DOES LATENT PLANNING SURVIVE POINT CLOUDS? ACTION-CONDITIONED JEPA WORLD MODELS FOR GEOMETRIC OBSERVATIONS AND GOALS

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent action world models let agents plan new behaviors at test time by predicting how actions change the environment, and joint-embedding predictive architectures (JEPAs) do so by forecasting future latent states rather than pixels. Yet nearly all such models see the world through a camera, even though robotic manipulation is fundamentally geometric: in robotics goals for manipulation are traditionally specified by target object poses, not by images of the object once placed. We ask whether latent planning survives a shift from appearance to geometry, on the observation side as well as on the goal specifications side. To answer this, we extend the stable-worldmodel evaluation platform with simulated LiDAR-style raycast point clouds as a new sensor modality, and adapt three JEPA designs to point clouds: a frozen-encoder model built on Utonia features, a distribution-prior model based on LeWM, and an action-sensitive model based on Delta-JEPA. We further introduce a goal-encoding mechanism that constructs the goal latent from the current latent and a 3D target pose, removing the need for goal images or goal point clouds. A comparative evaluation of the different anti-collapse mechanisms shows that point-cloud world models can match their image-based counterparts, demonstrating that the modality shift from appearance to geometry is achievable. All models are released as open weights with open-source training and inference code, to make world-model planning accessible for LiDAR-driven and pose-directed robotic tasks.

1 INTRODUCTION

World models learn to predict how actions change the environment, allowing agents to plan new behaviors at test time instead of being trained for each task (Ha & Schmidhuber, 2018; Hafner et al., 2019; LeCun, 2022). Joint-embedding predictive architectures (JEPAs) made this practical by predicting future *latent* states conditioned on actions rather than generating observations, whether over frozen features (DINO-WM (Zhou et al., 2025)), distribution-prior (LeWM (Maes et al., 2026b)), or action-sensitive (Delta-JEPA (Zhang et al., 2026b), SMWM (Ivashkov et al., 2026)). And with its standardized simulated environments, and planners, stable-worldmodel (SWM) makes world-model results comparable. However, almost all of the existing works see the world through a camera.

Images are a rational default: cheap sensors, dense semantics, and transferable foundation models (Kirillov et al., 2023; Oquab et al., 2023; Radford et al., 2021; Schwingshackl et al., 2025). But most robotic manipulation is inherently geometric, and pose is its operative quantity (Thalhammer et al., 2024; Newbury et al., 2023): pick-and-place is specified by a target pose of the object, not by an image of the object once placed (Zeng et al., 2020; Sundermeyer et al., 2021), and success is a matter of where things are, not what they look like. An appearance-driven planner, by contrast, needs a goal image to encode, whether rendered, demonstrated, or generated (Nair et al., 2018; Du et al., 2023; Black et al., 2024). Latent planners relax this by comparing latents rather than pixels, lowering the bar on goal-image quality, but the comparison is still appearance-based.

Moreover, images are only one of many sensing modalities in robotics. Robots also perceive the world geometrically, and detection on raw range scans has matured accordingly, progressing from 3D

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.

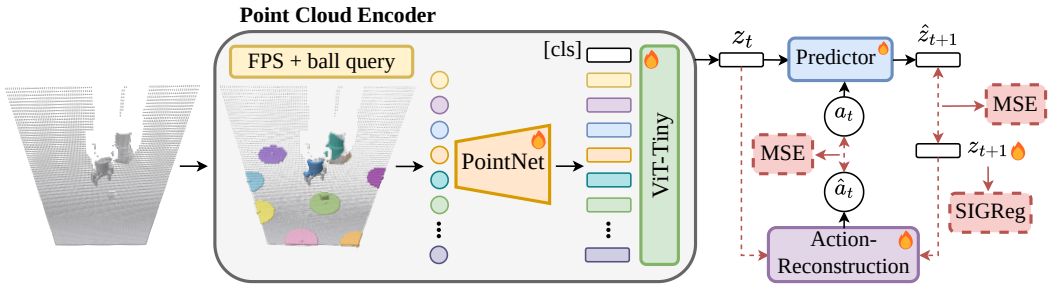


Figure 1: **Encoder of Point-LeWM** and **Point-Delta-JEPA**: Furthest point sampling (FPS) + radius query (only a few queries are shown), PointNet (Qi et al., 2017a) tokenizer, and ViT-Tiny whose class token (CLS) yields z_t. The next latent z_{t+1}, from the same encoder, is matched to $\hat{z}_{t+1}$ via MSE. SIGReg (Point-LeWM) and the Action-Reconstruction-Loss (Point-Delta-JEPA) prevent collapse.

bounding boxes (Misra et al., 2021) to 6-DoF and parametric object poses (Oberweger et al., 2026; Schwingshackl et al., 2026).

In an effort to make planning with latent action world models more accessible to roboticists, we propose shifting from an appearance-based to a geometry-based paradigm. This is accomplished by (a) using LiDAR-style raycasted point clouds as input for the adapted JEPA architecture (see Figure 1), and (b) using goal latents derived from 3D target poses as explicit goal specification (Figure 2). But this opens up the driving research question behind this work: Does latent planning survive this shift from appearance to point clouds?

Existing 3D world models do not answer it: driving models generate future scans or occupancy judged by geometric fidelity (Zhang et al., 2024; Zheng et al., 2024; Yang et al., 2024; Liu et al., 2025), manipulation models decode explicit future geometry (Huang et al., 2026; Peri et al., 2026), and AD-L-JEPA (Zhu et al., 2026) predicts latents but is action-free. To our knowledge, no prior work learns an action-conditioned latent-prediction world model on point clouds and plans with it.

We build that missing piece. First, we create the necessary data source by extending the established SWM evaluation platform (Maes et al., 2026a) with simulated LiDAR-style raycast point clouds as input modality (Section 3.1). Then, we build action-conditioned world models for point clouds for three JEPA designs: frozen-encoder (DINO-WM-like), distribution-prior (LeWorldModel-like), and action-sensitive (Delta-JEPA-like) (Section 3.2.1). We add a novel goal-encoding mechanism avoiding goal images or goal point clouds (Section 3.2.2) and plan with the SWM-MPC. We isolate each of their respective anti-collapse mechanisms and evaluate pointcloud-based vs. image-based approaches (Section 4). In Section 5, we conclude that comparable results can be achieved with point clouds and therefore that the modality shift from appearance to geometry can be achieved. Our contributions are:

- **LiDAR for SWM**: Simulated LiDAR-like point clouds as a new sensor modality for the SWM platform and benchmark.
- **Point-LeWM and Point-Delta-JEPA**: Point-cloud adaptations of LeWM and Delta-JEPA, which use the anti-collapse approaches distribution-prior and action-sensitive, respectively.
- **Utonia-WM**: A novel point-cloud JEPA world model built on a frozen Utonia encoder (Zhang et al., 2026a), and a matched voxel-feature baseline, Vox-WM, that isolates the effect of the pretrained features.
- **Planning toward 3D targets**: Encoding the goal latent from the current latent and a target pose, replacing goal images.
- **Point-cloud vs. image-based world models**: A comparative evaluation of all anti-collapse approaches on LiDAR-enabled SWM, showing that point-cloud models can match their image-based counterparts.

By providing all models as open weight and source (training and inference) under `double-blind.com` we hope to make the classically very image driven world model capabilities accessible to the robotics community interested in exploiting them for LiDAR and pose-directed tasks.

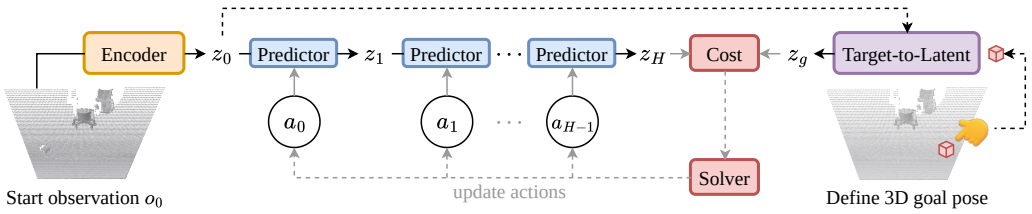


Figure 2: **Planning toward 3D goal poses.** The target-to-latent module enables specifying a goal directly as a 3D pose. This pose is encoded into a goal latent, which the planner then optimises against, removing the need for a goal observation.

2 Related Work

JEPA world models. Where classical world models learn latent dynamics through reconstruction (Ha & Schmidhuber, 2018; Hafner et al., 2019) or reward prediction (Hansen et al., 2024), JEPA-style world models (LeCun, 2022) learn them by predicting future representations directly, without a decoder or reward. Trained end-to-end, latent prediction alone collapses, and the reward-free designs differ in the anti-collapse route they take. DINO-WM (Zhou et al., 2025) predicts the tokens of a frozen DINOv2 (Oquab et al., 2023) encoder. PLDM (Sobal et al., 2025) and LeWorldModel (Maes et al., 2026b) regularize the latent distribution, the latter with a single term (SIGReg (Balestriero & LeCun, 2025)). SMWM (Ivashkov et al., 2026) and Delta-JEPA (Zhang et al., 2026b) add an inverse-dynamics loss on consecutive latents or their displacement, so latents must discriminate between actions. The SWM platform (Maes et al., 2026a) consolidates the first three and their planners into the interface we re-sense. Our Utonia-WM, Point-LeWM and Point-Delta-JEPA carry the three anti-collapse routes to point clouds. Utonia-WM takes the frozen-encoder route of DINO-WM with Utonia (Zhang et al., 2026a), a reusable backbone pretrained on diverse point-cloud data.

World models on point clouds. To the best of our knowledge, none of the existing work plans in latent space. *Manipulation* models regress future geometry: PointWorld (Huang et al., 2026) predicts per-point displacements and drives MPPI on a real arm, 3D Point World Models (Peri et al., 2026) place point completion before the dynamics, and both plan over explicit predicted point clouds. *Driving* models are generative, synthesizing future scans or occupancy (Zhang et al., 2024; Zheng et al., 2024; Yang et al., 2024; Liu et al., 2025) without searching over actions. Closest to us, AD-L-JEPA (Zhu et al., 2026) brings JEPA to automotive LiDAR, but as a representation learner. It predicts masked bird's-eye-view features within one scan, so neither time nor actions enter, and its encoder is fine-tuned for object detection instead of being queried by a planner. Its temporal extension (Zhu & Choromanska, 2026) forecasts occupancy latents, with SIGReg, but is also action-free.

Beyond this gap, point clouds are a difficult domain for latent prediction, which admits a trivial optimum at temporally constant features (Sobal et al., 2022), and point clouds are full of such features (further referred to as *slow-feature shortcut*). Height and surface normals follow directly from the coordinates, and Wu et al. (2025) show that 3D self-supervised encoders rely on these cues instead of learning semantics, which they call the *geometric shortcut*. In our scans 80–98% of returns lie on a static ground plane and only 0.3–15% on anything that moves (Table 1). The most prominent cues are thus also the most constant ones, promoting the slow-feature shortcut. Whether the mechanisms we study survive this compounding is the empirical question of Section 4.

Goal specification and planning. Sampling-based model predictive control (MPC) is the common planning strategy of latent world models (Hafner et al., 2019; Hansen et al., 2024; Zhou et al., 2025), and in the reward-free setting the goal enters as an *encoded goal observation* (Maes et al., 2026a). Work that improves the search itself (Wang et al., 2026; Cheng et al., 2026; Nguyen et al., 2026) keeps the goal observation, presuming that such an observation exists. FF-JEPA (Masip et al., 2026) removes goal images by forecasting latent subgoals, but its objective is implicit in the demonstrations, so the planner cannot be commanded to an arbitrary target. In 3D, PointWorld (Huang et al., 2026) plans toward target positions for a selected subset of points, a sparse goal cloud with known correspondences that only a model predicting explicit geometry can evaluate. Our target-to-latent module keeps an explicit, controllable goal. It is specified by the 3D pose of the

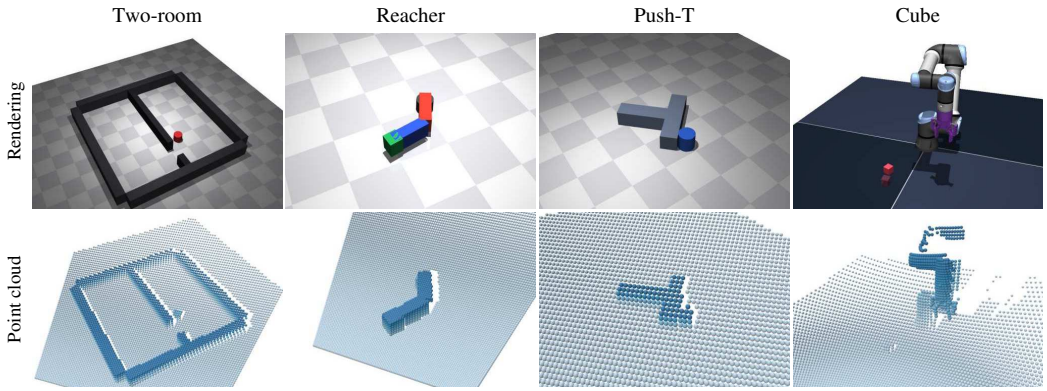


Figure 3: **From image environments to point cloud environments.** Top: the 3D scene. Bottom: the generated point cloud of that same scene, which replaces the image as the observation.

task-relevant object, and constructs the goal latent from that pose and the current latent, giving latent planners the coordinate interface of geometry-predicting models without decoding geometry.

