# JEPATH: A WORLD MODEL FOR FIRST-PERSON NAVIGATION

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

## ABSTRACT

JEPA world models combined with test-time trajectory optimization have recently achieved strong results on tasks conditioned on a single goal image. However, most of these results assume fully observable environments, and their adaptation to first-person navigation raises specific challenges. In this study, we introduce PADoom, a benchmark built on ViZDoom where the same environments and trajectories can be observed either from a fully observable top-down view or from a first-person view. We show that perceptual aliasing, where different locations produce nearly identical observations, requires combining visual and spatial information, and that the spatial signal must avoid absolute position encoding to generalize across maps. Based on these findings, we propose JEPath, a new JEPA world model that jointly predicts future visual features and ego-motion from observations and actions. Unlike models that predict absolute position, JEPath predicts motion and integrates it along candidate paths. The planner scores paths by distance to the goal rather than visual similarity, resolving perceptual aliasing while still relying on vision to handle navigation-relevant scene structure such as obstacles. On PADoom, JEPath outperforms recent world models and goal-conditioned policies for first-person navigation. It also generalizes substantially better to unseen maps, where competing methods experience a pronounced performance drop. The benchmark and models will be released upon acceptance.

## 1 INTRODUCTION

A world model (WM) learns how the environment responds to the actions of an agent and is used instead of the environment to plan or generate experience in imagination (Sutton, 1991; Ha & Schmidhuber, 2018). JEPA WMs (LeCun, 2022; Assran et al., 2025) are reward-free methods that predict future latent features, so that one model of the dynamics can serve many tasks. The task is then specified as a goal image, and the planner performs trajectory optimization to find action sequences that lead to a latent state closest to the goal. Recent approaches use a frozen feature space (Zhou et al., 2025), or straighten it to facilitate planning, as in Temporal Straightening (TS) (Wang et al., 2026). Provided the goal is within the planning horizon, these methods reach these goals through planning alone, with no task-specific training.

JEPA WMs have been applied to scenes that a fixed camera sees almost entirely, a tabletop or a maze seen from above, where every observation identifies a state almost certainly, an assumption this recipe relies on. Most agents that act in the world do not enjoy such a fully informative view. They see the scene from inside, through a sensor that moves with them and covers only part of their surroundings. First-person navigation, studied by Bar et al. (2025); Zhang et al. (2026), is the simplest instance of this general case, and the one we study here.

Successful planning with JEPA WMs requires two main conditions: the cost function must reflect spatial (geodesic) proximity to the goal, and the world model must accurately represent the environment dynamics, faithfully predicting the positions of the ego agent or scene-critical elements such as walls or obstacles. In this paper, we show that state-of-the-art JEPA WMs do not fulfill these assumptions in first-person navigation, requiring non-trivial adaptations.

To identify the main challenges this raises, we introduce PADoom, a benchmark built on ViZDoom (Kempka et al., 2016). We generate a variety of maps that vary in geometry and appearance, and render every scene in two synchronized views: a top-down view, fully observable like the fixed



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.

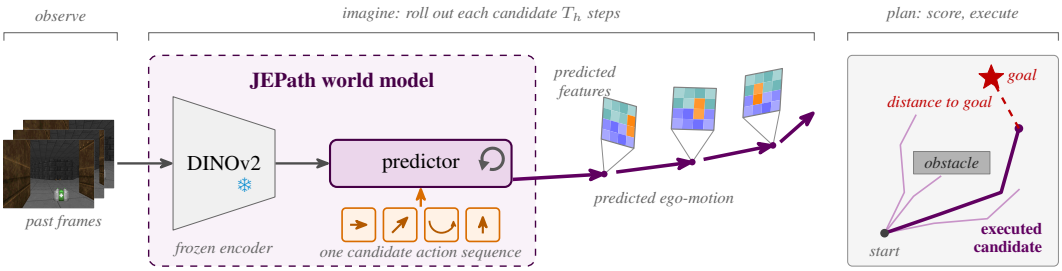



Figure 1: **Planning with JEPath.** From past frames encoded by a frozen DINOv2, JEPath rolls out candidate action sequences, predicting the next visual features and ego-motion. Integrated from the current position, the ego-motions predict each candidate path (right); the one ending nearest the goal is executed (bold path). Scoring paths by distance to the goal rather than visual similarity resolves perceptual aliasing: reading motion from vision handles obstacles and transfers to unseen maps.

camera of DINO-WM (Zhou et al., 2025), and the first-person view. By isolating full from partial observability in PADoom, we draw three main conclusions. First, perceptual aliasing (Whitehead & Ballard, 1991) is a major issue in first-person navigation: since a rendered image can correspond to several possible locations, planning or learning the transition model from visual information alone is a dead end. Second, combining spatial and visual information is crucial to plan and navigate through the environment with walls and obstacles. Last, one must depart from absolute spatial position encoding for the WM and planner to generalize zero-shot to new, unseen environments.

Building on these insights, we introduce JEPath, a JEPA model that plans over paths, illustrated in Fig. 1. The key idea behind JEPath is to jointly predict future visual features and ego-motion. The goal is given as a position and not an image, so two places that may look similar induce different planning costs. A frozen DINOv2 backbone encodes past frames, and JEPath rolls out several candidate action sequences, producing at each step the future features and ego-motion. Planning is performed by integrating these motions into the path each candidate would follow (Fig. 1, right). The planner then executes the candidate sequence of actions that ends closest to the goal. The result of our design choices is a world model that plans effectively from a first-person view and generalizes to new maps. Our contributions are threefold:

• We release PADoom, a first-person navigation benchmark of 100 procedurally generated maps, 50 for training and 50 held out, where every scene is also rendered from a fully observable top-down view, so that partial observability is the only variable. From PADoom we draw important conclusions explaining the limitations of state-of-the-art models in first-person navigation, providing insights for the community to develop more effective solutions.

• We propose JEPath, a JEPA world model that predicts future visual features and ego-motion, and plans over the path obtained by integrating that motion. By predicting ego-motion from images alone, we avoid error accumulation in spatial forecast rollouts. By departing from absolute position, we endow the model with generalization abilities in unseen environments and new maps.

• Experiments on PADoom show that JEPath significantly outperforms recent JEPA world models and goal-conditioned policies. The gap is even more pronounced on the held-out maps, where the baselines' performance drops. We also quantify the perceptual aliasing faced by visual JEPA WMs, and shed light on the inadequacy of specific regularizers, e.g., TS (Wang et al., 2026), in this regime. Finally, we validate several design choices, such as the importance of ego-motion prediction.

## 2 Related Work

**Task-agnostic world models.** We study world models trained to predict future latent features without a reward (LeCun, 2022), so that the same dynamics can serve any task and the goal is given as an observation: the model encodes a goal image, rolls out candidate action sequences in latent space, and selects the one whose predicted state lies closest to the goal encoding. Reward-conditioned world models (Hafner et al., 2025; Schrittwieser et al., 2020; Hansen et al., 2024) specify the goal through a scalar rather than an observation and are outside our scope. In our scope, DINO-WM (Zhou

et al., 2025) plans directly in the DINOv2 feature space, LeWorldModel (Maes et al., 2026) in a compressed latent trained end to end, and temporal straightening (Wang et al., 2026) regularizes the latent trajectory with physical priors. Three properties of the setting limit what reward decoupling buys. The task is a single static goal image over a short horizon, assumed discriminative, so aliasing never arises. The scene is observed exocentrically and in full with a different model per environment family, so partial observability is never tested and the learned dynamics are indistinguishable from a map of the training scene. The benchmarks of this line of work are built accordingly, fully or almost fully observable and with discriminative goals: PointMaze (Fu et al., 2020), Push-T (Chi et al., 2025), the Wall, Rope and Granular environments of DINO-WM (Zhou et al., 2025), OGBench (Park et al., 2025), DMControl (Tassa et al., 2018) and Atari (Bellemare et al., 2013; Farebrother & Castro, 2024). Spatial information enters as the agent's absolute pose, projected and concatenated to every visual token, which anchors the dynamics to one coordinate frame and leaves the predictor consuming its own pose estimate during the rollout, integrating it with no observation to correct the drift.

**Egocentric navigation world models.** Pixel-space diffusion world models render games interactively from the player's viewpoint as in DIAMOND (Alonso et al., 2024) for Counter-Strike and GameNGen (Valevski et al., 2025) for Doom. Navigation World Models (NWM, Bar et al., 2025) bring this class to real egocentric navigation using a conditional diffusion transformer trained on robot and human egocentric video across several embodiments. It predicts the next frame from past frames, a translation and yaw action. Unlike goal-conditioned policies such as GNM, ViNT and NoMaD (Shah et al., 2023a;b; Sridhar et al., 2024) which are trained by behavior cloning and fixed afterward, NWM is goal-agnostic and plans at test time: it samples action sequences with the cross-entropy method, rolls them out and scores the decoded final frame by perceptual similarity to a goal image, or applies the same score to rank trajectories proposed by NoMaD. RAE-NWM (Zhang et al., 2026) keeps this formulation and predicts in a dense visual representation space rather than the VAE latent. Two assumptions come with using a goal image. First, the scene must be known. A goal image taken outside the field of view can only be matched by imagining what lies beyond the view, which the model can do only for a scene it was trained on. Therefore, success measures how well the model memorized the scene and how densely the planner samples it. Second, the goal image is assumed to identify a unique state, ignoring perceptual aliasing. The evaluation never tests this: plans are scored by trajectory error against the recorded trajectory that produced the goal image and are never executed, so no observation is ever taken from the plan's endpoint.

