter pixel–video models, Qantara trades raw capacity for low-latency deployment (17–24 ms per goal-blind step, §3.3).

5 Conclusion

Qantara trains a single ~ 21M-parameter JEPA world model and serves three inference paradigms from one set of weights: goal-conditioned latent planning, behaviour-cloning sampling, and video–inverse composition. On the LeWM-suite the planning path averages 91.2 SR and lifts OGBench-Cube to 93.7 SR, $+7.7$ over the prior JEPA world model SOTA (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). To our knowledge Qantara is the first sub-billion-parameter JEPA world model that defers the planning-vs-imitation choice to deployment.

Two design choices are responsible for the multi-paradigm capability. A Brownian-bridge interpolant between consecutive clean latents on the state axis matches the previous clean latent the planner already feeds in at $\tau^z = 0$, so the predictor need not learn a noise-to-latent map. Edge-aligned (τ^a, τ^z) sampling concentrates training capacity on the four corners that inference queries plus the diagonal between them. Replacing either piece is costly, and most of the cost falls on the most action-sensitive environment: switching to uniform $[0, 1]^2$ sampling at matched compute regresses Push-T CEM by 36.8 SR (§3.4); ablating the state-head residual + zero-init pair regresses it by 15.2 SR (§3.4).

The mode-set ablation 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 goal-conditioned planner whose noise-square corner the dropped mode does not cover (§3.4, Tab. 3). The five modes therefore co-regularise the shared trunk beyond their per-corner coverage role: even a deployment that will only ever query the planner benefits from training-time exposure to the behaviour-cloning and inverse-dynamics corners.

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. We expect the same rule to carry over to larger JEPA backbones [3] and longer-horizon forcing schemes [8, 34], and the corner co-regularisation effect to grow with the number of paradigms a single backbone serves.

Limitations. The capability claim is bounded along three axes. All three dispatches denoise one block per control step, and chunked multi-block inference [8, 34] is the direct extension. Evaluation is simulation-only on near-expert offline data, so real-robot transfer and natural-video pre-training [3] remain open. The recipe is shown only at the ~ 21M-parameter scale, and scaling to larger JEPA backbones [3] is untested.

References

[1] A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. 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.

[2] 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.

[3] M. Assran, A. Bardes, D. Fan, Q. Garrido, R. Howes, M. Muckley, A. Rizvi, C. Roberts, K. Sinha, A. Zholus, et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

[4] R. Balestriero and Y. LeCun. LeJEPA: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025.

[5] A. Bar, G. Zhou, D. Tran, T. Darrell, and Y. LeCun. Navigation world models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15791–15801, 2025.

[6] J. Cen, S. Huang, Y. Yuan, K. Li, H. Yuan, C. Yu, Y. Jiang, J. Guo, X. Li, H. Luo, F. Wang, D. Zhao, and H. Chen. RynnVLA-002: A unified vision-language-action and world model. *arXiv preprint arXiv:2511.17502*, 2025.

[7] J. Cen, C. Yu, H. Yuan, Y. Jiang, S. Huang, J. Guo, X. Li, Y. Song, H. Luo, F. Wang, D. Zhao, and H. Chen. WorldVLA: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

[8] B. Chen, D. M. Monso, Y. Du, M. Simchowitz, R. Tedrake, and V. 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.

[9] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

[10] M. Destrade, O. Bounou, Q. L. Lidec, J. Ponce, and Y. LeCun. Value-guided action planning with JEPA world models. *arXiv preprint arXiv:2601.00844*, 2026.

[11] Y. Du, M. Yang, B. Dai, H. Dai, O. Nachum, J. B. Tenenbaum, D. Schuurmans, and P. Abbeel. Learning universal policies via text-guided video generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[12] D. Gao, B. Zhao, A. Lee, I. Chuang, H. Zhou, H. Wang, Z. Zhao, J. Zhang, and I. Soltani. VITA: Vision-to-action flow matching policy. In *ICLR*, 2026. URL https://openreview.net/forum?id=BTe5VLBjPg.