3 METHODS

Section 3.1 discusses how simulated LiDAR is added to the SWM environments. Section 3.2 builds the three JEPA world-model designs on the resulting point clouds and introduces a complementary module that removes the need for a global goal observation at inference time.

3.1 POINT-CLOUD SENSING AND DATASET REPLICATION

Since our work is a direct successor to LeWM, we consider the same environments: OGB-Cube (Park et al., 2025), Two-Room (Sobal et al., 2025), Push-T (Chi et al., 2023), and Reacher (Todorov et al., 2012). To compare observation modalities rather than different datasets, we *re-sense* the released image benchmarks. Specifically, we attach a raycast range sensor to the corresponding simulator, replay every recorded state, and store the resulting point cloud alongside the original observation. Thus, episodes, episode lengths, actions, and goals are identical to those of the image datasets (Table 6).

Sensor. The sensor casts a fixed 100×100 grid of rays matched to its mount camera's field of view, recording each ray's first intersection within a per-environment maximum range. Returns are expressed in the sensor frame (x forward, y left, z up), and rays intersecting nothing take a sentinel value, so each frame is a fixed $(10{,}000, 3)$ array. The sensor is idealized, exact intersections, no noise, dropout or sweep distortion, so the comparison isolates observation geometry. Only the OGBench cube provides a mountable camera. The others offer a top-down view alone, from which a scan reduces to a height map. Hence, for these environments, we place the sensor at $45°$ azimuth and elevation, derived from the scene's bounding sphere rather than tuned, to create more realistic LiDAR scene simulations.

Table 1: **The four re-sensed datasets.** *Ground share* and *moving returns* are the fractions of returns on the ground plane and on non-static entities.

	OGB-Cube	Two-Room	Push-T	Reacher
Ground share	∼79%	∼80%	∼98%	∼97%
Moving returns	∼15.0%	∼0.3%	∼0.5%	∼2.0%

Dataset conversion. We set the simulator to each state recorded in the original dataset and cast a LiDAR scan, adding the point cloud alongside the original data. Two-Room and Push-T are not 3D simulators, so we leave each engine unmodified and mirror its state into a lightweight

MuJoCo (Todorov et al., 2012) scene that serves only as geometry for the sensor (Appendix C). Dynamics, rewards, and termination are still computed by the original environment. At evaluation time, the same sensor scans each state the planner visits. In the resulting clouds, ground returns dominate and only 0.3–15% of points fall on moving entities (Table 1), which poses two distinct challenges for latent prediction (Section 2).

3.2 Point cloud world models and planning toward 3D goal poses

We adopt the offline, reward-free setting of prior JEPA world models (Maes et al., 2026b): the training data is a set of n action-interleaved trajectories $\mathcal{D} = \{(o_1, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{T_i-1}, o_{T_i})\}_{i=1}^{n}$, without rewards or task labels. Each observation o_t is a point cloud, an unordered point set in $\mathbb{R}^3$ of frame-varying size, and each action $\boldsymbol{a}_t$ a continuous control input. At test time the model must plan actions that reach a specified goal, conventionally given as a *goal observation* o_g (Maes et al., 2026a; Zhou et al., 2025). Our target-to-latent module will instead assume training-time annotations $\boldsymbol{c}_t$ of low-dimensional 3D targets, the only supervision used in this paper.

3.2.1 Model architecture

All models follow the JEPA template (LeCun, 2022): an encoder f_θ maps each observation to a latent representation, and a predictor p_ϕ predicts the next latent from a context of ℓ latents and actions,

$$\boldsymbol{z}_t = f_\theta(o_t) \in \mathcal{Z}, \qquad \hat{\boldsymbol{z}}_{t+1} = p_\phi(\boldsymbol{z}_{t-\ell+1:t}, \boldsymbol{a}_{t-\ell+1:t}). \tag{1}$$

Observations are never reconstructed. Point-LeWM and Point-Delta-JEPA use the same encoder and predictor architectures and are trained end-to-end, differing only in their objectives. This controlled setup isolates the effect of their respective anti-collapse mechanisms. In contrast, Utonia-WM does not learn an encoder.

Point-LeWM & Point-Delta-JEPA. For the end-to-end pair, f_θ adapts LeWorldModel's pixel encoder to point clouds (Figure 1): a Point-BERT-style set tokenizer (Yu et al., 2022; Zhou et al., 2024) feeds the image model's ViT-Tiny trunk (Dosovitskiy et al., 2021), whose CLS token gives $\mathcal{Z} = \mathbb{R}^{192}$. The predictor is LeWorldModel's causal action-conditioned transformer (Maes et al., 2026b). We make three changes to standard recognition practice. Coordinates are normalized using a *fixed* workspace affine rather than per cloud, since per-cloud statistics would remove the absolute object positions that define the task signal. Points are grouped by Euclidean radius (Qi et al., 2017b) rather than k-NN, so each token covers a fixed physical scale. Finally, the encoder uses only LayerNorm, following R3D's fix (Hong et al., 2026) for the BatchNorm-induced 3D "scaling paradox" in control.

Utonia-WM. Utonia-WM (Figure 4) follows DINO-WM (Zhou et al., 2025): a frozen pretrained encoder supplies the state, and only the predictor trains. Our frozen encoder is Utonia (Zhang et al., 2026a), a Point Transformer V3 (Wu et al., 2023) consuming point coordinates alone. Unlike image patch tokens, Utonia's super-points have no persistent identity across frames, making DINO-WM's token-wise prediction undefined. We therefore rotate each point cloud into the gravity-aligned frame expected by Utonia, encode it with the frozen backbone, and pool the resulting features into a fixed voxel grid, yielding $\approx$256 tokens. Each pooled token is then compressed from its 1152 pooled dimensions to the 192-dimensional predictor width by a fixed random orthogonal projection (Appendix A.2). The projection is fixed and training-free, while approximately preserving distances by the Johnson–Lindenstrauss lemma (Johnson & Lindenstrauss, 1984). These tokens are predicted by DINO-WM's frame-block-causal transformer (Zhou et al., 2025). Nothing downstream of the backbone is trainable, so the frozen latent space cannot collapse under the prediction loss.

Vox-WM. Vox-WM replaces the frozen Utonia features with 25 point statistics per voxel (occupancy, count, and per-axis moments; Appendix A.3), while keeping the same gravity-aligned frame, voxel grid, predictor, and fixed projection to 192 dimensions. Thus, the two models differ only in their voxel features, isolating the contribution of Utonia's learned representation.

Joint training objectives. All models minimize the teacher-forced latent prediction loss (Zhou et al., 2025; Maes et al., 2026b)

$$\mathcal{L}_{\text{pred}}(\theta, \phi) = \mathbb{E}_{\mathcal{D}}\Big[\big\|p_\phi(\boldsymbol{z}_{t-\ell+1:t}, \boldsymbol{a}_{t-\ell+1:t}) - \boldsymbol{z}_{t+1}\big\|_2^2\Big], \tag{2}$$

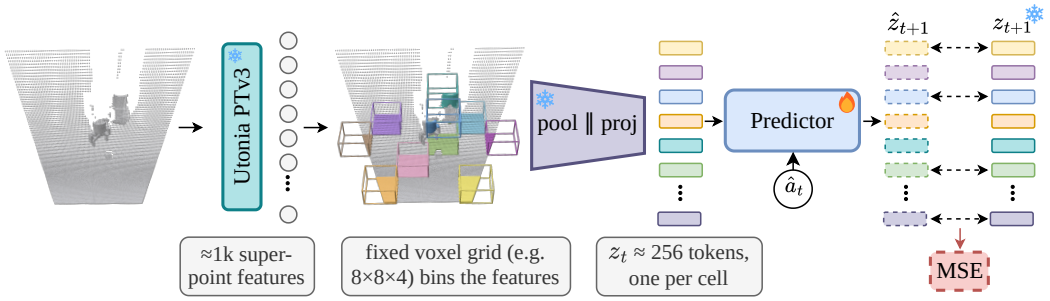


Figure 4: **Utonia-WM**: frozen Utonia (Zhang et al., 2026a) encodes the point cloud into super-point features, which a fixed canonical-frame voxel grid bins into ≈256 tokens (only a few cells shown). Per cell, features are pooled and compressed by a fixed random orthogonal projection to the predictor width. Only the predictor trains, with a token-wise MSE against the frozen targets $\boldsymbol{z}_{t+1}$.

i.e. the prediction of equation 1 must match the encoded next observation. Trained end-to-end, this loss alone leads to a collapsed solution (Sobal et al., 2022). Point-LeWM (P-LeWM) and Point-Delta-JEPA (PD-JEPA) avoid collapse by adding different auxiliary objectives to equation 2:

$$\mathcal{L}_{\text{P-LeWM}}(\theta, \phi) = \mathcal{L}_{\text{pred}} + \lambda\, \Omega_{\text{SIGReg}}(\theta), \tag{3}$$

$$\mathcal{L}_{\text{PD-JEPA}}(\theta, \phi, \psi) = \mathcal{L}_{\text{pred}} + \beta\, \mathbb{E}_{\mathcal{D}}\Big[\frac{1}{N}\big\|h_\psi(\boldsymbol{z}_{t+N} - \boldsymbol{z}_t) - \boldsymbol{a}_{t:t+N-1}\big\|_2^2\Big]. \tag{4}$$

In equation 3, the SIGReg regularizer (Balestriero & LeCun, 2025; Maes et al., 2026b), weighted by λ, drives the latent distribution toward an isotropic Gaussian, preventing collapse. In equation 4, a training-only decoder h_ψ, discarded after training and weighted by β, must recover the $N{=}5$ executed actions from the change in latent representation, yielding latents that discriminate between actions (Zhang et al., 2026b; Ivashkov et al., 2026). Utonia-WM needs neither term: only the predictor is trained against the fixed targets of the frozen encoder, so its representation cannot adapt to the task.

Implementation. All models are implemented in PyTorch (Paszke et al., 2019) and trained on two A100 GPUs within a 72-hour training budget per run. The shared end-to-end encoder is a ViT-Tiny trunk of ≈5.5M parameters with a 192-dimensional latent, trained for 10 epochs, whereas Utonia-WM pairs its frozen ≈137M-parameter backbone with a predictor trained for 100 epochs on precomputed features. All models step at frameskip 5 and plan from a context of $\ell = 3$ past latents. We reimplemented image Delta-JEPA and DINO-WM from their official implementations, adjusting batch size and learning rate to the 72-hour budget. For LeWM, we evaluate the released checkpoint. Under our evaluation protocol all three land at or near their published success rates (Appendix A.7), with retrained Delta-JEPA matching or exceeding its published numbers on every benchmark. More implementation details are provided in Appendix A.

3.2.2 Planning toward 3D goal poses

As in the pixel models (Maes et al., 2026b; Zhou et al., 2025), we plan with receding-horizon MPC in latent space, using the cross-entropy method (CEM) to find action sequences of horizon H that minimize $\|\hat{\boldsymbol{z}}_{t+H} - \boldsymbol{z}_g\|_2^2$. Here $\hat{\boldsymbol{z}}_{t+H}$ is the autoregressive rollout of the predictor in equation 1, and $\boldsymbol{z}_g$ is a goal latent, conventionally an encoded goal observation $\boldsymbol{z}_g = f_\theta(o_g)$. In robotics, however, goals are usually given as target placements of the robot or objects. To avoid requiring a scan of an unvisited state, we instead specify the goal as a low-dimensional 3D target $\boldsymbol{c}_g \in \mathcal{C}$ containing only task-relevant coordinates: the cube position in OGB-Cube, the agent position in Two-Room, the agent and block positions plus the block's planar angle in Push-T, and the joint positions in Reacher.