**Visual odometry.** In robotics, planning from first-person view often involves visual odometry (VO) (Janny et al., 2025). VO recovers the ego-motion from one frame to the next, regressed end-to-end by DeepVO (Wang et al., 2017), predicted from optical flow by TartanVO (Wang et al., 2020), or from keypoints on frozen DINOv2 features by DINO-VO (Azhari & Shim, 2025). In PointNav (Anderson et al., 2018), where the goal is a position and a GPS+compass sensor localizes the agent, a learned odometer replaces that sensor. Datta et al. (2020) integrate per-action estimates into a running pose, and Zhao et al. (2021) and Partsey et al. (2022) condition the estimate on the action taken, which helps planning. Compared to this line of work, our ego-motion head takes almost the same inputs as an action-conditioned VO estimator, minus one: JEPath never observes the frame it is predicting. It instead regresses the ego-motion with a linear head from the current latents and the action alone, in the same pass that predicts the next latent, without ever decoding a pixel. Here, these motions are chained along a candidate action sequence and trace the path the planner scores (Fig. 1).

## 3 Planning from the First Person

This section recalls planning with a latent world model and states two necessary conditions to reach the goal (Sec. 3.1). We show with PADoom how the first-person view breaks each of them (Sec. 3.2), and JEPath follows from what the broken conditions require (Sec. 3.3).

### 3.1 Planning with a Latent World Model

We consider goal-reaching problems in which an agent acts in an environment through discrete actions $a_t \in \mathcal{A}$ and must reach a goal within a fixed budget of actions. At every step, it receives an observation $o_t$, an image viewed either from the first person or from above, and occupies a pose $p_t = (q_t, \theta_t)$, where $q_t$ is its position and $\theta_t$ its heading. The agent's own pose is observed, as in

PointNav with a GPS and compass sensor (Anderson et al., 2018) while the map is not. A goal is specified either by an image $o_g$ taken at position $q_g$ (image goal) or by the position $q_g$ itself (spatial goal), on a map covered by the trajectories (seen) or on another map (unseen). The goal is reached when $\|q_t - q_g\|_2 \leq \delta$, for a success radius $\delta$.

Before any task is specified, the only data available are trajectories $(o_{0:T}, p_{0:T}, a_{0:T-1})$ of observations, poses and actions, collected by unknown policies. A world model learned from these trajectories stands in for the environment. At test time, a planner rolls candidate action sequences forward in the model, scores their imagined endpoint with a cost instead of the goal test, and replans over a receding horizon. Each candidate action sequence traces a *path*, the route through the map the agent would follow, and the cost scores how close to the goal that imagined path ends.

**Imagined rollout.** An encoder $E_\phi$ maps each observation to a latent vector $z_t = E_\phi(o_t)$, and a learned pose embedding $\zeta$ optionally turns the pose into a token of its own, $u_t = \zeta(p_t)$. A hat marks a predicted quantity throughout, so $\hat{z}_{t+1}$ is the predicted next latent and $\hat{u}_{t+1}$ the predicted next pose token. A predictor $f_\varphi$ forecasts both from the $K$ most recent latent vectors, the pose tokens of the same frames when the model has them, and the current action $a_t$,

$$\big(\hat{z}_{t+1},\, [\hat{u}_{t+1}]\big) = f_\varphi\big(z_{t-K+1:t},\, [u_{t-K+1:t}],\, a_t\big), \tag{1}$$

where the bracketed terms are the optional pose pathway. Over a planning horizon $T_h$, Eq. (1) is applied recursively to a candidate action sequence $a_{0:T_h-1}$, initialized from the $K$ most recent observations and, when available, the current pose $p_0$. This produces an imagined endpoint $(\hat{z}_{T_h}, [\hat{u}_{T_h}])$. No new observation arrives within the horizon to correct the rollout.

**Objective and planner.** Planning seeks the action sequence $a^\star_{0:T_h-1}$ that minimizes a cost $J$, with $\omega \in [0, 1]$ weighting an image term $J_{\text{img}}$ against a spatial term $J_{\text{pos}}$:

$$a^\star_{0:T_h-1} = \arg\min_{a_{0:T_h-1} \in \mathcal{A}^{T_h}} J(a_{0:T_h-1}), \qquad J = (1-\omega)\, J_{\text{img}} + \omega\, J_{\text{pos}}. \tag{2}$$

The two terms score the two goal modalities. An image goal $o_g$ is compared in latent space, $J_{\text{img}} = \|\hat{z}_{T_h} - E_\phi(o_g)\|_2^2$. A spatial goal, the pose $p_g$ at position $q_g$, is compared in one of two ways. The first compares pose tokens, $J_{\text{pos}} = \|\hat{u}_{T_h} - \zeta(p_g)\|_2^2$, a proxy for the distance. The second decodes the position $\hat{q}_{T_h} = \psi(\hat{u}_{T_h})$ from the predicted pose token and compares positions, $J_{\text{pos}} = \|\hat{q}_{T_h} - q_g\|_2^2$, the distance measured by the success test.

The minimization is approximated by the Cross-Entropy Method (CEM, Rubinstein, 1997), the standard planner both in model-based control (Chua et al., 2018; Hafner et al., 2019) and in recent works planning with latent world models (Zhou et al., 2025; Bar et al., 2025). A discrete action set changes only its sampling distribution. The discrete form (de Boer et al., 2005) refits a table of per-step categorical logits on the elites at every iteration, see App. B for details. The planner executes the first $T_e \leq T_h$ actions, observes again and replans until the budget is spent or the goal reached with $T_h$ and $T_e$ fixed.

**Two conditions for successful planning.** The planner picks the candidate with the lowest cost $J$ on its *imagined* endpoint and executes it, so it reliably reaches the goal when two conditions hold:

> **(I)** ***The cost smoothly reflects the geodesic distance to the goal:*** the smaller the geodesic distance to the goal, the smaller the cost should be.
> **(II)** ***The transition model ranks like the world:*** the candidate with the lowest imagined cost is also the one with the lowest cost when executed.

### 3.2 PADOOM: A PERSPECTIVE AND ABSTRACT DOOM BENCHMARK

So far, the world model planning recipe has been run under a fixed camera that sees the whole scene (Sec. 2). There an image goal names one place, and a rollout starts from a frame that shows everything the plan will meet, so both conditions hold. From the first person, one view fits many poses and what lies beyond it is unseen. In this context, whether either condition holds has not been measured. PADoom is built to measure it, on top of the ViZDoom platform (Kempka et al., 2016).

**Observability.** At every step the agent observes its pose $p_t = (q_t, \theta_t)$ and an image $o_t$, rendered in one of two views of the same scene (Fig. 2): the first-person view, PDoom, which is partially observable, and the top-down view, ADoom, which is fully observable. Dynamics, actions and maps are shared, so any performance gap between the two is the cost of observability alone. That gap measures whether an image cost still meets (I) and a learned model still meets (II), once the view no longer specifies the pose.

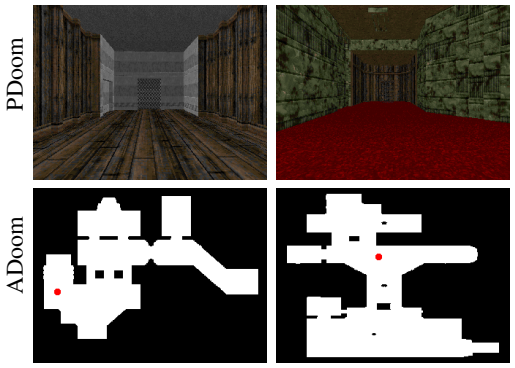



Figure 2: **PADoom maps, synchronized views.** Top: the first-person view (PDoom). Bottom: the top-down view (ADoom) of the same scene, walkable floor in white and the agent the red dot.

**Generalization.** The maps come from the open-source level generator Obsidian (Obsidian Level Maker Contributors, 2025), which produces them in unlimited number. Maps contain rooms joined by corridors, with pillars and crates inside, and have their own textures (Fig. 2), so the appearance changes from map to map. Every evaluated goal lies behind an obstacle, off the straight line from the starting point. World models are trained on trajectories from a training set of maps and planning is scored on those and on a held-out set, so the benchmark separates what a model learned about the world from what it learned about the maps it saw. Sec. 4.1 details the trajectories, the start–goal pairs evaluated on each set, and their size.

**Pathological cases.** A script compiled into every PADoom map restores the agent's pose and velocity exactly, so a planner can copy its current state into parallel engine instances and roll its candidates out there, so a copy of the game engine stands in for the world model wherever the model has to be taken out of the equation. PADoom uses it to build three dedicated maps, each isolating one situation that breaks the planner by violating one of the two conditions of Sec. 3.1.

- ***Perceptual aliasing***: in a map without landmarks, many positions look the same from the first person, so an image of the goal does not say where the goal is, and the image cost leads the planner to places that only look like it (condition (I) fails).
- ***Unobserved obstacle***: a wall stands between the agent and the goal, and a spatial cost points straight through it, so a planner whose model does not observe the wall drives into it and stays pinned (condition (II) fails).
- ***Unvisited coordinates***: the goal lies in a region no training trajectory reached, so a model fed its absolute position mispredicts at coordinates it never learned (condition (II) fails).

Together, the three cases described in more detail in App. A set three requirements: a spatial goal, a transition model that relies on vision and a spatial signal not tied to the coordinates of one map.

### 3.3 JEPath: Predicting Ego-Motion for Spatial Planning

To meet the above requirements, we introduce JEPath, a world model that predicts its own ego-motion, and the planner built upon it (Fig. 1). Each pathological case above specifies one part of the design: perceptual aliasing calls for a spatial goal, the unobserved obstacle calls for a motion read from the frames and the unvisited coordinates call for a motion in the body frame, tied to no map.

**From path to ego-motion.** Spatial planning scores an imagined path by its endpoint alone, through the distance $\|\hat{q}_{T_h} - q_g\|_2$ to the goal. That distance depends only on where the path ends relative to the goal, set by how the agent moves from its current pose, not by where that pose lies on the map.