[13] D. Ghosh, A. Gupta, A. Reddy, J. Fu, C. M. Devin, B. Eysenbach, and S. Levine. Learning to reach goals via iterated supervised learning. In *ICLR*, 2021. URL https://openreview.net/forum?id=rALA0Xo6yNJ.

[14] R. G. Goswami, A. Bar, D. Fan, T.-Y. Yang, G. Zhou, P. Krishnamurthy, M. Rabbat, F. Khorrami, and Y. LeCun. World models for learning dexterous hand-object interactions from human videos. *arXiv preprint arXiv:2512.13644*, 2025.

[15] R. G. Goswami, P. Krishnamurthy, Y. LeCun, and F. Khorrami. OSVI-WM: One-shot visual imitation for unseen tasks using world-model-guided trajectory generation. *arXiv preprint arXiv:2505.20425*, 2025.

[16] Y. Guo, Y. Hu, J. Zhang, Y.-J. Wang, X. Chen, C. Lu, and J. Chen. Prediction with action: Visual policy learning via joint denoising process. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[17] D. Hafner, W. Yan, and T. Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025.

[18] N. Hansen, H. Su, and X. 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.

[19] N. Hansen, H. Su, and X. Wang. Learning massively multitask world models for continuous control. In *ICLR*, 2026. URL https://openreview.net/forum?id=MPabX9LEds.

[20] M. Janner, Y. 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.

[21] M. J. Kim, Y. Gao, T.-Y. Lin, Y.-C. Lin, Y. Ge, G. Lam, P. Liang, S. Song, M.-Y. Liu, C. Finn, and J. Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.

[22] S. Li, Y. Gao, D. Sadigh, and S. Song. Unified video action model. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025.

[23] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *ICLR*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

[24] G.-H. Liu, A. Vahdat, D.-A. Huang, E. A. Theodorou, W. Nie, and A. Anandkumar. I2sb: 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.

[25] K. Liu, W. Hu, J. Xu, Y. Shan, and S. Lu. Rolling forcing: Autoregressive long video diffusion in real time. In *International Conference on Learning Representations (ICLR)*, 2026.

[26] X. Liu, C. Gong, and Q. 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.

[27] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[28] Z. Lyu, M. Li, J. Jiao, and C. 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.

[29] L. Maes, Q. Le Lidec, D. Scieur, Y. LeCun, and R. Balestriero. LeWorldModel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2025.

[30] H. Nam, Q. L. Lidec, L. Maes, Y. LeCun, and R. Balestriero. Causal-JEPA: Learning world models through object-level latent interventions. *arXiv preprint arXiv:2602.11389*, 2026.

[31] S. Park, K. Frans, B. Eysenbach, and S. 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.

[32] W. S. Peebles and S. 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.

[33] E. Perez, F. Strub, H. D. Vries, V. Dumoulin, and A. 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.

[34] R. Po, Y. Nitzan, R. Zhang, B. Chen, T. Dao, E. Shechtman, G. Wetzstein, and X. 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.

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

[36] V. Sobal, W. Zhang, K. Cho, R. Balestriero, T. G. J. Rudner, and Y. 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=j0N7H6A9UU`.

[37] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. de Las Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. P. Lillicrap, and M. A. Riedmiller. DeepMind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[38] B. Terver, T.-Y. Yang, J. Ponce, A. Bardes, and Y. LeCun. What drives success in physical planning with joint-embedding predictive world models? *arXiv preprint arXiv:2512.24497*, 2025.

[39] J. Won, K. Lee, H. Jang, D. Kim, and J. Shin. DUST: Dual-stream diffusion for world-model augmented vision-language-action model. *arXiv preprint arXiv:2510.27607*, 2025.

[40] H. Wu, Y. Jing, C. Cheang, G. Chen, J. Xu, X. Li, M. Liu, H. Li, and T. Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[41] D. Xie, Z. Xu, Y. Hong, H. Tan, D. Liu, F. Liu, A. E. Kaufman, and Y. 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`.

[42] G. Zhou, H. Pan, Y. LeCun, and L. 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`.

[43] C. Zhu, R. Yu, S. Feng, B. Burchfiel, P. Shah, and A. 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`.