With f_θ frozen, we then train a target-to-latent mapping $m_\xi : \mathcal{C} \times \mathcal{Z} \to \mathcal{Z}$ on pairs of frames from the same trajectory, separated by a random offset $k \le k_{\max}$:

$$\mathcal{L}_{\text{goal}}(\xi) = \mathbb{E}_{(o_t,\, \boldsymbol{c}_{t+k},\, o_{t+k})\sim\mathcal{D},\; k\sim\mathcal{U}\{1,\ldots,k_{\max}\}}\Big[\big\|m_\xi(\boldsymbol{c}_{t+k}, \boldsymbol{z}_t) - \boldsymbol{z}_{t+k}\big\|_2^2\Big], \tag{5}$$

where $\boldsymbol{z}_t = f_\theta(o_t)$ and $\boldsymbol{z}_{t+k} = f_\theta(o_{t+k})$. A target alone cannot determine a latent, because f_θ encodes the whole scene while $\boldsymbol{c}_g$ fixes only a few coordinates. The current latent $\boldsymbol{z}_t$ supplies most

Table 2: **Planning success rate (%)**. Mean$_{\pm \text{sd}}$ over 10 evaluation seeds of 50 episodes each (sd across seeds, here and throughout), with CEM behind the SWM solver (Maes et al., 2026a) on identical fixed start–goal pairs. Bold marks the best model within each family.

Family	Obs.	Method	Two-Room	Reacher	Push-T	OGB-Cube
		Random	$25.2_{\pm 4.1}$	$10.8_{\pm 4.0}$	$2.4_{\pm 1.3}$	$46.0_{\pm 8.5}$
SIGReg	Image	LeWM	$85.6_{\pm 5.7}$	$\mathbf{82.0}_{\pm 5.6}$	$\mathbf{86.4}_{\pm 5.2}$	$\mathbf{69.8}_{\pm 9.3}$
	Points	Point-LeWM	$\mathbf{87.0}_{\pm 4.4}$	$80.4_{\pm 4.6}$	$83.6_{\pm 3.4}$	$66.0_{\pm 5.9}$
Act.-reconst.	Image	Delta-JEPA	$99.8_{\pm 0.6}$	$\mathbf{86.0}_{\pm 3.3}$	$\mathbf{94.2}_{\pm 2.7}$	$80.2_{\pm 6.0}$
	Points	Point-Delta-JEPA	$\mathbf{100.0}_{\pm 0.0}$	$77.6_{\pm 4.4}$	$70.8_{\pm 7.7}$	$\mathbf{83.4}_{\pm 3.8}$
Frozen enc.	Image	DINO-WM	$\mathbf{99.8}_{\pm 0.6}$	$\mathbf{77.0}_{\pm 4.8}$	$\mathbf{76.0}_{\pm 9.5}$	$\mathbf{78.6}_{\pm 6.7}$
	Points	Utonia-WM	$94.0_{\pm 1.9}$	$71.2_{\pm 3.7}$	$46.6_{\pm 6.6}$	$71.2_{\pm 7.7}$
	Points	Vox-WM	$83.8_{\pm 5.2}$	$71.8_{\pm 6.7}$	$55.8_{\pm 4.9}$	$65.2_{\pm 9.1}$

of the missing scene content. Some ambiguity remains, since parts of the scene can change over k steps, and a regression model then predicts the mean of the plausible latents, which need not match any encoder output (Appendix B). We therefore compare two independent choices in Table 3. First, whether m_ξ is conditioned on $\boldsymbol{z}_t$. Second, its architecture: a residual MLP over Fourier-featurized target coordinates trained with equation 5, or a shortcut model (Frans et al., 2025) that samples a latent in one step and is trained with its own flow-matching objective on the same pairs.

At test time, $\boldsymbol{z}_g = m_\xi(\boldsymbol{c}_g, \boldsymbol{z}_t)$ or $\boldsymbol{z}_g = m_\xi(\boldsymbol{c}_g)$ replaces the encoded goal observation in the planning objective. The world model and planner are unchanged. Planning details and hyperparameters are listed in Appendix A.4, Architectures and training details are in Appendix A.5, and Appendix B discusses rendering a synthetic goal scan as an alternative.

4 Experiments

All models are trained on the datasets of Section 3.1, where each image episode has a matching point cloud episode, so both modalities share the same underlying scenarios. We evaluate goal-conditioned planning with receding-horizon CEM from the SWM solver layer (Maes et al., 2026a), using 10 seeds of 50 episodes each and the same fixed start–goal pairs for all models.

4.1 Baselines

Table 2 answers the title question. Every point cloud world model plans far above Random despite moving fractions of only 0.3–15%, so neither the geometric shortcut nor the slow-feature shortcut of Section 2 dominate. Significance tests are paired over seeds (Appendix E).

SIGReg. Point-LeWM does not differ significantly from LeWM on any benchmark, and the largest deficit is 3.8 points on OGB-Cube. Sparsity, missing ordering and self-occlusion thus cost little planning performance and little compute, 0.53 s versus 0.48 s per CEM solve (Appendix A.6).

Action reconstruction. Point-Delta-JEPA matches image Delta-JEPA, retrained with the same time budget (Zhang et al., 2026b), on Two-Room and on OGB-Cube, where it attains the best result of Table 2. On Reacher and Push-T it trails significantly, by 8.4 and 23.4 points. The point cloud input alone cannot explain these gaps, since Point-LeWM shows none on the same environments. Compared with Point-LeWM, Point-Delta-JEPA gains 13.0 and 17.4 points on Two-Room and OGB-Cube but loses 12.8 on Push-T. Probing suggests that these differences reflect which task-relevant information each objective preserves (Section 4.2). The results therefore show that the choice of anti-collapse objective is task-dependent.

Frozen encoder. Utonia-WM transfers the frozen-encoder recipe to point clouds but trails DINO-WM on every benchmark, most on Push-T (-29.4, versus -5.8 to -7.4 elsewhere). Vox-WM replaces only its pretrained features with voxel statistics. Utonia features help on Two-Room ($+10.2$)

Table 3: **Planning success rate (%) toward 3D targets.** Mean$_{\pm \text{sd}}$ over 10 seeds (50 episodes each). Target encoders (P-LeWM/PD-JEPA abbreviate Point-LeWM/Point-Delta-JEPA) map the 3D target to a goal latent. $\boldsymbol{z}_t$ marks whether the current latent is an additional input. *Goal cloud* encodes the stored goal cloud (reference). Best per column in bold, separately with and without $\boldsymbol{z}_t$.

		Two-Room		Reacher		Push-T		OGB-Cube	
Encoder	$\boldsymbol{z}_t$	P-LeWM	PD-JEPA	P-LeWM	PD-JEPA	P-LeWM	PD-JEPA	P-LeWM	PD-JEPA
MLP	✗	87.4 ± 4.8	100.0 ± 0.0	**77.4 ± 5.3**	**80.0 ± 5.5**	**85.0 ± 3.6**	**71.8 ± 7.1**	**65.2 ± 6.5**	72.0 ± 5.2
Shortcut	✗	**87.6 ± 4.8**	100.0 ± 0.0	73.6 ± 6.7	72.8 ± 4.5	83.2 ± 4.4	71.4 ± 6.8	61.6 ± 5.7	**78.8 ± 5.3**
MLP	✓	88.4 ± 4.3	100.0 ± 0.0	**77.0 ± 4.2**	**77.8 ± 4.9**	83.8 ± 5.0	70.8 ± 7.4	66.0 ± 7.8	82.4 ± 5.9
Shortcut	✓	**88.6 ± 3.9**	100.0 ± 0.0	73.6 ± 5.2	71.2 ± 4.9	**84.8 ± 2.3**	**73.6 ± 5.8**	**68.8 ± 8.3**	**83.2 ± 5.0**
Goal cloud		87.2 ± 4.1	100.0 ± 0.0	78.8 ± 4.6	76.8 ± 2.3	84.8 ± 4.1	71.2 ± 5.0	65.4 ± 7.5	84.2 ± 4.2

and OGB-Cube ($+6.0$), make little difference on Reacher (-0.6), and hurt on Push-T (-9.2). In both scans, 97–98% of returns lie on the ground plane beneath low, thin entities (Table 1), so voxel occupancy and moments already capture most of the geometry. Pretrained features help only where genuine 3D structure matters, such as the walls of Two-Room and the arm and cube of OGB-Cube.

4.2 Ablation

In this subsection, we examine what the latent spaces of Point-LeWM and Point-Delta-JEPA encode, how well physical state can be decoded from them, and how robust planning is to degraded input point clouds.

Physical latent probing. We regress the privileged simulator state from the 192-dimensional latent used by the CEM cost, training linear (ridge) and MLP probes on episode-disjoint splits (Appendix D.1). Both models encode *positions* well in every environment: linear probes reach $R^2 \geq 0.94$ and MLP probes $R^2 \geq 0.98$ (Tables 7–9). However, they differ in how well they encode the Push-T block *angle* (Table 7), which is linearly decodable from Point-LeWM ($R^2 = 0.89$) but much less so from Point-Delta-JEPA ($R^2 = 0.66$). An MLP recovers the angle from both models ($R^2 = 0.99$ and 0.96), so Point-Delta-JEPA retains most of this information but encodes it non-linearly. Since Push-T success requires matching the block's orientation as well as its position, this difference in angle encoding is consistent with the 13-point gap in success rate in favor of Point-LeWM. On OGB-Cube, Point-Delta-JEPA encodes joint *velocity* substantially better than Point-LeWM (MLP $R^2 = 0.63$ vs. 0.10; Table 9). This aligns with its action-reconstruction objective and its stronger planning performance on this environment.

Point cloud perturbations. We degrade the evaluation cloud with along-ray range noise ($\sigma \in \{0.25, 0.5\}\%$ of the scan's bounding-box diagonal) and random dropout (25 or 50% of returns), applied either to the observation alone or to both observation and goal (Table 10, Appendix D.2). Degrading the goal as well lowers success by a further 6% (relative) on average. We discuss this setting in Section 4.3 and report clean-goal results here. On the sparse scenes, Point-LeWM is more robust. Under 0.25% noise on Two-Room, it retains 80% of its clean success rate versus 43% for Point-Delta-JEPA. Under 50% dropout, it retains 99% on Two-Room and 95% on Reacher, versus 64% and 87% for Point-Delta-JEPA. Attention maps (Appendix D.3) show that both encoders still attend to moving entities that make up only 0.3% of returns, although Point-Delta-JEPA places more attention on the ground and table. Push-T fails under noise for both models (success below 5%): with 98% of returns on the ground plane, the noise overwhelms the thin pusher and block. OGB-Cube is barely affected (98–103% of clean success under noise, 99–102% under dropout), possibly because about 15% of its returns come from moving objects, which makes the arm and cube easier to separate from the table.

4.3 Target-to-latent module results

Table 3 compares planning toward a goal latent predicted by the target-to-latent module (Section 3.2) with planning toward the encoded goal cloud. In every column, at least one encoder conditioned on the current latent $\boldsymbol{z}_t$ matches the goal-cloud reference within one standard deviation, showing

Table 4: **Target-to-latent module under sensor degradation.** Success rate relative to each base model's clean performance (100 = no degradation, clean rates as in Table 2), mean$_{\pm sd}$ over 10 seeds (50 episodes each). *Base* encodes the degraded goal cloud, +MLP$_z$ receives the clean 3D target through the z_t-conditioned MLP. Better of the pair in bold.

	Two-Room				OGB-Cube			
	P-LeWM		PD-JEPA		P-LeWM		PD-JEPA	
Perturbation	Base	+MLP$_z$	Base	+MLP$_z$	Base	+MLP$_z$	Base	+MLP$_z$
Noise 0.25%	$61.6_{\pm 5.7}$	**80.2**$_{\pm 7.0}$	$31.0_{\pm 5.1}$	**41.0**$_{\pm 5.6}$	$97.0_{\pm 12.0}$	**100.6**$_{\pm 10.5}$	$93.5_{\pm 6.8}$	**98.3**$_{\pm 7.6}$
Noise 0.5%	$37.9_{\pm 6.0}$	**53.6**$_{\pm 8.9}$	$35.8_{\pm 6.8}$	**36.4**$_{\pm 4.3}$	$89.7_{\pm 15.9}$	**97.6**$_{\pm 11.2}$	$77.2_{\pm 9.4}$	**90.9**$_{\pm 8.3}$
Dropout 25%	$96.1_{\pm 5.9}$	**99.3**$_{\pm 5.4}$	$60.4_{\pm 6.4}$	**72.0**$_{\pm 6.8}$	$99.1_{\pm 12.3}$	**100.0**$_{\pm 10.0}$	**98.8**$_{\pm 4.1}$	$98.6_{\pm 7.4}$
Dropout 50%	$92.2_{\pm 4.6}$	**98.4**$_{\pm 5.7}$	$50.2_{\pm 5.5}$	**63.0**$_{\pm 4.2}$	**101.5**$_{\pm 8.5}$	$100.0_{\pm 13.0}$	$99.5_{\pm 4.4}$	**101.0**$_{\pm 6.6}$

that a low-dimensional target pose can replace the goal observation without reducing success. As predicted in Section 3.2.2, z_t matters only when the target leaves some moving entities unconstrained. On Two-Room, Reacher, and Push-T, the target fixes all moving entities, so the unconditioned MLP already matches the reference. On OGB-Cube, where the arm configuration remains free, dropping z_t costs up to 10 points (Point-Delta-JEPA with the MLP encoder). Neither encoder is consistently better. The shortcut is best or tied in six of the eight conditioned columns, with a maximum difference of 2.8 points, while the MLP is clearly better for Point-Delta-JEPA on Reacher (77.8 vs. 71.2).