A path is thus defined by its starting pose and one ego-motion $m_t = (m_t^q, m_t^\theta)$ per step, the motion to the next pose expressed in the agent's own frame. Each step applies it to the current pose, $p_{t+1} = p_t \oplus m_t$, where $\oplus$ composes two planar rigid motions: the heading turns by $m_t^\theta$, $\theta_{t+1} = \theta_t + m_t^\theta$, and the agent moves by $m_t^q$ in the direction it faces, $q_{t+1} = q_t + R(\theta_t)\, m_t^q$, with $R(\theta)$ the planar rotation by $\theta$. Conversely, the ego-motion between two recorded poses is the next

pose seen from the current one, $m_t = p_t^{-1} \oplus p_{t+1}$, the inverse of that step,

$$m_t^\theta = \theta_{t+1} - \theta_t, \qquad m_t^q = R(\theta_t)^\top (q_{t+1} - q_t), \tag{3}$$

with the angle difference wrapped to $(-\pi, \pi]$.

Therefore, the model never has to know where the agent is. It only needs to predict the motions; the planner, which does observe the current pose, chains them from it and scores where they end.

**Model.** JEPath pairs the frozen encoder $E_\phi$ with a causal predictor $f_\varphi$ that reads the $K$ latest latents $z_t = E_\phi(o_t)$ and the action $a_t$. It emits the next latent and the ego-motion of the step,

$$(\hat{z}_{t+1}, \hat{m}_t) = f_\varphi\big(z_{t-K+1:t},\, a_t\big), \tag{4}$$

where $\hat{m}_t$ estimates the ego-motion $m_t$ of Eq. (3). A conventional model decodes a predicted pose token with $\psi$ (Eq. (1)); JEPath instead predicts the motion and integrates it. Its design (Fig. 3) makes three decisions that keep the predicted motion grounded in what the agent sees.

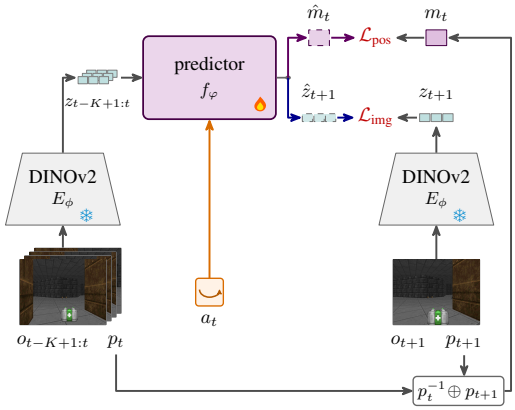



Figure 3: **JEPath.** From frozen DINOv2 latents of context frames and action $a_t$, the predictor $f_\varphi$ predicts next latent $\hat{z}_{t+1}$ and ego-motion $\hat{m}_t$.

*It reads no pose.* JEPath takes only frames and actions as input. Poses serve only as training targets, so motion must be predicted from visual cues: a wall seen ahead, for instance, implies a blocked step. Without absolute coordinates, it generalizes to new maps and unvisited regions. A model fed the pose token $\zeta(p_t)$ could instead predict from $p_t$ and $a_t$ alone and overfit the training coordinates.

*The motion is a readout of the visual predictor.* Ego-motion $\hat{m}_t$ is predicted jointly with future latents in a single forward pass (Fig. 3). To predict the next latent, the predictor must already know how the agent moves: a step forward and a turn change the view differently, and a wall ahead stops the view from changing. Reading the motion from these same features lets it use everything the predictor learns about the scene.

*The visual target is frozen.* The encoder $E_\phi$ is applied to context and next frame alike and never finetuned, so the target is a fixed function of the image, with nothing for the predictor to collapse onto and nothing the motion loss can distort.

**Training objective.** The model is trained by teacher forcing on clips of $T_c$ frames, each predicting the next in one causal pass. Its two targets are the frozen latent $z_{t+1} = E_\phi(o_{t+1})$ and the ego-motion $m_t = p_t^{-1} \oplus p_{t+1}$ of the recorded poses (Fig. 3, right). The loss $\mathcal{L} = \frac{1}{T_c} \sum_t (\mathcal{L}_{\text{img}} + \mathcal{L}_{\text{pos}})$ sums the two smooth-$\ell_1$ errors: $\mathcal{L}_{\text{img}} = \text{smooth}_{\ell_1}(\hat{z}_{t+1}, z_{t+1})$ and $\mathcal{L}_{\text{pos}} = \text{smooth}_{\ell_1}(\hat{m}_t, m_t)$.

**Implementation.** The encoder $E_\phi$ is a frozen DINOv2 (Oquab et al., 2024) on the RGB frames: each latent keeps its $N$ patch tokens, one per image patch, and its `[CLS]` token, a single token summarizing the whole image. The predictor $f_\varphi$ is a causal ViT, each frame conditioned by AdaLN (Peebles & Xie, 2023) on its action, with the next latent read from its outputs and the motion from its predicted `[CLS]` token by a linear head each. This choice is tied to the ViT encoder and gives the best readout among the aggregations compared in App. E.1 (Tab. 5). App. B gives the token-level architecture, the ego-motion encoding, the model sizes and training hyperparameters, and the planner settings.

**Closed-loop planning.** JEPath plans with CEM (Sec. 3.1; on par with MCTS, App. E.2), refitting its sampling distribution on the lowest-cost candidates with the spatial cost alone, i.e., $\omega = 1$. Each candidate $a_{0:T_h-1}$ is rolled out with Eq. (4) from the $K$ latest observations. The predicted latent is fed back at every step while the predicted ego-motion is never. The ego-motions are chained from the origin of the current frame, $\hat{p}_{t+1} = \hat{p}_t \oplus \hat{m}_t$ (App. B), into the dreamed path, whose endpoint $\hat{q}_{T_h}$ alone is scored by $J_{\text{pos}}$ against the goal $q_g$ (Fig. 1). The whole sequence is executed before replanning, $T_e = T_h$. With $\omega = 1$, the cost no longer depends on the image, so aliasing cannot break condition (I). In addition, Sec. 4 measures whether the model ranks like the world (II).

# 4 EXPERIMENTS

In this section, we aim to evaluate JEPath across the aspects offered by PADoom. Our experiments are designed to answer four questions: **(1) Planning on seen maps:** does predicting its own motion help JEPath plan better than world models and goal-conditioned policies trained on the same trajectories? **(2) Generalization to unseen maps:** does that hold on maps never seen in training? **(3) Cost of the first-person view:** on the seen maps, what do world models lose when the view goes from top-down to first-person? **(4) What drives performance:** does an image term add anything to the spatial cost, and how do the horizon, the distance to the goal and the planner affect performance?

## 4.1 EXPERIMENTAL PROTOCOL

**Training dataset.** A world model used for planning must still be trained for the world-modeling problem, not for the planning one. Its data has to be decoupled from the downstream task; otherwise the model hallucinates wherever a plan leaves the transitions it was trained on (Hansen & Wang, 2026). With a discrete action set as in PADoom, using expert actions or a random policy would fail: an expert never samples the actions that do not serve its targets whereas a random policy never builds up the speed a committed plan reaches; either way, the planner would optimize over dynamics the model has never seen. The training trajectories of PADoom are therefore collected to cover the whole action vocabulary along paths that actually go somewhere: 585 episodes from each of the 50 training maps, 29250 in all. The collection procedure is detailed in App. C.

**Evaluation dataset.** Evaluation episodes are start–goal pairs cut from separate collections on the same maps, connected by a shortest path so that every target is backed by a feasible action sequence and stays reachable within the planning budget. The targets are deliberately not drawn from the training trajectories: reusing them can result in episodes solvable by replaying the recorded actions, without any planning at all. We build two sets with different difficulties: the Normal sets, 499 pairs in domain and 500 out of domain, on which the straight line to the goal is blocked, and the Hard sets, 490 and 500 pairs, on which a goal-bearing controller and a random policy both fail as well. The collection and pair-selection procedures are detailed in App. C.

**Metrics.** We report the success rate (SR↑), the share of episodes ending within 25 world units of the goal (Sec. 3.1), and success weighted by path length (SPL↑) (Anderson et al., 2018), which discounts a success by its detour from the shortest route. Both are defined in App. C.3.

**Baselines.** Model-free policies map the current image and the goal coordinates to an action: goal-conditioned variants of behavior cloning (GCBC) and implicit Q-learning (GCIQL, Kostrikov et al., 2022). We also define Blind-GCIQL, which is GCIQL fed with black frames and measures what the goal coordinates alone achieve. The world models are DINO-WM (Zhou et al., 2025), JEPA-WM$_{\text{prop}}$[1], and Temporal Straightening (TS, Wang et al., 2026). We also use two diffusion WMs for navigation, the Navigation World Model (NWM, Bar et al., 2025) and its representation-space variant RAE-NWM (Zhang et al., 2026), which both plan as described in Sec. 3.1. Neither NWM nor RAE-NWM predicts a pose, so both plan on the image cost alone. TS is a latent world model whose projector over the frozen encoder is trained with a cosine term that penalizes turning between successive latent differences, so that the latents along a trajectory follow a straight line. Architectures, parameter counts, training objectives and the cost each model plans with are described in App. D.

## 4.2 MAIN RESULTS: PLANNING

JEPath is compared with model-free policies and world models on 50 seen and 50 held-out maps (Tab. 1), and with TS under both observation modes (Tab. 2).

Tab. 1 shows that JEPath clearly outperforms the baselines, by 12 to 31 points SR over the best of them. Methods that plan or act on the goal position, JEPath, the *prop* methods and the policies, do not suffer from perceptual aliasing, which explains the lower scores of the image-goal world models (DINO-WM, NWM and RAE-NWM). On unseen maps, predicting the ego-motion, a body-frame quantity tied to no map, transfers better than taking an absolute position as input or predicting one.

[1] A prop subscript marks models whose cost adds the spatial term to the image one.