[44] H. Zhu, M. Zhao, G. He, H. Su, C. Li, and J. Zhu. Causal forcing: Autoregressive diffusion distillation done right for high-quality real-time interactive video generation. *arXiv preprint arXiv:2602.02214*, 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** [36] is 2D continuous navigation through a wall-with-door (10k trajectories, average 92 steps, generated by a noisy door-then-target heuristic). **Push-T** [9] is contact-rich 2D manipulation reorienting a T-block; we use the DINO-WM-released dataset [42] of 20k human-designed expert demonstrations averaging 196 steps. **OGBench-Cube** [31] is 3D pick-and-place, 10k trajectories of 200 steps generated by the OGBench scripted data-collection policy. **Reacher-Hard** from the DeepMind Control suite [37] requires precise two-joint alignment, 10k trajectories of 200 steps collected with a Soft Actor-Critic expert policy.

B Optimisation and training-time hyperparameters

We train with AdamW [27]: 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 $5\cdot B=640$. 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 ≈ 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,

$$\tau[t] \;=\; \prod_{i \le t} u_i, \qquad u_i \sim \mathcal{U}[0,1], \tag{13}$$

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.

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: **Brownian-bridge γ sweep at the default recipe** (multi-seed, $K = 1$). $\gamma = 1$ is the Albergo et al. stochastic-interpolant canonical [2]; $\gamma = \sqrt{2}$ is the Schrödinger-bridge canonical [24]; 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. ‡mean masks seed-level collapse.

γ	Push-T	Two-Room	Cube	Reacher
$0^{\ddagger}$	31.5 ± 43.4	99.6 ± 0.0	92.5 ± 0.9	78.3 ± 3.4
$0.5^{\ddagger}$	60.1 ± 48.6	99.9 ± 0.2	93.6 ± 0.7	78.8 ± 1.2
1 (Albergo SI canonical, default)	$\mathbf{89.3} \pm 1.7$	$\mathbf{100.0} \pm 0.0$	$\mathbf{93.3} \pm 0.8$	80.5 ± 3.7
$\sqrt{2}$ (Schrödinger-bridge canonical)	89.3 ± 0.2	$\mathbf{100.0} \pm 0.0$	91.9 ± 2.0	$\mathbf{80.8} \pm 3.2$
$2^{\ddagger}$	53.6 ± 44.9	71.5 ± 49.4	$\mathbf{93.6} \pm 0.4$	79.6 ± 3.8

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

To complete the factorial we additionally 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 long-horizon envs where the previous latent is a stronger prior on the next, while the output-side residual stabilises training across all envs unconditionally.

Table 5: **State-head and bridge structural ablations, multi-seed.** The output-side residual head (row 1 vs row 2) is unconditionally 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 the longest-horizon env: restoring it yields $+12.7$ SR on Reacher, within-noise elsewhere. The two priors are complementary, not redundant.

bridge	residual	Push-T	Two-Room	Cube	Reacher
on	on (zero-init; default)	**89.3** $\pm$ 1.7	100.0 $\pm$ 0.0	**93.3** $\pm$ 0.8	**80.5** $\pm$ 3.7
on	off (default Linear init)	74.1 $\pm$ 12.3	100.0 $\pm$ 0.0	86.3 $\pm$ 1.3	80.4 $\pm$ 2.4
off	off (default Linear init)	86.5 $\pm$ 0.5	100.0 $\pm$ 0.0	87.1 $\pm$ 0.8	67.7 $\pm$ 1.0