Target-to-latent module under sensor degradation. When the deployed sensor is noisier than during training, the base model must plan with a degraded observation and a degraded goal cloud, whereas the target-to-latent module still receives a clean 3D target. In this setting (Table 4), the module matches or exceeds the base model in 14 of 16 cells, and the two exceptions are within seed noise. On Two-Room, Point-LeWM rises from 62% to 80% of its clean success rate under 0.25% noise and from 38% to 54% under 0.5% noise. Point-Delta-JEPA gains 10–13 points on Two-Room under 0.25% noise and both dropout levels, and 14 points on OGB-Cube under 0.5% noise. The full results (Table 11) show similar gains on Push-T and Reacher under dropout, while range noise on Push-T remains a failure mode for all variants.

Planning toward 3D goal poses also generalizes beyond the trained goal distribution, matching planning toward a goal cloud rendered at the specified state (Section D.2.1).

5 Conclusion

Limitations. Utonia-WM plans slowly: its predictor rolls out ≈ 256 latent tokens where the end-to-end models roll out one, a $\approx 50\times$ gap that mirrors DINO-WM (Maes et al., 2026b) and excludes it from the perturbation ablation. Its fixed grid pooling and random projection also impose token structure the end-to-end models learn, a plausible source of its success rate gap. The target-to-latent module assumes training-time target annotations. All evaluation is confined to four simulated tabletop scenes with an idealized sensor, so real-world transfer remains untested. We partially mitigate this by applying synthetic degradations at test time.

Conclusion. Our evaluations show that world models are not limited to image observations and that they can also operate effectively on LiDAR-style point clouds. This holds true despite two known trivial shortcuts that can cause point-cloud representations to degenerate: the geometric shortcut and the slow-feature shortcut. On OGB-Cube, the most representative of the evaluated 3D environments, action reconstruction (Point-Delta-JEPA) proves to be the most effective anti-collapse approach. Our analysis explains why. Object positions are almost perfectly linearly decodable from the learned representations, attention concentrates on the task-relevant entities, and planning remains robust under heavy dropout not seen during training. The main remaining failure mode is range noise on thin structures. Additionally, for planning, we provide a goal-pose encoder that offers a more natural interface for robotic manipulation than encoding an image of the desired end state. Taken together, these results establish the first geometry-driven, rather than image-based, approach to latent world-model-based planning.

AI USE STATEMENT

In this work, we used generative AI tools (Claude Code) for implementing methods, including the point-cloud sensing pipeline, providing feedback on the experimental methodology, and assisting in the interpretation of results. We have not used generative AI tools for proposing hypotheses, formulating the mathematical arguments (Appendix B), generating data or annotations, or translation; all point clouds and annotations are produced by the simulators. Proofs and qualitative data analysis are not applicable to this work. Additionally, we used generative AI tools for creating and editing software code, including the code for the statistical tests, creating figures, and editing the paper for readability and grammar. We have reviewed the AI-assisted work. All AI-assisted code was reviewed by the authors and validated end-to-end through the full training and evaluation pipeline, methodological suggestions were independently reproduced before being adopted, and all reported numbers, including the statistical tests, come from logged evaluation runs with multiple seeds rather than from AI output. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work involves no human subjects, crowdsourcing, or personal data. All experiments use four publicly released simulated benchmarks, which we re-sense with a simulated range sensor; every point cloud and annotation is produced by the simulators, so no real-world scans of people or private spaces are collected. All existing assets, including datasets, environments, pretrained checkpoints, and software, are used under their published licenses, listed in Appendix A. The work is foundational research on world-model learning and is not tied to a specific deployment; we discuss its broader impacts, including potential misuse of range-based autonomy for surveillance, in Appendix F. The compute used, including preliminary and failed runs, is reported in Appendix A. We are not aware of conflicts of interest related to this work.

REPRODUCIBILITY STATEMENT

We provide anonymized source code as supplementary material, covering the point-cloud sensing and dataset conversion pipeline, all world models and the target-to-latent module, training and evaluation scripts, the configuration files for every reported experiment, and a pinned software environment. Because the datasets are re-sensed from publicly released benchmarks, they can be regenerated with the supplied conversion code; the sensing setup and the choices behind it are described in Section 3.1 and Appendix C. Architectures, optimization constants, planner settings, and compute are listed in Section 3.2 and Appendix A, and the image baselines are re-evaluated from released checkpoints or retrained from their official implementations (Section 3.2). Evaluation uses fixed lists of start–goal pairs shared by all models, with 10 evaluation seeds per setting (Section 4); the probing, perturbation, and never-seen-goal protocols are given in Appendix D, and the statistical tests in Appendix E. [We will release the trained checkpoints and converted datasets upon publication.]

REFERENCES

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

Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models. In *ICLR*, 2024.

Letian Cheng, Qi Zhang, and Yisen Wang. SAGE: Subgoal-conditioned action generation for latent world model planning, 2026.

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.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. In *NeurIPS*, 2023.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations*, volume 2025, pp. 34668–34684, 2025.

David Ha and Jürgen Schmidhuber. World models, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *ICML*, 2019.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *ICLR*, 2024.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

Zhengdong Hong, Shenrui Wu, Haozhe Cui, Boyi Zhao, Ran Ji, Yiyang He, Hangxing Zhang, Zundong Ke, Jun Wang, Guofeng Zhang, and Jiayuan Gu. R3D: Revisiting 3D policy learning, 2026.

Wenlong Huang, Yu-Wei Chao, Arsalan Mousavian, Ming-Yu Liu, Dieter Fox, Kaichun Mo, and Li Fei-Fei. PointWorld: Scaling 3D world models for in-the-wild robotic manipulation, 2026.

Petr Ivashkov, Randall Balestriero, and Bernhard Schölkopf. Sensorimotor world models: Perception for action via inverse dynamics, 2026.

William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *2023 IEEE/CVF international conference on computer vision (ICCV)*, pp. 3992–4003. IEEE, 2023.

Yann LeCun. A path towards autonomous machine intelligence, 2022. URL `https://openreview.net/forum?id=BZ5a1r-kVsf`.

Pei Liu, Songtao Wang, Lang Zhang, Xingyue Peng, Yuandong Lyu, Jiaxin Deng, Songxin Lu, Weiliang Ma, Xueyang Zhang, Yifei Zhan, XianPeng Lang, and Jun Ma. LiSTAR: Ray-centric world models for 4D LiDAR sequences in autonomous driving, 2025.

Lucas Maes, Quentin Le Lidec, Luiz Facury, Nassim Massaudi, Ayush Chaurasia, Francesco Capuano, Richard Gao, Taj Gillin, Dan Haramati, Damien Scieur, Yann LeCun, and Randall Balestriero. stable-worldmodel: A platform for reproducible world modeling research and evaluation, 2026a.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorldModel: Stable end-to-end joint-embedding predictive architecture from pixels, 2026b.

Sergi Masip, Jonathan Swinnen, Yutong Hu, Renaud Detry, and Tinne Tuytelaars. FF-JEPA: Long-horizon planning in world models with latent planners, 2026.

Ishan Misra, Rohit Girdhar, and Armand Joulin. An end-to-end transformer model for 3D object detection. In *ICCV*, 2021.

Ashvin V. Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *NeurIPS*, 2018.

Rhys Newbury, Morris Gu, Lachlan Chumbley, Arsalan Mousavian, Clemens Eppner, Jürgen Leitner, Jeannette Bohg, Antonio Morales, Tamim Asfour, Danica Kragic, Dieter Fox, and Akansel Cosgun. Deep learning approaches to grasp synthesis: A review. *IEEE Transactions on Robotics*, 39(5), 2023.

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

Fabio F. Oberweger, Michael Schwingshackl, and Vanessa Staderini. PI3DETR: Parametric instance detection of 3D point cloud edges with a geometry-aware 3DETR. In *3DV*, 2026. doi: 10.1109/3DV69130.2026.00052.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations (ICLR)*, 2025.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Skand Peri, Hung Nguyen, Chanho Kim, Fuxin Li, and Stefan Lee. 3D point world models: Point completion enables more accurate dynamics learning, 2026.

Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, 2017a.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017b.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Michael Schwingshackl, Fabio Francisco Oberweger, and Markus Murschitz. Few-shot structure-informed machinery part segmentation with foundation models and graph neural networks. In *WACV*, 2025. doi: 10.1109/WACV61041.2025.00200.

Michael Schwingshackl, Fabio F. Oberweger, Mario Niedermeyer, Johannes Huemer, and Markus Murschitz. PIRATR: Parametric object inference for robotic applications with transformers in 3D point clouds. In *ICRA*, 2026.

Vlad Sobal, Jyothir S V, Siddhartha Jalagam, Nicolas Carion, Kyunghyun Cho, and Yann LeCun. Joint embedding predictive architectures focus on slow features. In *NeurIPS Workshop on Self-Supervised Learning: Theory and Practice*, 2022.

Vlad Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. In *NeurIPS*, 2025.

Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. Contact-GraspNet: Efficient 6-DoF grasp generation in cluttered scenes. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

Stefan Thalhammer, Dominik Bauer, Peter Hönig, Jean-Baptiste Weibel, José García-Rodríguez, and Markus Vincze. Challenges for monocular 6-D object pose estimation in robotics. *IEEE Transactions on Robotics*, 40, 2024.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5026–5033, 2012.

Yuhai Wang, Jiawei Xia, Rongxuan Zhou, Xiao Hu, Yongliang Shi, Jing Du, and Yang Ye. PRISM: PRior-guided imagination sampling in world models, 2026.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer V3: Simpler, faster, stronger. In *CVPR*, 2023.

Xiaoyang Wu, Daniel DeTone, Duncan Frost, Tianwei Shen, Chris Xie, Nan Yang, Jakob Engel, Richard Newcombe, Hengshuang Zhao, and Julian Straub. Sonata: Self-supervised learning of reliable point representations. In *CVPR*, 2025.

Zetong Yang, Li Chen, Yanan Sun, and Hongyang Li. Visual point cloud forecasting enables scalable autonomous driving. In *CVPR*, 2024. doi: 10.1109/CVPR52733.2024.01390.

Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-BERT: Pre-training 3D point cloud transformers with masked point modeling. In *CVPR*, 2022.

Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, and Johnny Lee. Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning (CoRL)*, 2020.

Lunjun Zhang, Yuwen Xiong, Ze Yang, Sergio Casas, Rui Hu, and Raquel Urtasun. Copilot4D: Learning unsupervised world models for autonomous driving via discrete diffusion. In *ICLR*, 2024.

Yujia Zhang, Xiaoyang Wu, Yunhan Yang, Xianzhe Fan, Han Li, Yuechen Zhang, Zehao Huang, Naiyan Wang, and Hengshuang Zhao. Utonia: Toward one encoder for all point clouds. In *ICML*, 2026a.

Zhenghao Zhang, Yuanxiang Wang, Zhenyu Guan, Yujia Yang, Bingkang Shi, Tianyu Zong, Hongzhu Yi, Guoqing Chao, Xingchen Chen, Tiankun Yang, Chenxi Bao, Tao Yu, Jingjing Zhou, and Jungang Xu. Delta-JEPA: Learning action-sensitive world models via latent difference decoding, 2026b.

Wenzhao Zheng, Weiliang Chen, Yuanhui Huang, Borui Zhang, Yueqi Duan, and Jiwen Lu. OccWorld: Learning a 3D occupancy world model for autonomous driving. In *ECCV*, 2024. doi: 10.1007/978-3-031-72624-8_4.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *ICML*, 2025.

Junsheng Zhou, Jinsheng Wang, Baorui Ma, Yu-Shen Liu, Tiejun Huang, and Xinlong Wang. Uni3D: Exploring unified 3D representation at scale. In *ICLR*, 2024.