Table 1: **Planning.** Success rate (SR) and success weighted by path length (SPL), in %, on the Normal and Hard pairs of the 50 training maps (seen) and the 50 held-out maps (unseen), mean ± std over 3 planner seeds, except NWM and RAE-NWM, which are single seeds sampled. Bold/underline: best/second best per column.

| | Seen | | | | Unseen | | | |
|---|---|---|---|---|---|---|---|---|
| | Normal | | Hard | | Normal | | Hard | |
| | SR | SPL | SR | SPL | SR | SPL | SR | SPL |
| GCIQL | <u>75.8</u>$_{\pm 1.3}$ | <u>62.0</u>$_{\pm 1.3}$ | 50.0$_{\pm 2.7}$ | 39.6$_{\pm 3.1}$ | <u>70.1</u>$_{\pm 0.8}$ | 56.4$_{\pm 1.1}$ | 47.3$_{\pm 0.3}$ | 38.0$_{\pm 0.2}$ |
| Blind-GCIQL | 42.4$_{\pm 4.5}$ | 36.0$_{\pm 3.6}$ | 20.7$_{\pm 1.6}$ | 17.3$_{\pm 0.8}$ | 43.5$_{\pm 5.4}$ | 36.9$_{\pm 5.3}$ | 25.3$_{\pm 1.3}$ | 20.7$_{\pm 1.3}$ |
| GCBC | 73.9$_{\pm 3.0}$ | 60.8$_{\pm 3.0}$ | <u>50.7</u>$_{\pm 0.7}$ | <u>41.1</u>$_{\pm 0.6}$ | 70.0$_{\pm 2.4}$ | <u>57.6</u>$_{\pm 2.7}$ | <u>48.4</u>$_{\pm 1.9}$ | <u>39.7</u>$_{\pm 2.8}$ |
| DINO-WM | 38.9$_{\pm 2.3}$ | 24.6$_{\pm 1.4}$ | 27.5$_{\pm 0.8}$ | 16.7$_{\pm 0.3}$ | 25.3$_{\pm 1.7}$ | 16.3$_{\pm 0.7}$ | 18.1$_{\pm 1.2}$ | 10.8$_{\pm 0.6}$ |
| DINO-WM$_{\text{prop}}$ | 48.7$_{\pm 2.1}$ | 30.6$_{\pm 2.1}$ | 37.6$_{\pm 2.1}$ | 24.0$_{\pm 1.3}$ | 41.3$_{\pm 2.3}$ | 26.4$_{\pm 1.9}$ | 31.9$_{\pm 0.7}$ | 19.8$_{\pm 0.2}$ |
| JEPA-WM$_{\text{prop}}$ | 53.0$_{\pm 3.1}$ | 34.9$_{\pm 1.7}$ | 39.5$_{\pm 1.3}$ | 25.4$_{\pm 0.8}$ | 36.6$_{\pm 2.3}$ | 23.3$_{\pm 2.0}$ | 27.1$_{\pm 1.9}$ | 16.9$_{\pm 1.3}$ |
| NWM | 19.4 | 12.7 | 13.5 | 7.7 | 16.6 | 10.4 | 18.0 | 10.6 |
| RAE-NWM | 20.0 | 12.7 | 7.8 | 4.9 | 19.4 | 12.8 | 8.8 | 5.7 |
| **JEPath (ours)** | **88.2**$_{\pm 0.6}$ | **63.8**$_{\pm 0.3}$ | **73.9**$_{\pm 1.2}$ | **51.8**$_{\pm 1.1}$ | **89.7**$_{\pm 0.3}$ | **63.5**$_{\pm 0.6}$ | **79.4**$_{\pm 1.6}$ | **55.5**$_{\pm 0.8}$ |

Table 2: **PADoom evaluation.** SR and SPL (%) on PADoom. Bold/underline: best/second best per column. TS: temporal straightening (Wang et al., 2026).

| | ADoom | | PDoom | |
|---|---|---|---|---|
| | SR | SPL | SR | SPL |
| DINO-WM | 35.3$_{\pm 3.6}$ | 22.5$_{\pm 2.4}$ | 27.5$_{\pm 0.8}$ | 16.7$_{\pm 0.3}$ |
| DINO-WM$_{\text{prop}}$ | 30.7$_{\pm 4.3}$ | 21.1$_{\pm 2.6}$ | <u>37.6</u>$_{\pm 2.1}$ | <u>24.0</u>$_{\pm 1.3}$ |
| TS | <u>45.4</u>$_{\pm 2.6}$ | <u>25.6</u>$_{\pm 1.6}$ | 8.4$_{\pm 2.2}$ | 4.9$_{\pm 1.3}$ |
| **JEPath (ours)** | **51.3**$_{\pm 1.0}$ | **32.2**$_{\pm 1.0}$ | **73.9**$_{\pm 1.2}$ | **51.8**$_{\pm 1.1}$ |

On unseen maps, JEPath loses nothing while DINO-WM$_{\text{prop}}$ and JEPA-WM$_{\text{prop}}$ drop 6 to 16 points SR. The low scores of NWM and RAE-NWM, around 5% to 20% SR, are attributed to their reliance on image cost for planning and heavy compute to generate reliable latent features. We attribute the lead of DINO-WM$_{\text{prop}}$ over DINO-WM to the pose input and the spatial term of its cost, which provides during the rollout the spatial information the first-person view withholds. The Model-free methods, GCIQL and GCBC, follow immediately after JEPath. These methods map the observation and goal directly to an action and thus avoid the compounding errors of the rollouts that world models optimize through. They also suffer little on unseen maps since, unlike world-model approaches, they do not have to generate a future they did not see at train time. Both JEPath and Blind-GCIQL lose nothing on unseen maps. Blind-GCIQL, which sees no frame, gains 1.1 and 4.6 points SR from the seen to the unseen pairs, so the unseen pairs are a slightly easier draw, the two sets being sampled independently and not matched in difficulty.

Tab. 2 first shows that TS collapses in first person, from 45.4% SR on ADoom to 8.4% on PDoom. Part of this is its image cost, which aliasing breaks (Sec. 3.2): DINO-WM, on the same cost, loses 7.8 points SR. We attribute the rest to the straightening prior, which assumes consecutive latent displacements keep one direction whatever the action. This nearly holds from above, since a step slides the dot linearly and a turn in place moves nothing, but in first person a turn swings the whole frame while a step only shifts it, so the latent changes direction at every turn. A frame-wise encoder can straighten such a trajectory only by mapping frames a turn apart to nearly the same latent, which merges more poses under one latent and worsens the image cost's aliasing. Second, JEPath and DINO-WM$_{\text{prop}}$, whose cost has a spatial term, gain 22.6 and 6.9 points SR in first-person view. A spatial cost needs the motion in the body frame, hence the heading. The first-person camera points where the agent faces, so the frame shows what is in its way, and every step or turn changes all of it. The top-down view shows where the agent is, but not how it moves. A step slides the dot by a few pixels, within one encoder patch, and nothing shows its heading. JEPath reads its motion from the

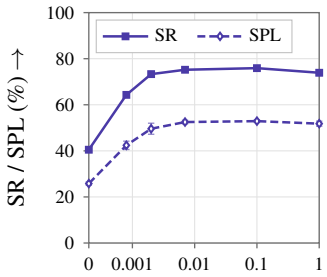
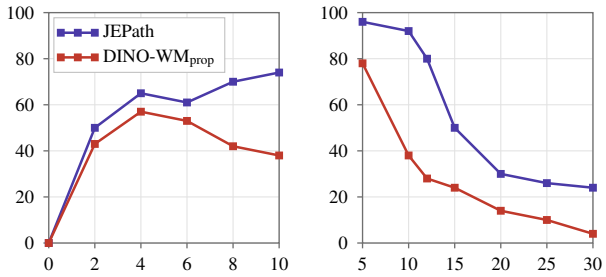



Figure 4: **Balancing planning costs.** SR and SPL of JEPath against cost weight $\omega$ of Eq. (2), from image-only ($\omega = 0$) to spatial-only cost ($\omega = 1$).




(a) Planning horizon (b) Target horizon

Figure 5: **Distance to target performance.** SR of JEPath (violet) and DINO-WM$_{\text{prop}}$ (red) against (a) the planning horizon and (b) the distance to the target.

frames alone and pays the full price, while DINO-WM$_{\text{prop}}$ is spared by its input pose. One explanation for this gap is that the motion in ADoom is subtle, which may make it harder for the model to capture. Karmann et al. (2026) show the difficulty of capturing dynamics with JEPA methods.

### 4.3 Analysis

In this section, we test JEPath on two axes. First, Fig. 4 shows how visual information can improve planning performance. Second, Fig. 5 compares JEPath with DINO-WM$_{\text{prop}}$ when the planning horizon is shorter than the target horizon (Fig. 5a) and when both horizons are extended (Fig. 5b).

Fig. 4 measures the impact of $\omega$ on planning performance, from the image cost alone ($\omega = 0$) to the spatial cost alone ($\omega = 1$). JEPath can plan with the image cost alone, achieving an SR of 40.5%, while the spatial cost adds further signal, the combination peaking at 75.9% for $\omega = 10^{-1}$. When getting closer to the target is not enough (e.g., behind a thin wall), the visual cost adds a penalty repelling from the wrong state. Performance stays stable over a large range of $\omega$ values, every value from $2 \times 10^{-3}$ to 1 staying within 2.6 points of that peak, so JEPath is robust to this hyperparameter.

Fig. 5 varies the planning horizon and the distance to the goal, each on its own set of 100 evaluation pairs. For a goal 10 actions away, Fig. 5a isolates the planning horizon. JEPath improves as the horizon grows, from 50.0% SR at horizon 2 to 74.0% at 10. Longer rollouts give the planner longer candidate paths, so more of them reach the goal instead of being stuck in a local minimum, such as behind a wall. Instead, DINO-WM$_{\text{prop}}$ peaks at horizon 4 (57.0%) and then falls back to 38.0%. Fig. 5b varies the distance to the goal from 5 to 30 actions. JEPath loses little between 5 and 10 actions (96.0% to 92.0%), whereas DINO-WM$_{\text{prop}}$ already gives up half of its success rate (78.0% to 38.0%). Both models then degrade as the target recedes, which we attribute to two factors: compounding errors, which more training should reduce, and the fact that a distant target lies in parts of the map the agent has not observed yet, so a long plan is committed to from the starting view alone.