F CEM and MPC configuration

We inherit the CEM and receding-horizon MPC setup from LeWM [29], which in turn follows DINO-WM [42]. CEM uses $N = 300$ Gaussian candidates per iteration, 30 refit iterations, 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. Identical settings are used across all 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). At intermediate $\tau \in (0, 1)$ the sampler injects bridge-marginal noise $\gamma\sqrt{\tau(1-\tau)}\xi$ after each bridge re-projection so each iteration's input lies on the training-time marginal (§2.2); at $K = 1$ no intermediate τ is reached and the sampler reduces to the deterministic single-call. 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 $\pm$ 1.3	100.0 $\pm$ 0.0	93.3 $\pm$ 0.7	80.5 $\pm$ 3.0
1	2	88.0 $\pm$ 0.9	100.0 $\pm$ 0.0	93.3 $\pm$ 2.1	80.1 $\pm$ 1.9
1	**4** (default)	**90.1** $\pm$ 1.1	100.0 $\pm$ 0.0	**93.7** $\pm$ 0.7	**80.9** $\pm$ 1.8
1	8	87.2 $\pm$ 1.4	100.0 $\pm$ 0.0	93.2 $\pm$ 0.6	78.7 $\pm$ 1.9
$\sqrt{2}$	1	89.3 $\pm$ 0.2	100.0 $\pm$ 0.0	91.9 $\pm$ 1.6	80.8 $\pm$ 2.6
$\sqrt{2}$	2	90.4 $\pm$ 2.0	100.0 $\pm$ 0.0	93.1 $\pm$ 2.2	79.3 $\pm$ 0.5
$\sqrt{2}$	4	89.7 $\pm$ 1.5	100.0 $\pm$ 0.0	92.3 $\pm$ 2.8	80.0 $\pm$ 2.3
$\sqrt{2}$	8	88.9 $\pm$ 1.3	100.0 $\pm$ 0.0	92.0 $\pm$ 2.7	77.3 $\pm$ 3.0

H Multi-paradigm decoded rollouts

What does the same checkpoint produce under each of the three dispatches? Starting from one context frame at $t = 0$, we roll the predictor forward six steps under three action sources (Fig. 2). 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. [29, Fig. 7].