Haoran Zhu and Anna Choromanska. Self-supervised JEPA-based world models for LiDAR occupancy completion and forecasting, 2026.

Haoran Zhu, Zhenyuan Dong, Kristi Topollai, Beiyao Sha, and Anna Choromanska. Self-supervised representation learning with joint embedding predictive architecture for automotive LiDAR object detection. In *AAAI*, 2026. doi: 10.1609/aaai.v40i16.38402.

APPENDIX

This appendix collects the material that supports the main text. Appendix A lists the architectures, optimization constants, and compute behind every model. Appendix B formalizes why the target-to-latent module conditions on the current latent, and Appendix C details how the four benchmark datasets are re-sensed as point clouds. Appendix D probes the latents for physical state, perturbs the point clouds at test time, and visualizes where the encoders attend. Appendix E backs the main comparisons with paired significance tests, and Appendix F discusses broader impacts.

A IMPLEMENTATION DETAILS

This section records the architecture and optimization constants behind Section 3.2. The released configurations remain authoritative. All world models are trained on two A100 GPUs.

A.1 POINT-CLOUD ENCODER (POINT-LEWM, POINT-DELTA-JEPA)

The shared encoder (Figure 1) consumes geometry only: the per-point input features are the sensor-frame coordinates, with no intensity or color channel. Each cloud is reduced to 256 farthest-point-sampled centres, and a ball query of per-environment metric radius (0.15 m Two-Room, 0.18 m Push-T, 0.04 m Reacher) draws up to 32 points per centre uniformly at random (Qi et al., 2017b). Coordinates are first normalized by the affine transformation $\boldsymbol{p} \mapsto (\boldsymbol{p} - \boldsymbol{\mu})/s$ with a per-environment centre $\boldsymbol{\mu} \in \mathbb{R}^3$ and isotropic scale s measured once on the dataset, which brings the workspace to approximately $[-1, 1]^3$ while preserving absolute object positions. The centre-relative offsets of each group then pass through a two-stage mini-PointNet (Qi et al., 2017a; Yu et al., 2022) with hidden widths $(128, 256)$, yielding one 192-dimensional token to which a log-spaced sinusoidal embedding of the group centre (Misra et al., 2021) is added. The tokens, prepended by a learned CLS token, feed a LayerNorm-only (Hong et al., 2026) ViT-Tiny trunk (12 layers, 3 heads, width 192, $\approx$5.5M parameters) (Dosovitskiy et al., 2021), whose CLS output is the latent $\boldsymbol{z}_t$. OGB-Cube uses 512 tokens with radius 0.03 m instead. The denser sampling helped only there, since the other scenes are dominated by the ground plane.

The predictor is LeWorldModel's causal action-conditioned transformer (Maes et al., 2026b): 6 blocks, 16 heads of dimension 64, MLP width 2048 at the 192-dimensional latent width, with a context of $\ell = 3$ past latents. Latents pass a BatchNorm MLP projector (hidden 2048), as in the image model. The world models step at frameskip 5, the five intervening environment actions concatenated and embedded by a shared linear action encoder. Both end-to-end models train for 10 epochs with AdamW (learning rate 5×10^{-5}, weight decay 10^{-3}) at effective batch size 128, bf16 precision, and gradient clipping 1.0. The SIGReg weight is $\lambda = 0.09$ (17 knots, 1024 projections) (Balestriero & LeCun, 2025; Maes et al., 2026b). The action-reconstruction weight is $\beta = 10$ at horizon N=5 (Zhang et al., 2026b), for which Point-Delta-JEPA trains on 6-frame windows while planning with the same 3-latent context. On OGB-Cube both models follow a 50-epoch cosine schedule, with the reported checkpoints still taken after 10 epochs.

A.2 UTONIA-WM

Completing Section 3.2: point clouds are expressed in a canonical frame derived from the fitted ground plane, rescaled by Utonia's own tabletop factor of 4.0, and voxel-downsampled at a grid size of 0.01, keeping one point per occupied voxel, before entering the frozen $\approx$137M-parameter PTv3 backbone (Wu et al., 2023; Zhang et al., 2026a). Per grid cell, the backbone's coarsest-stage 576-dimensional super-point features are mean- and max-pooled (1152 dimensions) and compressed to the 192-dimensional token width by the fixed random orthogonal projection of Section 3.2, a Gaussian matrix drawn once from a fixed seed and orthonormalized via QR. The grids are $8 \times 8 \times 4 = 256$ tokens (OGB-Cube, Reacher), $12 \times 11 \times 2 = 264$ (Push-T), and $10 \times 9 \times 3 = 270$ (Two-Room), spanning a per-environment workspace box. The predictor is DINO-WM's frame-block-causal token transformer (Zhou et al., 2025) at the same depth and widths as above, with the action embedding tiled over a frame's tokens and concatenated onto each. Only the predictor and action encoder train, against precomputed frozen features, for 100 epochs with AdamW (learning rate 10^{-3}, weight decay 10^{-3}) at effective batch size 512.

A.3 VOX-WM

Vox-WM is a parameter-free geometric control for Utonia-WM that ablates the frozen foundation encoder at fixed tokenization: same canonical frame, same token grid, same predictor and prediction-only loss, but each cell's token holds plain statistics of the cell's points instead of pooled frozen features. After the voxel downsampling above, a cell's points are expressed as offsets from the cell centre, and the 25-dimensional token stacks an occupancy bit (1), the point count as $\log(1+n)/\log(1+256)$ (1), the per-axis mean (3), standard deviation (3), minimum (3), and maximum (3) of the offsets, their three off-diagonal covariances (3), and the occupancy bits of a $2 \times 2 \times 2$ subdivision of the cell (8). Empty cells are zero tokens, which the occupancy bit disambiguates. The 25 statistics are lifted to the 192-dimensional predictor width by a fixed random orthogonal projection, mirroring the one above, so both models present an identical training-free interface to the predictor and differ only in what fills each token. The encoding is deterministic and costs microseconds per cloud, so it runs live without a feature cache. Training follows the Utonia-WM protocol.

A.4 PLANNING

All methods plan with the receding-horizon model predictive control (MPC) loop of the SWM framework (Maes et al., 2026a), with one configuration in all four environments. At each planning call, the policy encodes the goal observation once and conditions the predictor on the same three-frame observation history used in training. The cross-entropy method (CEM) then searches for an action sequence that minimizes the squared distance between the predicted latent at the end of the horizon and the goal latent (Section 3.2). Only the final predicted step enters the cost. CEM maintains a diagonal Gaussian over standardized actions, initialized with zero mean and unit variance. Each iteration samples 300 candidate sequences, always including the current mean, scores them in one batched rollout of the predictor, and refits the mean and standard deviation to the 30 lowest-cost candidates. After 30 iterations the mean of the distribution is returned as the plan. The planning horizon is 5 steps, and each planned action is repeated for 5 environment steps (frameskip 5), so one plan covers 25 environment steps. The agent executes the full plan and then replans from the new observation without warm-starting from the previous plan. The goal is the state 25 steps ahead of the start along a trajectory of the training dataset, and success is reaching it within a budget of 50 environment steps, so at most two plans are computed per episode. Every model is evaluated on the same 50 start states under each of the same ten seeds, so the image and point-cloud comparisons of Table 2 are paired under one protocol.

A.5 TARGET-TO-LATENT MODULE

Both instantiations of m_ξ (Section 3.2) share one conditioning interface. The target coordinates, namely the cube position (3-d), the Two-Room agent position (3-d), the Reacher joint positions (6-d), or the Push-T agent and block positions together with the block's heading vector (9-d), are expressed in the sensor frame, normalized by the same workspace map $\boldsymbol{p} \mapsto (\boldsymbol{p} - \boldsymbol{\mu})/s$ as the encoder inputs, and lifted by log-spaced sinusoidal Fourier features (32 bands) (Misra et al., 2021), which make sub-centimetre offsets linearly separable. The cube target carries no orientation channel, for the symmetry reason of Appendix D.1. The *MLP* head is a residual MLP (width 1024, hidden 2048, 4 blocks) regressing the normalized goal latent. The heading channels and, when used, $\boldsymbol{z}_t$ enter through a linear embedding added to the Fourier code.

The *shortcut* head (Frans et al., 2025) is an adaptive-LayerNorm-modulated residual MLP of the same widths over the normalized latent space, trained with flow matching plus the self-consistency loss against an EMA copy (decay 0.999). Here, the heading channels and $\boldsymbol{z}_t$ join the conditioning vector that modulates every block. Training pairs are frames of one trajectory separated by ± 5 to ± 25 environment steps, with $\boldsymbol{z}_t$ the latent of the earlier frame and the target read from the later. At evaluation $\boldsymbol{z}_t$ is re-encoded at every replan.

All heads train for 60 epochs with AdamW (learning rate 3×10^{-4}, weight decay 10^{-4}) at batch size 4096 on precomputed frozen latents, with 5 mm Gaussian jitter on the position channels, without which the Fourier code is sharp enough to memorize individual frames. During evaluation, the shortcut model generates a latent sample in a single inference step. Its inference time is comparable to that of the MLP baseline, which directly predicts the latent in one forward pass.

A.6 COMPUTE

Every world model trains on two A100 GPUs of an internal cluster. The time budget for end-to-end training runs was set to 72 hours. One evaluation sweep of 10 seeds $\times$ 50 episodes costs a few GPU-hours per model and environment, considerably more for Utonia-WM (Section 5). Completing the planning-cost numbers of Section 4: one CEM solve (300 candidates, 30 iterations, horizon 25) takes 0.53 s for the point-cloud model and 0.48 s for the image model on a single A100, and with replanning every 25 steps a 50-step episode requires about 1.1 s and 1.0 s of total planning time, an overhead shared by Point-Delta-JEPA whose encoder is identical. Including preliminary and failed runs, the full project consumed on the order of several thousand A100-hours.

A.7 FIDELITY OF THE IMAGE BASELINES

Table 5 places the image baselines of Table 2 next to the success rates their authors report. Retrained Delta-JEPA matches or exceeds its published rates on all four benchmarks, by $+4.7$ points on Reacher and $+5.1$ points on Push-T. Point-Delta-JEPA in Table 2 is therefore measured against a baseline at full strength. Retrained DINO-WM is within two points of its published rate on Reacher and Push-T and 7.4 points below on OGB-Cube. The released LeWM checkpoint is within four points of its published rate on Two-Room, Reacher and OGB-Cube and 9.6 points below on Push-T. An independent retraining of LeWM by Zhang et al. (2026b) lands at 84.5 on Push-T and 64.1 on OGB-Cube, in line with our 86.4 and 69.8 and below the published 96 and 74.

These offsets are consistent with the evaluation protocols. Maes et al. (2026b) report a single evaluation seed of 50 trajectories, Zhang et al. (2026b) report three seeds of 500 trajectories each sampled by them, and we report ten seeds of 50 episodes on fixed start–goal pairs shared by every model. On OGB-Cube our seed standard deviations range from 5.9 to 9.3 points, so a single seed of 50 episodes can sit a full offset above or below the ten-seed mean, and both independent evaluations of LeWM on that environment land below the published rate. Two further differences apply. Maes et al. (2026b) run 10 CEM iterations outside Push-T, whereas we run 30 everywhere. On Two-Room they sample the goal 100 steps ahead with a 150-step budget, whereas we keep the 25-step goal and 50-step budget of the other environments, which is also why our Two-Room random baseline is 25% rather than 0%. The comparisons in Table 2 are paired within one protocol and are unaffected by these offsets.

Table 5: **Image baselines versus their published success rates (%).** Ours are the Table 2 values under our protocol. Published DINO-WM excludes proprioception, as in our runs. Two-Room protocols differ (see text).

Model	Source	Two-Room	Reacher	Push-T	OGB-Cube
Delta-JEPA	published (Zhang et al., 2026b, Tab. 1)	100.0	81.3	89.1	79.3
	ours (retrained)	99.8	86.0	94.2	80.2
DINO-WM	published (Maes et al., 2026b, Fig. 6)	100	79	74	86
	ours (retrained)	99.8	77.0	76.0	78.6
LeWM	published (Maes et al., 2026b, Fig. 6)	87	86	96	74
	retrained by Zhang et al. (2026b, Tab. 1)	74.9	79.9	84.5	64.1
	ours (released checkpoint)	85.6	82.0	86.4	69.8

A.8 LICENSES

All existing assets are used under their published licenses: the stable-worldmodels suite (0.1.1) and its released LeWM checkpoints (Maes et al., 2026a, MIT), OGBench 1.2.1 (Park et al., 2025, MIT), the Push-T environment (Chi et al., 2023, MIT), MuJoCo 3.10.0 (Todorov et al., 2012, Apache 2.0), the pretrained Utonia backbone (Zhang et al., 2026a, CC-BY-NC-4.0), and PyTorch 2.7.0 (Paszke et al., 2019, BSD-3-Clause).