## 5 Conclusion

World models paired with test-time trajectory optimization are a promising route to general-purpose planning, solving a wide range of tasks. However, we have shown that the planning performance of this recipe, developed under full observability, drops in first person. The main issue is perceptual aliasing, as an image alone tells the model neither where it stands nor where the goal lies. We measured this in PADoom, a new benchmark that renders the same trajectories from above and in first-person view so that observability is the only variable. We also introduced JEPath, a JEPA world model that predicts the future features and the ego-motion between frames, and integrates that motion into the path each candidate follows. Scoring a candidate on the position it reaches rather than on how it looks resolves the aliasing by design, as two places that look alike still differ in position. On PADoom, JEPath outperforms goal-conditioned policies and recent world models. In future work, we will investigate the combination of CEM planning with reinforcement learning to address planning problems over much longer horizons.

## Ethics Statement

This work involves no human subjects or personal data: all data are generated in simulation, in ViZDoom (Kempka et al., 2016), with a navigation-only action set.

## Reproducibility Statement

PADoom and its evaluation pairs are specified in Sec. 4.1 and App. C, JEPath and its planner in Sec. 3.3 and App. B, the baselines in App. D and the metrics in App. C.3. Code, data and checkpoints will be released upon publication.

## AI Use Statement

Generative AI (Claude) was used to implement code, give feedback on the experimental methodology, help analyze results, and draft and edit the text, figure code and references. It was not used to generate data or to formulate mathematical claims; the other required-disclosure tasks do not apply. The authors reviewed all AI-assisted work and take responsibility for the content of this paper.

## References

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos J. Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/6bdde0373d53d4a501249547084bed43-Abstract-Conference.html.

Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir R. Zamir. On evaluation of embodied navigation agents. *CoRR*, abs/1807.06757, 2018. URL http://arxiv.org/abs/1807.06757.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew J. Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *CoRR*, abs/2506.09985, 2025. URL https://doi.org/10.48550/arXiv.2506.09985.

Maulana Bisyir Azhari and David Hyunchul Shim. DINO-VO: A feature-based visual odometry leveraging a visual foundation model. *IEEE Robotics Autom. Lett.*, 10(9):9152–9159, 2025. URL https://doi.org/10.1109/LRA.2025.3592065.

Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 15791–15801. Computer Vision Foundation / IEEE, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Bar_Navigation_World_Models_CVPR_2025_paper.html.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.*, 47:253–279, 2013. URL https://doi.org/10.1613/jair.3912.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *Int. J. Robotics Res.*, 44(10-11):1684–1704, 2025. URL https://doi.org/10.1177/02783649241273668.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 4759–4770, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/3de568f8597b94bda53149c7d7f5958c-Abstract.html.

Samyak Datta, Oleksandr Maksymets, Judy Hoffman, Stefan Lee, Dhruv Batra, and Devi Parikh. Integrating egocentric localization for more realistic point-goal navigation agents. In Jens Kober, Fabio Ramos, and Claire J. Tomlin (eds.), *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*, volume 155 of *Proceedings of Machine Learning Research*, pp. 313–328. PMLR, 2020. URL https://proceedings.mlr.press/v155/datta21a.html.

Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the cross-entropy method. *Ann. Oper. Res.*, 134(1):19–67, 2005. URL https://doi.org/10.1007/s10479-005-5724-z.

Jesse Farebrother and Pablo Samuel Castro. CALE: continuous arcade learning environment. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/f359c6545bea1ffc971f483469b37122-Abstract-Datasets_and_Benchmarks_Track.html.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: datasets for deep data-driven reinforcement learning. *CoRR*, abs/2004.07219, 2020. URL https://arxiv.org/abs/2004.07219.

David Ha and Jürgen Schmidhuber. World models. *CoRR*, abs/1803.10122, 2018. URL http://arxiv.org/abs/1803.10122.

Danijar Hafner, Timothy P. Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 2019. URL http://proceedings.mlr.press/v97/hafner19a.html.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy P. Lillicrap. Mastering diverse control tasks through world models. *Nat.*, 640(8059):647–653, 2025. URL https://doi.org/10.1038/s41586-025-08744-2.

Nicklas Hansen and Xiaolong Wang. Hallucination in world models is predictable and preventable. *CoRR*, abs/2606.27326, 2026. URL https://doi.org/10.48550/arXiv.2606.27326.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=Oxh5CstDJU.

Steeven Janny, Hervé Poirier, Leonid Antsfeld, Guillaume Bono, Gianluca Monaci, Boris Chidlovskii, Francesco Giuliari, Alessio Del Bue, and Christian Wolf. Reasoning in visual navigation of end-to-end trained agents: A dynamical systems approach. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 12111–12121. Computer Vision Foundation / IEEE, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Janny_Reasoning_in_Visual_Navigation_of_End-to-end_Trained_Agents_A_Dynamical_CVPR_2025_paper.html.

Markus Karmann, Shile Li, Christian Internò, Bruno Andreis, David Klindt, Randall Balestriero, Jindong Gu, Philip Torr, Qi Zhang, Peng-Tao Jiang, Hao Zhang, Bo Li, and Onay Urfalioglu. Motionjepa: Preventing temporal feature collapse by capturing visual changes in latent space, 2026. URL https://arxiv.org/abs/2609.23881.

Michal Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaskowski. Vizdoom: A doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games, CIG 2016, Santorini, Greece, September 20-23, 2016*, pp. 1–8. IEEE, 2016. URL https://doi.org/10.1109/CIG.2016.7860433.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

Yann LeCun. A path towards autonomous machine intelligence. *OpenReview preprint*, 2022.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorldModel: Stable end-to-end joint-embedding predictive architecture from pixels. *CoRR*, abs/2603.19312, 2026. URL https://doi.org/10.48550/arXiv.2603.19312.

Obsidian Level Maker Contributors. Obsidian: A community continuation of the OBLIGE level maker, 2025. URL https://github.com/GTD-Carthage/Obsidian-Content. Version 21, build 2025.12.26. Random level generator for Doom, continuing OBLIGE by Andrew Apted.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=a68SUt6zFt.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=M992mjgKzI.

Ruslan Partsey, Erik Wijmans, Naoki Yokoyama, Oles Dobosevych, Dhruv Batra, and Oleksandr Maksymets. Is mapping necessary for realistic pointgoal navigation? In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 17211–17220. IEEE, 2022. URL https://doi.org/10.1109/CVPR52688.2022.01672.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 4172–4182. IEEE, 2023. URL https://doi.org/10.1109/ICCV51070.2023.00387.

Reuven Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997. ISSN 0377-2217. URL https://www.sciencedirect.com/science/article/pii/S0377221796003852.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nat.*, 588 (7839):604–609, 2020. URL https://doi.org/10.1038/s41586-020-03051-4.

Dhruv Shah, Ajay Sridhar, Arjun Bhorkar, Noriaki Hirose, and Sergey Levine. GNM: A general navigation model to drive any robot. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pp. 7226–7233. IEEE, 2023a. URL https://doi.org/10.1109/ICRA48891.2023.10161227.

Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. ViNT: A foundation model for visual navigation. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Conference on Robot Learning, CoRL 2023, 6-9 November 2023, Atlanta, GA, USA*, volume 229 of *Proceedings of Machine Learning Research*, pp. 711–733. PMLR, 2023b. URL https://proceedings.mlr.press/v229/shah23a.html.

Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. Nomad: Goal masked diffusion policies for navigation and exploration. In *IEEE International Conference on Robotics and Automation, ICRA 2024, Yokohama, Japan, May 13-17, 2024*, pp. 63–70. IEEE, 2024. URL https://doi.org/10.1109/ICRA57147.2024.10610665.

Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4):160–163, 1991. URL https://doi.org/10.1145/122344.122377.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, and Martin A. Riedmiller. Deepmind control suite. *CoRR*, abs/1801.00690, 2018. URL http://arxiv.org/abs/1801.00690.

Basile Terver, Tsung-Yen Yang, Jean Ponce, Adrien Bardes, and Yann LeCun. What drives success in physical planning with joint-embedding predictive world models? *Trans. Mach. Learn. Res.*, 2026, 2026. URL https://openreview.net/forum?id=cHZn5Gdh8e.

Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=P8pqeEkn1H.

Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*, pp. 2043–2050. IEEE, 2017. URL https://doi.org/10.1109/ICRA.2017.7989236.

Wenshan Wang, Yaoyu Hu, and Sebastian A. Scherer. Tartanvo: A generalizable learning-based VO. In Jens Kober, Fabio Ramos, and Claire J. Tomlin (eds.), *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*, volume 155 of *Proceedings of Machine Learning Research*, pp. 1761–1772. PMLR, 2020. URL https://proceedings.mlr.press/v155/wang21h.html.

Ying Wang, Oumayma Bounou, Gaoyue Zhou, Randall Balestriero, Tim G. J. Rudner, Yann LeCun, and Mengye Ren. Temporal straightening for latent planning. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Ik1mKtUYlZ.

Steven D. Whitehead and Dana H. Ballard. Learning to perceive and act by trial and error. *Mach. Learn.*, 7:45–83, 1991. URL https://doi.org/10.1007/BF00058926.

Mingkun Zhang, Wangtian Shen, Fan Zhang, Haijian Qin, Zihao Pei, and Ziyang Meng. RAE-NWM: navigation world model in dense visual representation space. In *European Conference on Computer Vision (ECCV)*, 2026. URL https://doi.org/10.48550/arXiv.2603.09241.

Xiaoming Zhao, Harsh Agrawal, Dhruv Batra, and Alexander G. Schwing. The surprising effectiveness of visual odometry techniques for embodied pointgoal navigation. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pp. 16107–16116. IEEE, 2021. URL https://doi.org/10.1109/ICCV48922.2021.01582.