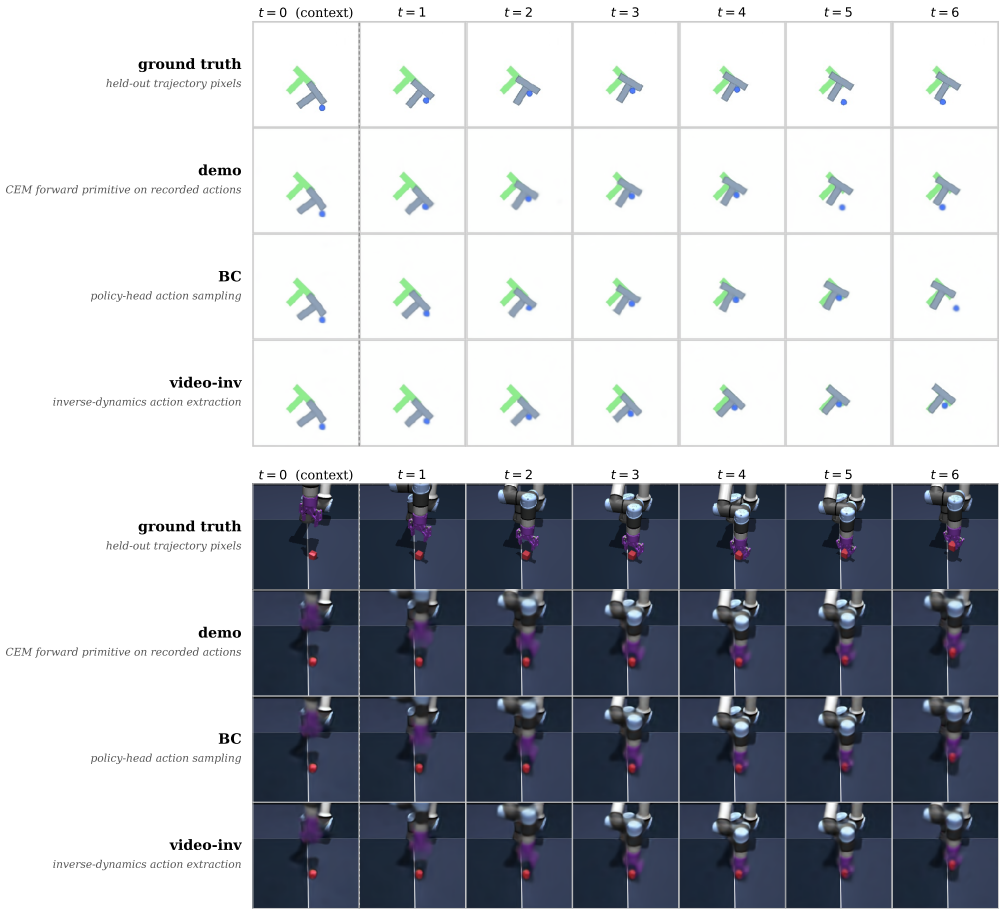

Figure 2: **Same checkpoint, three action sources, three coherent decoded rollouts** (Push-T top, Cube bottom; Two-Room and Reacher continued in Fig. 2). Columns are timesteps $t = 0-6$; the dashed separator marks the boundary between the encoded context ($t = 0$) and the open-loop predictor rollout ($t = 1-6$). 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.

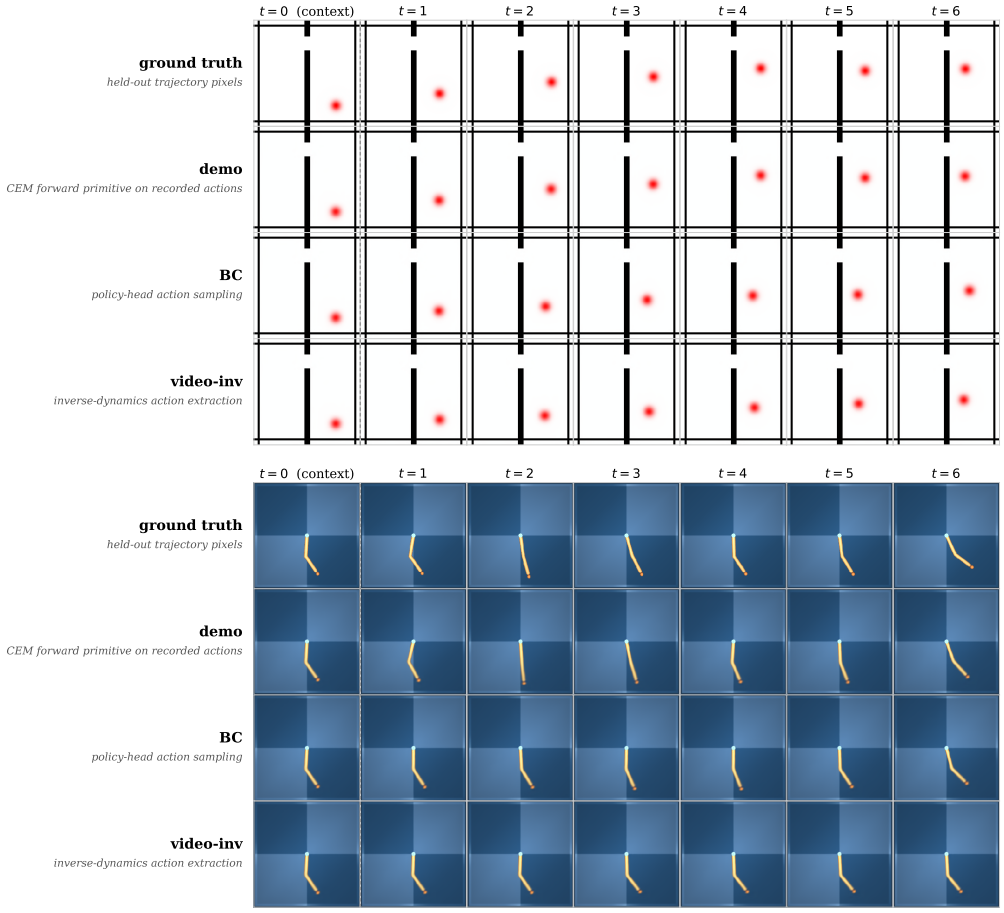

Figure 2: (Continued.) Two-Room (top) and Reacher (bottom) decoded rollouts under the same three action sources; layout and column semantics as in Fig. 2.