B THE UNCONDITIONED TARGET-TO-LATENT OBJECTIVE

This section makes precise the claim of Section 3.2 that, without the current latent, the regression target of equation 5 need not be the latent of any reachable observation. Write $\mathcal{Z}_f = \{f_\theta(o) : o \in \operatorname{supp} \mathcal{D}\}$ for the set of latents the frozen encoder assigns to observations in the support of the data distribution $\mathcal{D}$, that is, to observations that actually occur in the data, and let $(\boldsymbol{z}_t, \boldsymbol{c}_{t+k}, \boldsymbol{z}_{t+k})$ be distributed as in equation 5. The unconditioned variant minimizes

$$\mathcal{L}^{\emptyset}_{\text{goal}}(\xi) = \mathbb{E}\big[\|m_\xi(\boldsymbol{c}_{t+k}) - \boldsymbol{z}_{t+k}\|_2^2\big], \tag{6}$$

and over unconstrained functions $m : \mathcal{C} \to \mathcal{Z}$ this mean-squared error is minimized pointwise by the conditional mean

$$m^*(\boldsymbol{c}) = \mathbb{E}\big[\boldsymbol{z}_{t+k} \,\big|\, \boldsymbol{c}_{t+k} = \boldsymbol{c}\big], \tag{7}$$

the barycenter of the latents of *all* data scenes whose annotation equals $\boldsymbol{c}$, covering every configuration of the scene content the target leaves free, on OGB-Cube every arm pose. $\mathcal{Z}_f$ is the image of the low-dimensional scene configuration manifold under the continuous encoder f_θ, hence itself a low-dimensional subset of $\mathbb{R}^{192}$, and no term of the training objective encourages it to be convex. SIGReg in particular shapes one-dimensional projections of the latent distribution rather than the geometry of its support. A barycenter of distinct points of a non-convex set need not lie in the set, and $m^*(\boldsymbol{c})$ can therefore be a latent that no observation in the data distribution produces. A planner minimizing $\|\hat{\boldsymbol{z}}_{t+H} - m^*(\boldsymbol{c})\|_2^2$ is then steered toward a goal latent it may be unable to attain.

Conditioning changes the average, not the argument: with $\boldsymbol{z}_t$ as input the minimizer becomes

$$m^*(\boldsymbol{c}, \boldsymbol{z}_t) = \mathbb{E}\big[\boldsymbol{z}_{t+k} \,\big|\, \boldsymbol{c}_{t+k} = \boldsymbol{c},\ \boldsymbol{z}_t\big], \tag{8}$$

an average over futures of the current scene only. Scene content that $\boldsymbol{z}_t$ determines and that the dynamics leave unchanged over the offset window contributes no variance to this average, so the residual spread stems only from the moving entities the target leaves unconstrained. Whether that residue matters is an empirical question, and Table 3 answers it per environment: on Push-T and Two-Room the target constrains all moving entities and the unconditioned variant already matches the goal-cloud reference, while on OGB-Cube the arm remains free at the goal even given $\boldsymbol{z}_t$, and dropping $\boldsymbol{z}_t$ costs up to 10 points. The two heads of Section 3.2 also treat the average differently: the MLP fits the conditional mean of equation 7 directly, while the shortcut model samples from the conditional distribution and so returns one consistent latent rather than a barycenter, although Table 3 separates the two heads only within seed noise.

Why not synthesize the goal scan? Since we control the mirror geometry, one could place the task-relevant entity at the specified target, cast a synthetic scan of that state, and encode it, with no learned module. The construction founders on the same ambiguity as the unconditioned regression: a scene must be fully configured before a scan can be cast, the target constrains only the task-relevant coordinates, so the synthesizer has to invent the remaining content, on OGB-Cube an arm pose, and an invented configuration need not be reachable or consistent with the present scene. The conditioned module resolves this from $\boldsymbol{z}_t$ without reconstructing geometry, and in deployment the mismatch grows, since no perfect mirror of the scene exists to render from. The module itself is also modality-agnostic: given the same target annotations it could be trained on image latents. What geometry specifically contributes is that the target already lives in the metric space of the observation, so no cross-modal correspondence must be learned.

C POINT-CLOUD SENSING AND DATASET REPLICATION: DETAILS

Exact sensor parameters, mirror geometry and scene constants are given by the released configurations. This section records only the choices behind them.

Ray budget. The tokenizer reduces every cloud to 256 farthest-point-sampling centres, so the ray grid fixes the sampling density of the observation rather than the sequence length seen by the model. We use 100×100 as the coarsest grid at which the smallest task-relevant entity, the Push-T pusher or the Two-Room agent, still returns on the order of a hundred points.

Table 6: **The four re-sensed datasets.** *Ground share* and *moving returns* are the fractions of returns on the ground plane and on non-static entities. Both are constant within an environment.

	OGB-Cube	Two-Room	Push-T	Reacher
Sensor pose	scene camera	derived oblique	derived oblique	derived oblique
Ground share	∼79%	∼80%	∼98%	∼97%
Moving returns	∼15.0%	∼0.3%	∼0.5%	∼2.0%
Episodes	10,000	10,000	18,685	10,000
Frames	2,010,000	920,809	2,336,736	2,010,000
Episode length	201	31–101	49–246	201

Viewpoint. Oblique viewpoints are derived rather than hand-placed: given bounding-sphere radius R the camera sits at $d = 2.5R$ along the 45° azimuth and elevation bearing, with vertical field of view $2\arcsin(R/d)$, so no pose is tuned per environment. The OGBench cube is the exception: its scene already contains the oblique camera its released images were rendered from, and the sensor is mounted on it, so that its scan and the image baseline's observation share a viewpoint.

Mirror scenes. Entity heights in the two mirrored environments are ours to choose rather than inherited from the benchmark, and were fixed once per environment: large enough that an entity is separable from the floor in an oblique scan, small enough that self-occlusion does not dominate. Reacher3D is native, but the geometry of the released environment did not match the dataset it shipped with. We recovered the two link lengths by least-squares fitting to the recorded joint angles and fingertip positions, after which forward kinematics reproduces the recorded fingertip.

Goal handling. The goal marker is excluded from every scan, by an alpha threshold in the cube and by omission from the mirror geometry elsewhere, since a goal specifies a task rather than a surface a range sensor would return, and retaining it would permit a policy to read the goal off the observation rather than infer it. The goal reaches the planner as the stored cloud of the goal frame, indexed from the same converted table by the same goal offset the image pipeline uses, so no separate goal scan is cast.

D Ablations

D.1 Physical latent probing

Protocol. For each environment the frozen checkpoints of the two end-to-end world models encode every tenth frame of its training table. The probed representation is $\boldsymbol{z} = f_\theta(o)$ after the projector, i.e. exactly the 192-dimensional latent the CEM cost operates on. Two probes per target regress privileged simulator state from $\boldsymbol{z}$: a *linear* probe, closed-form ridge regression on standardized features with the regularization weight selected on a validation split over a logarithmic grid, and an *MLP* probe ($192 \to 512 \to 512 \to d$, GELU, AdamW with cosine schedule, early-stopped on the same validation split). Splits are drawn at *episode* granularity (72/8/20 train/val/test), so every test frame comes from a trajectory no probe ever saw. We report test-set MSE in native units and pooled R^2. All targets live in the sensor frame of the stored clouds, and planar angles enter as unit heading vectors, $(\cos\gamma, \sin\gamma, 0)$ for a planar angle γ rotated into the sensor frame, so that no probe has to fit a wrap-around discontinuity.

Results. Table 7 reports Push-T, analyzed in the main text. Tables 8 and 9 complement it on Reacher and OGB-Cube. Reacher is saturated: the arm is fully visible from the oblique viewpoint, and every property, positions and link headings alike, is linearly decodable at $R^2 \approx 1$ from both models, which is why it offers no signal for separating the objectives. On OGB-Cube the two yaw rows require care: a cube is 24-fold rotationally symmetric, so its orientation is not identifiable from geometry at all, and the raw yaw label is not folded into a fundamental domain. The near-zero R^2 therefore reflects an unidentifiable *label*, not a representation failure. The end-effector yaw is similarly flat for both models. The wrist subtends only a handful of returns at the 100×100 ray budget, and neither objective rewards retaining it. Joint velocity, finally, is not observable from a single scan and can be decoded only through its correlation with configuration along the data distribution. That

Table 7: **Physical latent probing results on Push-T.** All targets are 3-dimensional: locations are coordinates in the sensor frame, and the block angle is encoded as a heading vector in the sensor frame. Lower MSE and higher R^2 indicate better representation quality.

		Linear		MLP	
Property	**Method**	**MSE** ↓	R^2 ↑	**MSE** ↓	R^2 ↑
Agent Location	Point-LeWM	0.008	0.955	**0.001**	**0.997**
	Point-Delta-JEPA	**0.007**	**0.957**	0.002	0.989
Block Location	Point-LeWM	**0.004**	**0.958**	**0.000**	**0.998**
	Point-Delta-JEPA	0.005	0.946	0.001	0.992
Block Angle	Point-LeWM	**0.027**	**0.891**	**0.002**	**0.992**
	Point-Delta-JEPA	0.084	0.657	0.009	0.962

Table 8: **Physical latent probing results on Reacher.** Positions are 3D coordinates in the sensor frame, and joint angles are encoded as per-link unit heading vectors in the sensor frame. Lower MSE and higher R^2 indicate better representation quality.

		Linear		MLP	
Property	**Method**	**MSE** ↓	R^2 ↑	**MSE** ↓	R^2 ↑
Finger Position	Point-LeWM	**0.000**	**1.000**	**0.000**	**1.000**
	Point-Delta-JEPA	**0.000**	0.999	**0.000**	**1.000**
Elbow Position	Point-LeWM	**0.000**	**1.000**	**0.000**	**1.000**
	Point-Delta-JEPA	**0.000**	0.999	**0.000**	**1.000**
Joint Angles	Point-LeWM	**0.000**	**1.000**	**0.000**	**1.000**
	Point-Delta-JEPA	0.001	0.998	**0.000**	**1.000**

Point-Delta-JEPA alone recovers a substantial share of it (R^2 0.63 vs. 0.10 with the MLP probe) is consistent with its objective, which forces the latent *displacement* to be decodable into the executed actions.

D.2 Point cloud perturbations

Protocol. Perturbations are applied at evaluation time only, downstream of a faithful scan. No model ever saw a degraded cloud in training. Test-time-only degradation is the probe matched to our question: it asks whether latents trained on clean geometry anchor to structure a worse sensor still measures, the situation of a model deployed on hardware noisier than its training data. Degrading the training data instead would measure whether the objectives can absorb noise, a property of training rather than of the learned representation. *Noise* is along-ray rather than isotropic, since a range sensor errs in the range it reports and each return therefore moves along its own line of sight. Its standard deviation σ is set as a fraction of the per-frame bounding-box diagonal of the valid returns. The relative scale is what makes the ablation comparable across environments: an absolute σ of 2 cm is a light haze on the 4.9 m Push-T arena and a catastrophe on the 1.4 m Reacher workspace. *Dropout* marks a random fraction of returns as misses, which is what a lost return physically is, so the dropped points leave through the same invalid-return filter the pipeline applies to genuine misses. Because the goal cloud is read from the dataset rather than scanned live, it stays clean under the first condition (a degraded sensor pursuing a stored target, the deployment case) and is degraded identically under the second, which tests the representation itself but no longer distinguishes "cannot see" from "aiming at the wrong thing". Success rates in Table 10 are reported relative to each model's own clean performance, so values may exceed 100 within seed noise.

Beyond the trends discussed in the main text, the table shows that degrading the goal in addition to the observation costs a further, roughly uniform margin (e.g. Two-Room noise 0.25%: 80.2 vs. 61.6 relative for Point-LeWM), and that dropout is benign wherever the surviving returns still sample the

Table 9: **Physical latent probing results on OGB-Cube.** Positions are 3D coordinates in the sensor frame, stacked across the five arm joints for *Joint Positions*. Yaw is encoded as a heading vector in the sensor frame, and joint velocity is the raw six-dimensional vector in rad/s. Lower MSE and higher R^2 indicate better representation quality. The near-zero yaw rows reflect label unidentifiability (a cube is rotationally symmetric, and the wrist returns few points), not representation failure, as discussed in the text.