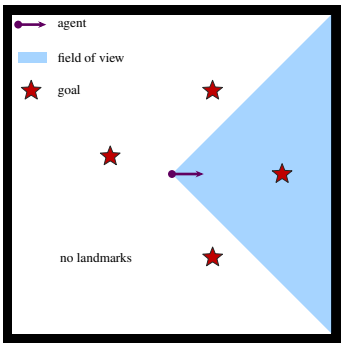
Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: world models on pre-trained visual features enable zero-shot planning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/zhou25t.html.

# Supplementary Material

## A Pathological cases

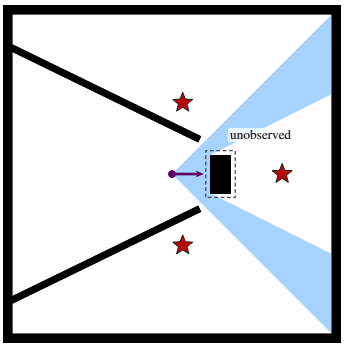
The three cases of Sec. 3.2 each isolate one situation that breaks the planner described in Sec. 3.1. For each case, this appendix gives the map and what varies, why the planner fails, and the measured success rates, which Fig. 6 gathers with the three maps. Every rate is over 10 planner seeds per goal, at the 25-unit reach radius of the evaluation. In the first two cases no world model is learned. The planner rolls its candidates out in a copy of the game engine and executes the chosen actions in the real map, so what varies is the cost, the view it reads, or the content of that copy; the third case compares two learned world models.

**(a) Perceptual aliasing**
image vs. position cost



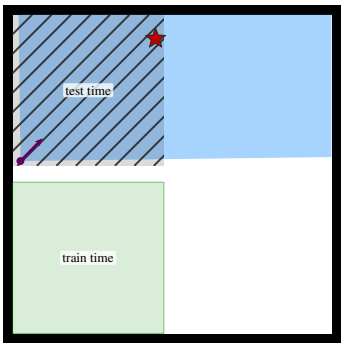

| | Observation | |
|---|---|---|
| **Cost** | top-down | first-person |
| $J_{\text{img}}$ | 100.0% | 25.0% |
| $J_{\text{pos}}$ | 100.0% | 100.0% |

**(b) Unobserved obstacle**
unobserved vs. observed wall




| | Wall observed | |
|---|---|---|
| **Cost** | no | yes |
| $J_{\text{pos}}$ | 26.7% | 96.7% |

**(c) Unvisited coordinates**
absolute vs. relative pose




| | Pose representation | |
|---|---|---|
| **Cost** | absolute | relative |
| $J_{\text{pos}}$ | 0.0% | 100.0% |

Figure 6: **The three pathological planning cases.** Each column: the map from above, walkable floor in white, and the planner's success rate on it; the shaded cell is the configuration the case is built to break.

### A.1 Perceptual aliasing

**Setup.** The map is an empty square room with no landmark. The agent starts at its center facing east, and four goals lie to its east, north-east, south-east and west, the last one behind it. The planner scores its candidates with the image cost, read either from the top-down view (ADoom) or from the first-person view (PDoom), or with the spatial cost, which reads no image.

**Why it fails.** The image cost is a good guide only if frames that look alike come from nearby poses. From above, the frame shows where the agent is, and the cost forms a single basin at the goal that any descent reaches. From the first person, poses that see a similar stretch of floor and wall render near-identical frames and score alike wherever they are, so the cost is pocked with low regions far from the goal, as deep as the true one. The planner descends into one of them and cannot tell it from the goal, since the image is all it scores, and this holds for every goal direction and both room sizes, so it is a property of the cost and not of the search.

**Result.** With the image cost read from above, the planner reaches every goal on every seed, 40/40. Read from the first person, the same cost reaches 10/40, and the goal behind the agent is never reached, no seed ever turning around. The spatial cost reaches 40/40 whichever view is rendered, and the planner, its copy and the goals are unchanged, so the failure lies in the cost.

### A.2 UNOBSERVED OBSTACLE

**Setup.** The wall map puts a block straight ahead of the agent, which starts at the center facing east, and two tilted bars that funnel toward it, all three half walls lower than the agent's eye. Three goals lie behind them, north behind the upper bar, south behind the lower bar, and east behind the block. The planner scores its candidates with the spatial cost, the exact distance to the goal, and the only variable is its copy, which either contains the walls or has them deleted; the chosen actions are always executed in the real map, walls included.

**Why it fails.** The spatial cost points straight at the goal, through any wall in the way. A copy without the walls therefore ranks the straight line first, and executed in the real map that plan drives the agent into a wall; the next plan, made from where the agent stopped, points at the wall again, so the agent stays pinned. A copy with the walls stops those candidates short of the goal and the ones that go around rank first. The two bars are tilted perpendicular to the agent's approach, so that sliding along them cannot carry it past by accident, while the block is axis-aligned, and an agent pressed against its face slides along it and can round the corner without planning to.

**Result.** With the walls in the copy, the planner reaches 29/30 episodes, 10/10, 9/10 and 10/10 on the north, south and east goals. With the walls deleted, it reaches 8/30, that is 2/10, 1/10 and 5/10, Fisher exact $p = 1.5 \times 10^{-8}$ against the copy with the walls. Three of the five successes on the east goal come from sliding along the block, and on the two bar goals, where no slide helps, the gap is 3/20 against 19/20. The cost and the search are exact in both rows, so the gap is the model's, since a planner avoids only the obstacles its model contains, and with no map given a learned model can take them only from what it sees.

### A.3 UNVISITED COORDINATES

**Setup.** The map is the empty room again, with a region covering most of one quadrant that no training trajectory enters. Two world models are trained on these trajectories with the same recipe and differ only in how they treat the pose, one taking its absolute pose as an input and predicting the next one, the other being JEPath, which takes no pose and predicts the displacement of each step, integrated outside the model. Both plan with the spatial cost toward a goal inside the region.

**Why it fails.** The absolute model was never given a pose target inside the region, so what it predicts there is unconstrained, and since the true pose is unavailable inside the horizon, it feeds its own estimate back and the error compounds. A displacement is the same quantity wherever the agent is, so JEPath needs no training trajectory inside the region to predict it there, and the integration carries the agent into coordinates the model has never seen.

**Result.** Planning toward a goal inside the region, the absolute model reaches it with none of the 10 planner seeds and JEPath with all of them.

## B JEPATH IMPLEMENTATION DETAILS

**Architecture.** A frozen DINOv2 ViT-S/14 with registers encodes every frame into a `[CLS]` token and a grid of patch tokens, the latents the model is trained to predict. A causal ViT predictor, conditioned on the one-hot action by AdaLN, reads the tokens of the $K$ context frames and returns one predicted token per input token. The latent head $g_\nu$ reads every output token, since the next `[CLS]` token belongs to the target $z_{t+1}$, while the motion head $g_\eta$, a single linear layer, reads the predicted `[CLS]` token alone. Fig. 7 redraws the model of Fig. 3 at that granularity, where the `[CLS]` token sits in the sequence, which output tokens each head reads, and where the two loss terms attach.

**Ego-motion target.** Rather than the absolute pose, JEPath regresses an encoded form of the body-frame motion of Eq. (3), its translation scaled per channel and its heading change as a point on the unit circle:

$$m'_t = \big( m^q_t/\sigma,\ \sin m^\theta_t,\ \cos m^\theta_t \big), \tag{5}$$

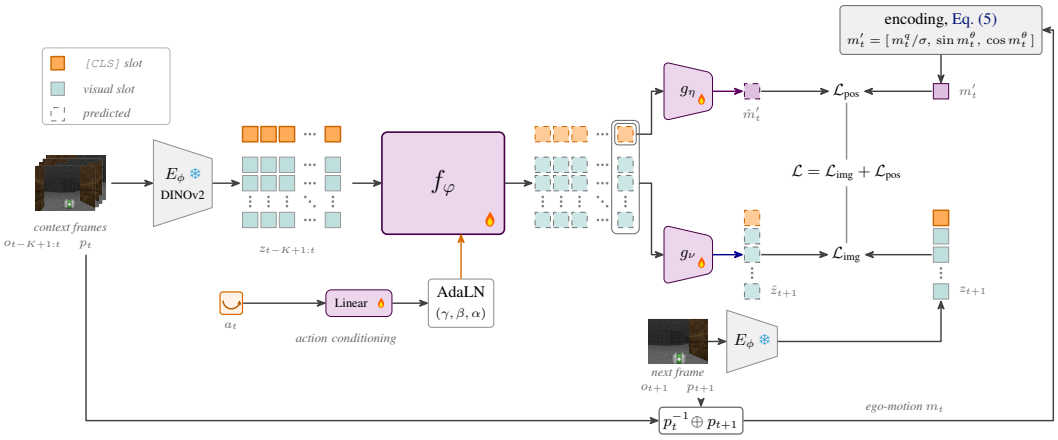



Figure 7: **Token-level view of the JEPath architecture.** The same model as Fig. 3, drawn at the granularity of the token sequence. Each context frame contributes $N$ visual tokens (teal) and DINOv2's [CLS] token (orange), all from the frozen encoder; the predictor $f_\varphi$ mixes them under AdaLN action conditioning and emits one output token per input token (dashed, in the color of its slot), so each context frame yields a prediction of the frame after it. Both heads read the last frame's outputs, the prediction of frame $t+1$: $g_\eta$ its predicted [CLS] token into the encoded ego-motion $\hat{m}'_t$ (Eq. (5)), and $g_\nu$ all its tokens into the next latent $\hat{z}_{t+1}$, each scored by a smooth-$\ell_1$ term against its target. In training, every frame's outputs are scored in the same way against the frame that follows it.

where the division is componentwise and $\sigma \in \mathbb{R}^2$ the per-channel standard deviation of the per-step body-frame displacement on the training split, with no mean subtraction so that zero keeps meaning "no movement." The scale is multiplied back when the ego-motions are integrated into a path, and no map extent enters: the normalization is set by the dynamics, not the map.