		Linear		MLP	
Property	**Method**	**MSE** $\downarrow$	R^2 $\uparrow$	**MSE** $\downarrow$	R^2 $\uparrow$
Joint Positions	Point-LeWM	**0.000**	0.992	**0.000**	0.999
	Point-Delta-JEPA	**0.000**	**0.998**	**0.000**	**1.000**
Joint Velocity	Point-LeWM	0.142	0.021	0.130	0.102
	Point-Delta-JEPA	**0.071**	**0.507**	**0.054**	**0.626**
End-Effector Position	Point-LeWM	**0.000**	0.992	**0.000**	0.999
	Point-Delta-JEPA	**0.000**	**0.999**	**0.000**	**1.000**
End-Effector Yaw	Point-LeWM	0.330	-0.001	0.332	-0.006
	Point-Delta-JEPA	**0.321**	**0.029**	**0.321**	**0.026**
Block Position	Point-LeWM	0.001	0.956	**0.000**	0.996
	Point-Delta-JEPA	**0.000**	**0.996**	**0.000**	**0.999**
Block Yaw	Point-LeWM	**0.333**	$\mathbf{-0.001}$	**0.335**	$\mathbf{-0.006}$
	Point-Delta-JEPA	**0.333**	$\mathbf{-0.001}$	**0.335**	$\mathbf{-0.006}$

Table 10: **Point cloud degradation.** Planning success rate over 10 seeds with 50 episodes each, reported *relative to each model's own clean performance* (100 = no degradation). The top row gives the absolute clean success rates used as denominators. The two point-cloud world models are shown side by side under each condition, better of the pair in bold. Due to Utonia-WM's long computation times, it is not part of this ablation table. Noise σ is a fraction of the frame's scan extent, and dropout is the fraction of returns lost.

	Two-Room		Reacher		Push-T		OGB-Cube	
Perturbation	P-LeWM	PD-JEPA	P-LeWM	PD-JEPA	P-LeWM	PD-JEPA	P-LeWM	PD-JEPA
None	$87.0_{\pm 4.4}$	$100.0_{\pm 0.0}$	$80.4_{\pm 4.6}$	$77.6_{\pm 4.4}$	$83.6_{\pm 3.4}$	$70.8_{\pm 7.7}$	$66.0_{\pm 5.9}$	$83.4_{\pm 3.8}$
Observation degraded, goal clean								
Noise 0.25%	$\mathbf{80.2}_{\pm 9.4}$	$42.8_{\pm 6.1}$	$\mathbf{96.3}_{\pm 8.2}$	$89.4_{\pm 6.9}$	$1.9_{\pm 1.9}$	$\mathbf{3.7}_{\pm 2.7}$	$\mathbf{103.0}_{\pm 10.8}$	$99.8_{\pm 6.2}$
Noise 0.5%	$\mathbf{52.2}_{\pm 11.5}$	$36.0_{\pm 6.7}$	$\mathbf{90.8}_{\pm 5.9}$	$62.2_{\pm 6.1}$	$1.2_{\pm 1.7}$	$\mathbf{4.0}_{\pm 2.0}$	$97.6_{\pm 11.4}$	$\mathbf{98.3}_{\pm 6.8}$
Dropout 25%	$\mathbf{99.8}_{\pm 5.4}$	$72.2_{\pm 6.4}$	$\mathbf{97.5}_{\pm 6.1}$	$90.7_{\pm 6.5}$	$90.7_{\pm 6.8}$	$\mathbf{94.6}_{\pm 8.8}$	$\mathbf{101.5}_{\pm 12.7}$	$99.3_{\pm 4.8}$
Dropout 50%	$\mathbf{98.9}_{\pm 5.1}$	$64.0_{\pm 6.7}$	$\mathbf{94.8}_{\pm 6.1}$	$87.3_{\pm 5.6}$	$49.0_{\pm 8.5}$	$\mathbf{52.8}_{\pm 10.5}$	$99.4_{\pm 12.0}$	$\mathbf{100.7}_{\pm 4.9}$
Observation and *goal degraded*								
Noise 0.25%	$\mathbf{61.6}_{\pm 5.7}$	$31.0_{\pm 5.1}$	$\mathbf{91.8}_{\pm 7.8}$	$84.7_{\pm 6.0}$	$3.6_{\pm 2.9}$	$\mathbf{4.0}_{\pm 3.2}$	$\mathbf{97.0}_{\pm 12.0}$	$93.5_{\pm 6.8}$
Noise 0.5%	$\mathbf{37.9}_{\pm 6.0}$	$35.8_{\pm 6.8}$	$\mathbf{87.6}_{\pm 7.5}$	$58.0_{\pm 6.7}$	$\mathbf{4.5}_{\pm 2.4}$	$2.5_{\pm 2.1}$	$\mathbf{89.7}_{\pm 15.9}$	$77.2_{\pm 9.4}$
Dropout 25%	$\mathbf{96.1}_{\pm 5.9}$	$60.4_{\pm 6.4}$	$\mathbf{94.8}_{\pm 5.0}$	$89.1_{\pm 8.7}$	$\mathbf{88.6}_{\pm 6.3}$	$87.0_{\pm 8.2}$	$\mathbf{99.1}_{\pm 12.3}$	$98.8_{\pm 4.1}$
Dropout 50%	$\mathbf{92.2}_{\pm 4.6}$	$50.2_{\pm 5.5}$	$\mathbf{92.3}_{\pm 5.8}$	$75.9_{\pm 3.2}$	$27.4_{\pm 5.5}$	$\mathbf{35.3}_{\pm 8.6}$	$\mathbf{101.5}_{\pm 8.5}$	$99.5_{\pm 4.4}$

entities. Only Push-T, whose pusher is near the ray-budget floor of Appendix C, breaks under 50% dropout.

Target-to-latent module under degradation. Table 11 extends the joint-degradation rows of Table 10 to planners that receive their goal through the target-to-latent module of Section 3.2, with the unconditioned MLP (+MLP) and the z_t-conditioned MLP (+MLP$_z$) as target encoders. The base planner encodes the degraded goal cloud. The module instead receives the clean 3D target and, in the conditioned case, the latent of the degraded current observation, which is the deployment case of a noisy sensor pursuing a target given in coordinates. All numbers remain relative to the base model's clean success rate, so the clean cost of the module itself (Table 3) is included in every cell.

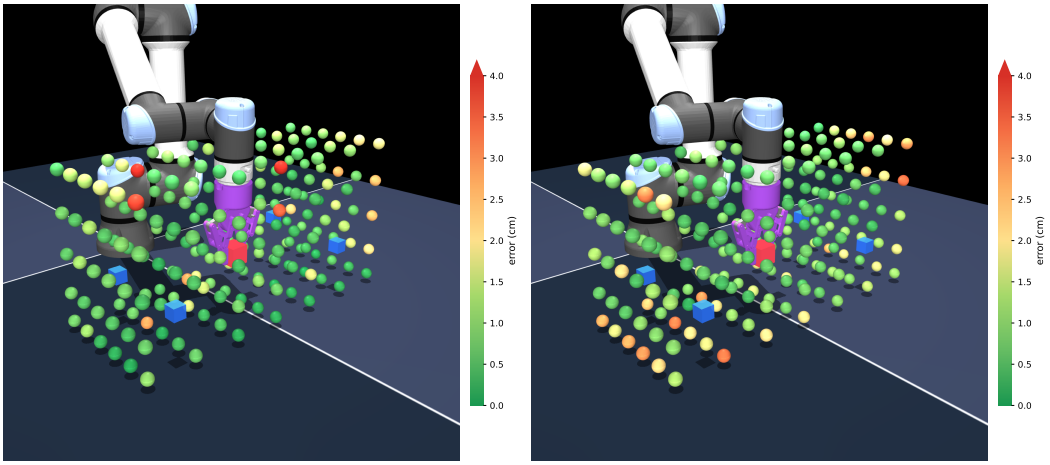
Table 11: **Effect of the target-to-latent module under joint observation-and-goal degradation.** Planning success rate over 10 seeds with 50 episodes each, reported *relative to each base model's clean performance* (100 = no degradation; clean rates as in Table 10). *Base* repeats the observation-and-goal-degraded rows of Table 10; *+MLP* and *+MLP$_z$* add the unconditioned and the z_t-conditioned MLP target encoder of Section 3.2 on top of the same backbone, so the goal enters as a clean 3D target instead of a degraded goal cloud. Best of the three variants per model in bold. Noise σ is a fraction of the frame's scan extent, and dropout is the fraction of returns lost.

		P-LeWM			PD-JEPA		
Env.	**Perturbation**	Base	+MLP	+MLP$_z$	Base	+MLP	+MLP$_z$
Two-Room	Noise 0.25%	61.6$_{\pm 5.7}$	80.0$_{\pm 6.6}$	**80.2**$_{\pm 7.0}$	31.0$_{\pm 5.1}$	**44.4**$_{\pm 5.1}$	41.0$_{\pm 5.6}$
	Noise 0.5%	37.9$_{\pm 6.0}$	**53.8**$_{\pm 10.1}$	53.6$_{\pm 8.9}$	35.8$_{\pm 6.8}$	35.0$_{\pm 5.8}$	**36.4**$_{\pm 4.3}$
	Dropout 25%	96.1$_{\pm 5.9}$	**100.7**$_{\pm 5.3}$	99.3$_{\pm 5.4}$	60.4$_{\pm 6.4}$	71.4$_{\pm 7.5}$	**72.0**$_{\pm 6.8}$
	Dropout 50%	92.2$_{\pm 4.6}$	**99.5**$_{\pm 4.5}$	98.4$_{\pm 5.7}$	50.2$_{\pm 5.5}$	**63.4**$_{\pm 3.9}$	63.0$_{\pm 4.2}$
Reacher	Noise 0.25%	**91.8**$_{\pm 7.8}$	90.8$_{\pm 7.1}$	91.0$_{\pm 8.2}$	84.7$_{\pm 6.0}$	86.9$_{\pm 7.6}$	**89.2**$_{\pm 10.1}$
	Noise 0.5%	87.6$_{\pm 7.5}$	86.6$_{\pm 6.5}$	**89.1**$_{\pm 5.8}$	**58.0**$_{\pm 6.7}$	54.6$_{\pm 12.8}$	56.2$_{\pm 12.9}$
	Dropout 25%	94.8$_{\pm 5.0}$	**96.5**$_{\pm 9.2}$	**96.5**$_{\pm 8.0}$	89.1$_{\pm 8.7}$	**96.9**$_{\pm 8.9}$	94.1$_{\pm 7.7}$
	Dropout 50%	92.3$_{\pm 5.8}$	**96.5**$_{\pm 8.0}$	93.8$_{\pm 6.0}$	75.9$_{\pm 3.2}$	**86.9**$_{\pm 8.5}$	83.0$_{\pm 12.6}$
Push-T	Noise 0.25%	**3.6**$_{\pm 2.9}$	2.2$_{\pm 2.4}$	2.6$_{\pm 2.9}$	**4.0**$_{\pm 3.2}$	**4.0**$_{\pm 3.5}$	3.1$_{\pm 2.5}$
	Noise 0.5%	**4.5**$_{\pm 2.4}$	2.2$_{\pm 2.4}$	2.2$_{\pm 2.4}$	2.5$_{\pm 2.1}$	3.1$_{\pm 3.7}$	**3.4**$_{\pm 3.7}$
	Dropout 25%	88.6$_{\pm 6.3}$	**91.1**$_{\pm 6.0}$	90.2$_{\pm 8.6}$	87.0$_{\pm 8.2}$	**98.6**$_{\pm 7.3}$	95.2$_{\pm 7.6}$
	Dropout 50%	27.4$_{\pm 5.5}$	49.0$_{\pm 9.0}$	**49.8**$_{\pm 6.7}$	35.3$_{\pm 8.6}$	**51.4**$_{\pm 9.0}$	50.6$_{\pm 7.9}$
OGB-Cube	Noise 0.25%	97.0$_{\pm 12.0}$	100.0$_{\pm 11.8}$	**100.6**$_{\pm 10.5}$	93.5$_{\pm 6.8}$	84.9$_{\pm 7.8}$	**98.3**$_{\pm 7.6}$
	Noise 0.5%	89.7$_{\pm 15.9}$	**99.7**$_{\pm 11.7}$	97.6$_{\pm 11.2}$	77.2$_{\pm 9.4}$	79.6$_{\pm 9.5}$	**90.9**$_{\pm 8.3}$
	Dropout 25%	99.1$_{\pm 12.3}$	**100.6**$_{\pm 12.0}$	100.0$_{\pm 10.0}$	**98.8**$_{\pm 4.1}$	84.9$_{\pm 7.4}$	98.6$_{\pm 7.4}$
	Dropout 50%	**101.5**$_{\pm 8.5}$	101.2$_{\pm 12.3}$	100.0$_{\pm 13.0}$	99.5$_{\pm 4.4}$	85.4$_{\pm 5.6}$	**101.0**$_{\pm 6.6}$

Three regularities emerge. First, the conditioned module is at or above the base model in 19 of 24 cells, and the five exceptions lie within seed noise, three of them at the Push-T noise floor and two on OGB-Cube at most 1.5 points below. The largest gains appear where the base model loses most: Two-Room under range noise (+19 and +16 points for Point-LeWM, +10 for Point-Delta-JEPA at 0.25%), Two-Room under dropout for Point-Delta-JEPA (+12 and +13), Push-T under 50% dropout (+22 and +15) and OGB-Cube under 0.5% noise (+8 and +14). Second, the module recovers the observation-only level. In all but one cell the conditioned module lands within seed noise of the corresponding *observation degraded, goal clean* row of Table 10, the exception being Point-Delta-JEPA on OGB-Cube at 0.5% noise (90.9 vs. 98.3). Specifying the goal as a low-dimensional target thus removes the sensor from the goal channel, and what remains is the robustness of the observation encoder alone. Consequently the module does not repair failures on the observation side. Range noise on Push-T erases the thin task entities from the scan (Section 4) and stays below 5 relative points for every variant. Third, conditioning on z_t matters under degradation exactly where it matters on clean data. On Two-Room and Push-T the unconditioned and conditioned MLPs are tied within seed noise. On OGB-Cube the unconditioned MLP sits at 80 to 85 relative points for Point-Delta-JEPA regardless of the perturbation, below the base in three of four cells. This offset is not a degradation effect. It matches the unconditioned MLP's clean deficit on that environment (72.0 vs. 84.2 absolute in Table 3, about 86 relative), which arises because the OGB-Cube target leaves the arm configuration unconstrained. Taken together, planning toward a learned target latent is at least as robust to sensor degradation as planning toward an encoded goal cloud, and substantially more robust whenever the goal cloud itself is degraded.

D.2.1 Commanding goals the expert never aimed at

Every cube goal in the dataset is a placement on the table, so a cube held above it is a goal the expert never aimed at. We command 201 positions on five height layers up to 32 cm, each from five grasped start poses (blue in Figure 5), and compare against the strongest goal-cloud baseline: the specified state rendered and re-scanned in simulation. On all never-seen heights, target-to-latent matches it (0.68 vs. 0.67 cm, $p = 0.94$, Table 12) from the coordinates alone. Each method has one weak region:



(a) Rebuilt goal cloud: 0.66 cm median.

(b) Target-to-latent: 0.77 cm median.

Figure 5: **Specified cube goals.** 201 cube positions at five heights ($z = 2, 8, 16, 24, 32$ cm), each approached from five grasped start poses with the frozen Point-Delta-JEPA point-cloud world model. Target-to-latent maps the position, together with the latent of the current observation, to the goal latent. Planner and goal latents operate on LiDAR point clouds; the rendered scene is shown only for legibility. Colour is the median over the five starts of the smallest cube-to-goal distance, from 0 cm (green) to the 4 cm success threshold (red). Blue cubes mark the four corner start poses, and the red cube in the gripper marks the center start. Only $z = 2$ cm occurs as a goal in training.

Table 12: **Accuracy on unseen goals.** Cube goals at heights the expert never specified. Each 3-D position is approached from five grasped start poses with the same Point-Delta-JEPA point-cloud world model and CEM planner, so only the goal latent differs between target-to-latent, which predicts it from the position and the current latent, and the rebuilt goal cloud, which encodes the state rendered and re-scanned in simulation. Error is the cube's closest approach to the goal, and success uses the environment's 4 cm threshold. Δ gives the median paired difference between the two methods with a bootstrap 95 % interval, where negative values favour target-to-latent. Over all elevated goals the two are indistinguishable (Wilcoxon signed-rank, $p = 0.94$).

		Median error (cm)			Success (%)	
Goal height	n	T2L	Goal cloud	Δ [95 % CI]	T2L	Goal cloud
8 cm	165	0.76	0.68	+0.05 [−0.06, +0.14]	96	95
16 cm	210	0.58	0.51	+0.12 [+0.03, +0.22]	97	95
24 cm	210	0.63	0.68	−0.07 [−0.15, +0.01]	98	94
32 cm	210	0.85	0.85	−0.02 [−0.18, +0.09]	96	91
All	795	0.68	0.67	+0.02 [−0.05, +0.07]	97	94
outside sensor view†	35	2.20	1.98	+0.17 [−0.40, +0.73]	83	89
Goal input		pose	point cloud			

† Positions beyond the LiDAR's field of view; the rebuilt goal cloud does not contain the cube.

target-to-latent on the table layer, where the two goals differ (held vs. released cube), and the goal cloud in the far corners at height, which lie outside the LiDAR's field of view.

D.3 Attention

Figure 6 visualizes where the two end-to-end encoders look: the CLS-token attention is propagated through the ViT trunk by attention rollout, assigned from each group token to the points it covers, and normalised per cloud. Both models place their attention mass on the moving entities. In all four

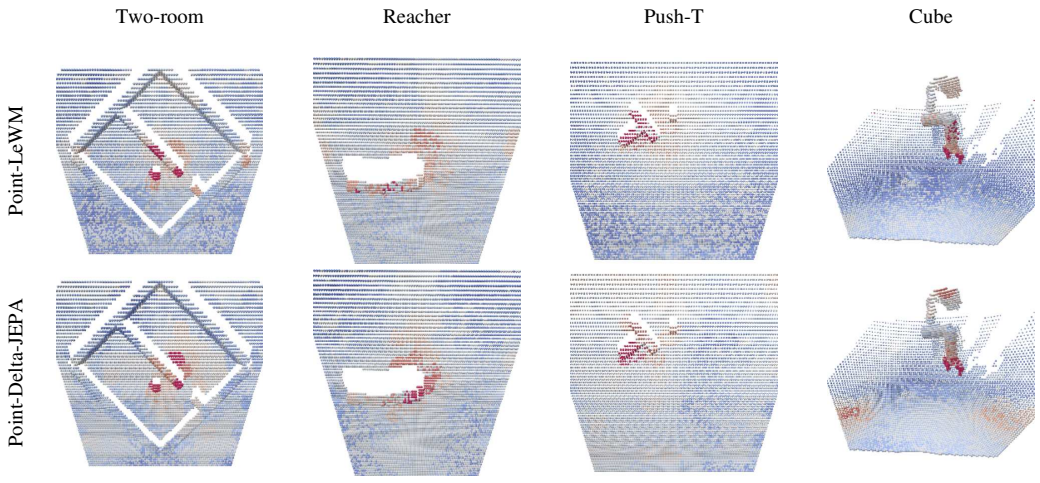


Figure 6: **Encoder attention.** CLS-token attention rollout of the Point-LeWM/Point-Delta-JEPA encoder mapped onto the input points. Warmer colours indicate higher attention, and points no group token covers are omitted. Columns are environments, rows the two models. Attention is normalised per point cloud. Utonia-WM is absent because its frozen PTv3 encoder has no CLS token and attends only locally within serialised patches, so it admits no comparable attention map.

environments Point-LeWM is sharply peaked on the entities, while Point-Delta-JEPA allots visibly more attention to ground and table.

E STATISTICAL TESTS

The tests compare the models within each family of Table 2, the image model against its point-cloud counterpart. The frozen-encoder family adds a second comparison, Utonia-WM against Vox-WM, which share voxel grid, predictor and projection and differ only in whether the voxel features are pretrained or training-free. Every model is scored on the same ten evaluation seeds, and each seed fixes the same 50 start–goal windows for every model, so the tests are paired by seed ($n = 10$). Pairing removes the seed-to-seed variation in episode difficulty from the comparison. Each model is a single trained checkpoint, so the tests capture the variance of evaluation over episodes, not the variance across training runs.

We follow the standard protocol for comparing two methods over several tasks (Demšar, 2006). The paired t-test on the ten per-seed differences is the primary test, reported with the 95% confidence interval of the mean difference. The Wilcoxon signed-rank test (Wilcoxon, 1945) repeats the comparison without assuming normally distributed differences, which ten seeds cannot verify. Each comparison spans four benchmarks, so all p-values are Holm–Bonferroni corrected over them (Holm, 1979). The correction keeps the probability of any false positive within a comparison at 0.05 and is uniformly more powerful than the plain Bonferroni correction. A non-significant difference is not evidence of parity, so wherever the text states that two models do not differ, the confidence interval bounds how large their difference can be.

SIGReg. No benchmark separates Point-LeWM from LeWM after correction under either test. The 2.8-point Push-T deficit is significant only before correction (t-test $p = 0.025$, Wilcoxon $p = 0.027$). The 95% intervals bound the difference to $[-8.1, +4.8]$ points over all four benchmarks and to within ± 6 points on Two-Room, Reacher and Push-T.

Action reconstruction. Point-Delta-JEPA trails Delta-JEPA on Reacher (-8.4) and Push-T (-23.4), significant under both tests. It does not differ from it on Two-Room and OGB-Cube, with intervals $[-0.3, +0.7]$ and $[-1.3, +7.7]$. On Two-Room the two models differ on a single seed, which leaves the signed-rank test without power, while the interval is narrow.

Table 13: **Paired tests within the families of Table 2.** Difference of means Δ (points, first model minus second) with its 95% confidence interval, and Holm–Bonferroni corrected p-values of the paired t-test and the Wilcoxon signed-rank test over the four benchmarks of each comparison ($n = 10$ seeds). With ten seeds the exact Wilcoxon p cannot fall below 0.002, or 0.008 after correction.

	Δ	95% CI	t-test p	Wilcoxon p
SIGReg: Point-LeWM − LeWM				
Two-Room	$+1.4$	$[-2.0, +4.8]$	0.75	0.86
Reacher	-1.6	$[-5.9, +2.7]$	0.75	0.86
Push-T	-2.8	$[-5.2, -0.4]$	0.099	0.11
OGB-Cube	-3.8	$[-8.1, +0.5]$	0.24	0.30
Action reconstruction: Point-Delta-JEPA − Delta-JEPA				
Two-Room	$+0.2$	$[-0.3, +0.7]$	0.34	1.00
Reacher	-8.4	$[-10.4, -6.4]$	< 0.001	0.008
Push-T	-23.4	$[-29.4, -17.4]$	< 0.001	0.008
OGB-Cube	$+3.2$	$[-1.3, +7.7]$	0.29	0.50
Frozen encoder: Utonia-WM − DINO-WM				
Two-Room	-5.8	$[-7.2, -4.4]$	< 0.001	0.008
Reacher	-5.8	$[-11.4, -0.2]$	0.044	0.047
Push-T	-29.4	$[-37.0, -21.8]$	< 0.001	0.008
OGB-Cube	-7.4	$[-11.4, -3.4]$	0.005	0.012
Voxel features: Utonia-WM − Vox-WM				
Two-Room	$+10.2$	$[+6.9, +13.5]$	< 0.001	0.008
Reacher	-0.6	$[-5.1, +3.9]$	0.77	0.74
Push-T	-9.2	$[-12.6, -5.8]$	< 0.001	0.008
OGB-Cube	$+6.0$	$[+2.3, +9.7]$	0.010	0.031

Frozen encoder. Utonia-WM trails DINO-WM on all four benchmarks, significant after correction under both tests, with Reacher borderline (t-test $p = 0.044$, Wilcoxon $p = 0.047$). Against Vox-WM, Utonia-WM is better on Two-Room ($+10.2$) and OGB-Cube ($+6.0$) and worse on Push-T (-9.2), all significant under both tests, and does not differ on Reacher (-0.6, interval $[-5.1, +3.9]$).

Correction scope. We correct within each comparison, since each answers its own question. Correcting over all 16 tests instead changes three decisions: the frozen-encoder gap on Reacher loses significance under both tests, and the OGB-Cube gaps of the frozen-encoder and voxel-feature comparisons lose it under the Wilcoxon test.

F Broader impacts

This work is foundational research on world-model learning in simulated tabletop environments and is not tied to a particular application or deployment. Its societal effects are those generic to improved robot autonomy. Range-based world models could benefit assistive and industrial robots that must operate where cameras fail, and lower the cost of adapting robots to new tasks. Conversely, more capable autonomous systems can displace labor and could be misused, for example in surveillance platforms that favor range sensors precisely because they work in darkness. The models released here are trained on four simulated benchmarks and pose no direct risk, and we see no mitigations required beyond the norms of open release.