**Configuration and training.** Each 320×240 frame is center-cropped to 308×224 and encoded by the frozen DINOv2 (22.1M parameters) into 352 patch tokens, one per 14×14 patch. The predictor is a 6-layer transformer of width 1024 (8 heads), and the motion head is a single linear layer from the predicted [CLS] token to the 4 ego-motion channels; together they hold 114.2M trainable parameters. It is trained on the objective of Sec. 3.3 for 117k steps using AdamW (learning rate $5 \times 10^{-4}$, weight decay 0.05) with a learning rate that warms up, stays constant, then decays, and 16 clips per GPU on four GPUs, a batch of 64. Tab. 3 lists every training setting.

**Pose integration.** The motion head outputs $\hat{m}'_t = (\hat{u}_t, \hat{s}_t, \hat{c}_t)$, its prediction of the encoded ego-motion $m'_t$ of Eq. (5), one component per term. Inverting Eq. (5) decodes it into a body-frame motion,

$$\hat{m}^q_t = \sigma \odot \hat{u}_t, \qquad \hat{m}^\theta_t = \operatorname{atan2}(\hat{s}_t, \hat{c}_t), \tag{6}$$

with $\odot$ the componentwise product, and the step of Sec. 3.3 chains these motions over the imagined poses $\hat{p}_t = (\hat{q}_t, \hat{\theta}_t)$ for $t = 0, \ldots, T_h - 1$,

$$\hat{\theta}_{t+1} = \hat{\theta}_t + \hat{m}^\theta_t, \qquad \hat{q}_{t+1} = \hat{q}_t + R(\hat{\theta}_t)\, \hat{m}^q_t. \tag{7}$$

The atan2 ignores the radius of $(\hat{s}_t, \hat{c}_t)$, so a prediction short of the unit circle still gives the right angle, while each step is rotated by the accumulated predicted heading, so a heading error propagates into every later position, as in any open-loop rollout.

**Discrete CEM.** The $K = 4$ most recent observations are encoded once and rolled out under $n = 150$ candidate sequences of $T_h = 10$ actions, $\{a^i_{0:T_h-1}\}_{i=1}^n$. Each rollout is integrated with Eq. (7) from the current pose $p_0$, which is observed but never fed to the model, and scored by the spatial cost $J_{\text{pos}}$ of Eq. (2), in its decoded-pose form, on its final position $\hat{q}^i_{T_h}$; the heading is not scored. The action set being discrete, the sampling distribution is a table of per-step categorical logits rather than a Gaussian, initialized uniform. Candidates are drawn from the temperature-scaled logits, and the greedy sequence under the current logits is always among them, so the mode of the distribution is evaluated at every iteration. The logits are then refit on the $m = 30$ elites, the best

Table 3: **JEPath training hyperparameters.** Settings of the checkpoint evaluated in Tab. 1. Lengths are in Doom world units.

| | |
|---|---|
| *Encoder* | |
| Backbone | DINOv2 ViT-S/14 with 4 registers, frozen (22.1M) |
| Input | 320×240 frame, center-cropped to 308×224 |
| Tokens per frame | 1 `[CLS]` + 16×22 patches, dimension 384 |
| *Predictor* | |
| Architecture | ViT, 6 blocks, width 1024, 8 heads, MLP ratio 4 |
| Attention mask | block-causal over frames |
| Action conditioning | AdaLN-Zero on the one-hot action, 19 actions |
| Motion head | linear, predicted `[CLS]` $\rightarrow$ 4 channels |
| Trainable parameters | 114.2M |
| *Objective* | |
| Latent loss $\mathcal{L}_{\text{img}}$ | smooth-$\ell_1$ ($\beta = 1$) |
| Motion loss $\mathcal{L}_{\text{pos}}$ | smooth-$\ell_1$ ($\beta = 1$) |
| Regime | teacher forcing |
| *Data* | |
| Clip length $T_c$ | 10 frames, one random window per episode per epoch |
| Frames per action | 1, engine frame skip 4 |
| *Optimization* | |
| Optimizer | AdamW, $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$ |
| Learning rate | $5 \times 10^{-4}$ |
| Weight decay | 0.05, on all parameters |
| Schedule | 250 linear warmup steps, constant until step 102.8k, then linear decay toward $5 \times 10^{-6}$ over 15k steps |
| Batch size | 64 clips, 16 per GPU on 4 H100 |
| Training steps | 116,992 (≈256 epochs) |
| Checkpoint | fixed step, not selected on validation |
| Gradient clipping | norm 1.0 |
| Precision | bf16 mixed |

candidates of the iteration, as the log of their per-step action frequencies with additive smoothing, the standard update of the discrete CEM (de Boer et al., 2005). After a fixed number of iterations the best sequence found over all iterations is executed to its end. The planner then reads the $K$ latest observations and the new pose and starts again. Tab. 4 lists the planner and evaluation settings.

# C PADOOM BENCHMARK

## C.1 TRAINING TRAJECTORIES

**Collection.** Every training trajectory is a shortest path between two random points of a map, followed in the engine. A start pose and a goal are sampled on the map's walkable mask, the A* shortest path between them is planned on that mask, and a pure-pursuit controller follows it closed-loop in ViZDoom, one recorded frame per action at a frame skip of 4, over the 19-action navigation table, which includes the no-op. The engines run on CPU as independent processes that share no state, so collection scales with the number of processes.

**Action coverage.** Followed greedily, such paths would collapse the action distribution onto forward motion, so the controller runs in an exploration mode. The heading is decoupled from the direction of travel and roams on its own. At every step the action is sampled among the primitives that track the path to within a tolerance, weighted toward the actions used least so far and away from moves into walls, with short bursts of the no-op injected so that its dynamics are in the data. The planned path itself can be pushed away from walls and routed through random waypoints.

**Volume and budget.** Each episode records the frame after every action, the pose before it and the action, and episodes with fewer than 11 frames are dropped. 20000 trajectories are collected per map, of which 15456 to 20000 survive the pruning; on MAP001 they use all 19 actions at a normalized

Table 4: **JEPath planning and evaluation hyperparameters.** Settings of every JEPath cell of Tab. 1. Lengths are in Doom world units.

| | |
|---|---|
| *Discrete CEM* | |
| Candidates per iteration $n$ | 150, including the greedy sequence |
| Elites $m$ | 30 |
| Iterations | 5 |
| Sampling temperature | 2 |
| Additive smoothing | 1 |
| Logit initialization | uniform, reset at every replan |
| Executed sequence | best over all iterations |
| *Rollout and cost* | |
| Context frames $K$ | 4 |
| Horizon $T_h$ | 10 actions |
| Executed actions per replan $T_e$ | 10 |
| Cost | $J_{\text{pos}}$ of Eq. (2) ($\omega = 1$) on the final dreamed position, heading unscored |
| Pose anchor | true pose, read at every replan |
| Precision | fp32 |
| *Episode and scoring* | |
| Action space | 19 discrete actions, engine frame skip 4 |
| Action budget | 60 actions (6 replans) |
| Initial context | replay of the recorded approach to the start pose |
| Success | any executed pose within $\delta = 25$ of the goal |
| Metrics | SR and SPL (Eq. (8)) |
| Pairs, Normal / Hard | 499 / 490 seen, 500 / 500 unseen |
| Planner seeds | 42, 43, 44, mean $\pm$ standard deviation |

action entropy of 0.91 and visit 77% of the walkable cells, and MAP021's collection stopped at 15515. Every model of Tab. 1 is trained on a budget of 29250 episodes split equally across the 50 training maps (585 per map, 29250 episodes).

### C.2 Evaluation pairs

**Source trajectories.** Evaluation pairs come from separate collections on the same maps, made with the greedy follower and no exploration noise, between points whose straight line is blocked. The Normal sets draw 20 trajectories per map between points at least 200 units apart, the Hard sets a larger collection of 200 per map between points at least 1200 units apart.

**Window selection.** From each trajectory one window of $k = 10$ actions is cut, its start at least 4 actions into the trajectory so that a real approach precedes it. A window is kept only if the straight segment from its start to its goal is blocked for the 32-unit player box, its geodesic is at least 1.05 times its chord, and the chord at least 150 units. Of a trajectory's passing windows the most detoured one is kept, up to 10 pairs per map.

**Normal and Hard sets.** On the first collection this gives the Normal sets, 499 pairs on the 50 training maps and 500 on the 50 held-out maps. On the second, the Hard sets add two tests run in the engine from the start pose, with the 60-action budget and 25-unit reach radius of the evaluation. A window is dropped if a uniformly random policy comes within 25 units of the goal at any of the evaluation seeds 0, 1, 2, 42, 43 and 44, and dropped if a goal-bearing controller that turns toward the goal and drives does so; a failure whose closest approach still has a clear line to the goal is discarded as well, as a controller artifact rather than an obstacle. Both controllers therefore score exactly 0 on the Hard sets by construction, at that radius and budget, the random one only at those seeds. The Hard sets hold 490 pairs on 49 training maps, MAP021 being excluded since it yields too few qualifying pairs (3, even after pooling three collection seeds), and 500 on the 50 held-out maps. Their chords are slightly longer than the Normal sets', consistent with the filters removing the short windows such a controller stumbles into. Fig. 8 draws every pair of the in-domain Normal set on its map's walkable mask, each path being the executed $k$-action window from start to goal.

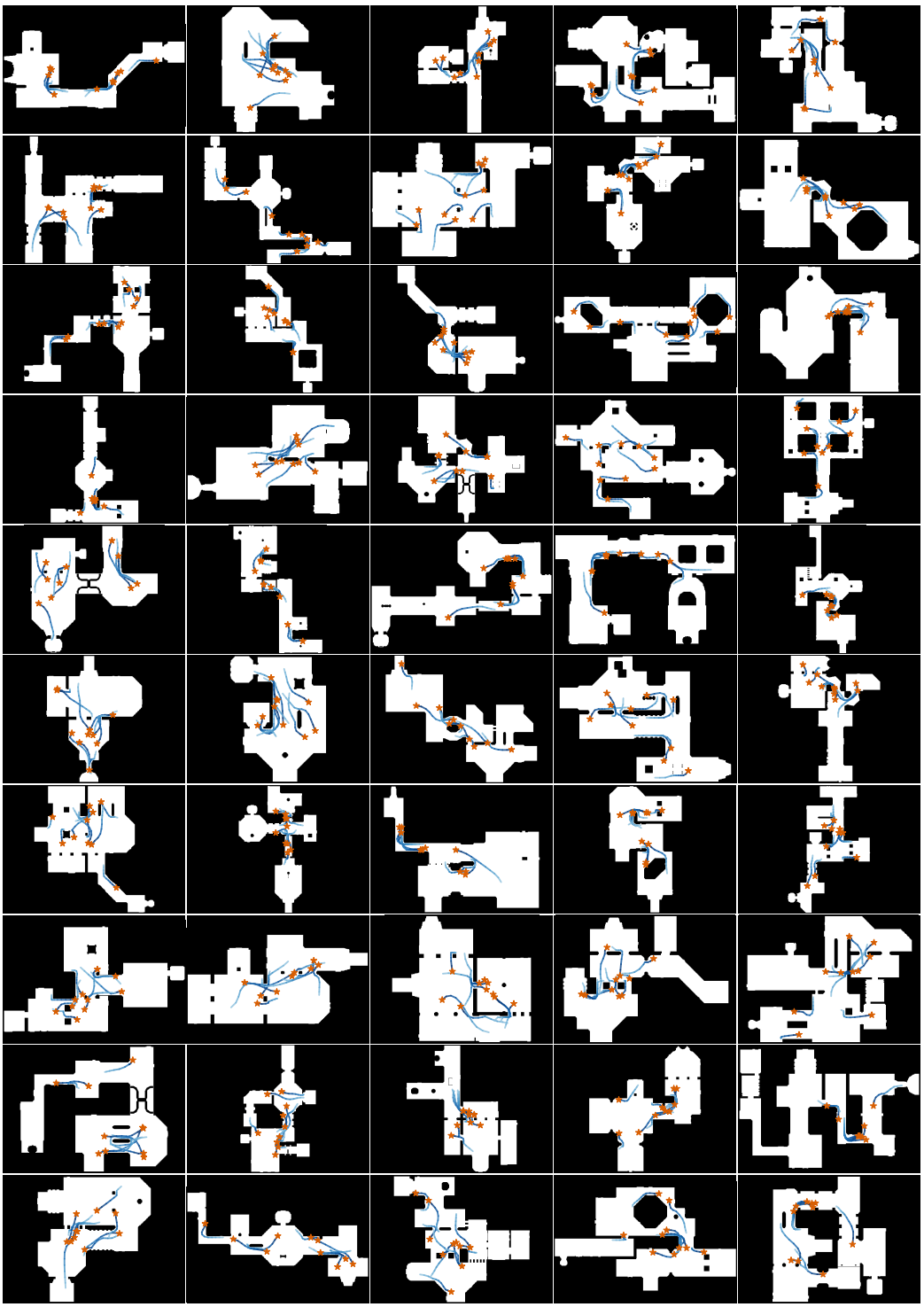


Figure 8: **In-domain evaluation trajectories (Normal set).** Each panel is one map's walkable mask (black/white, ADoom convention); each line is the *executed* path of one evaluation pair, running pale at the start pose to dark at the goal, where an orange star marks the goal. Panels are independently scaled and not comparable in size. 499 pairs over 50 maps; MAP037 has 9.

### C.3 Evaluation metrics

An episode is a success once the agent comes within a reach radius $\delta$ of the goal position, $\|q_t - q_g\|_2 \leq \delta$ (Sec. 3.1). The success rate, noted SR@$\delta$, is the share of successful episodes when the agent reached the target within the reach radius $\delta$. In our experiments, we use $\delta = 25$ world units unless stated otherwise. Success weighted by path length (Anderson et al., 2018) discounts each success by how far the agent strayed from the shortest route,

$$\mathrm{SPL} = \frac{1}{N} \sum_{i=1}^{N} S_i \frac{\ell_i}{\max(p_i, \ell_i)}, \tag{8}$$

with $S_i$ the success of episode $i$, $\ell_i$ the geodesic distance from its start to its goal and $p_i$ the length of the path it executed. A success reached along the shortest path contributes 1 and a wandering one less, so SPL never exceeds SR and the gap between them measures detour rather than failure.

## D Baseline details

Every baseline is trained on the same PADoom trajectories as JEPath and evaluated on the same pairs. For each, this appendix gives its role in the comparison, its architecture and objective, what was changed from the upstream code, and its training settings. Expectations are over the offline dataset $\mathcal{D}$, sg denotes a stopped gradient, and $\mathrm{smooth}_{\ell_1}$ the Huber loss of the training objective of Sec. 3.3. Every world model is planned with the loop of Sec. 3.1 under the cost of Eq. (2), and they differ only in $\omega$: DINO-WM, NWM and RAE-NWM score the image term alone ($\omega = 0$), the prop arms weight the image and spatial terms equally ($\omega = 1/2$), and JEPath scores the spatial term alone ($\omega = 1$).

### D.1 Model-free policies

**Goal-conditioned policies (GCBC, GCIQL, Blind-GCIQL).** The three policies measure what the goal coordinates buy without any model: GCBC imitates the recorded actions, GCIQL learns a value and improves on them, and Blind-GCIQL, fed black frames, isolates what the coordinates alone achieve. They share one architecture: a four-layer convolutional encoder maps the $48 \times 64$ frame to a 256-d feature, which is concatenated with the 3-d polar goal vector and read by a width-512 MLP with one output per action. GCBC is that network alone, a policy $\pi(a \mid o, g)$ over the 19 discrete actions given the frame $o$ and the goal $g$, trained by cross-entropy:

$$\mathcal{L}_{\mathrm{GCBC}} = \mathbb{E}_{\mathcal{D}}\big[ -\log \pi(a \mid o, g) \big]. \tag{9}$$

GCIQL (Kostrikov et al., 2022) adds twin critics $Q_{1,2}$ and a value $V$ of the same shape, without weight sharing and no target value network, the critics bootstrapping off $V$ with a stopped gradient, and sums the expectile loss of the value, the TD loss of both critics and the advantage-weighted log-likelihood of the policy:

$$\mathcal{L}_{\mathrm{GCIQL}} = \mathbb{E}_{\mathcal{D}}\Big[ L_2^\tau\big(\hat{A}\big) + \sum_{i=1,2} \big(Q_i - r - \gamma(1-d)\,\mathrm{sg}(V')\big)^2 - \min\big(e^{\beta \hat{A}},\, 100\big) \log \pi(a \mid o, g) \Big], \tag{10}$$

with $L_2^\tau(u) = |\tau - \mathbf{1}\{u < 0\}|\, u^2$, $\hat{A} = \mathrm{sg}\big(\min_i \bar{Q}_i - V\big)$ the advantage under the twin target critics, $\tau = 0.9$, $\gamma = 0.97$, $\beta = 3$, $r$ the normalized progress toward the goal, $V'$ the value of the next state and $d$ its end-of-episode flag; the targets are Polyak copies of the critics (rate 0.005), and the frames get DrQ random shifts. Blind-GCIQL is GCIQL with every input frame replaced by black, all-zero pixels; the architecture, the objective and the data are unchanged. Both train for 60k steps at batch 256, one Adam per network at learning rate $3 \times 10^{-4}$.

### D.2 World models

**Absolute pose targets.** The prop arms below, and the absolute-pose row of Tab. 5, regress the GPS-style absolute position rather than a motion, normalized per axis onto $[-1, 1]$, each coordinate $x$ of $q_t = (x, y)$ as $x_{\mathrm{norm}} = (2x - (x_{\max} + x_{\min}))/(x_{\max} - x_{\min})$, with the extremes measured on the training split; heading enters as $(\sin\theta, \cos\theta)$, already bounded. On a bounded map the walls make $x_{\min}$ and $x_{\max}$ hard geometric limits, so the normalization depends only on the map, not on where the data-collection policy happened to walk.

**DINO-WM / DINO-WM$_{\text{prop}}$ (Zhou et al., 2025).** DINO-WM is the image-goal recipe of Sec. 3.1 as published. JEPath departs from it on two axes: the pose enters as an input token rather than being predicted, and the planner scores the image ($\omega = 0$); the prop variant keeps the pose input and adds the spatial term to the cost ($\omega = 1/2$). The official implementation is trained on our data at its released size, which is why its predictor is a fifth of JEPath rather than a match for it. A frozen DINOv2 ViT-S/14 maps each frame to patch tokens, a proprioceptive embedding $h$ and an action embedding $e$ are concatenated to them channel-wise, written $[\,\cdot\,;\,\cdot\,]$ into the token $Z_t$, and a causal ViT (depth 6, 16 heads, MLP width 2048) predicts the next token from a three-frame window; the decoder is disabled. The prediction is regressed onto the detached next-frame token on its visual and proprioceptive channels, the action channels being predicted but not scored:

$$\mathcal{L}_{\text{DINO-WM}} = \mathbb{E}_{\mathcal{D}} \big\| \hat{Z}_{t+1} - \text{sg}(Z_{t+1}) \big\|_2^2, \qquad Z_t = \big[\, E_\phi(o_t)\,;\, h(p_t)\,;\, e(a_t) \,\big]. \tag{11}$$

DINO-WM drops the proprioceptive channels, the only difference between the two variants, which leaves their size unchanged to three digits. Both train for 234k steps at batch 32 with AdamW at learning rate $5 \times 10^{-4}$.

**JEPA-WM$_{\text{prop}}$ (Terver et al., 2026).** The predictor of JEPath with the pose fed back in, and thus the closest ablation of the *"reads no pose"* decision of Sec. 3.3. A trainable linear map $W$ encodes the pose $(x, y, \sin\theta, \cos\theta)_t$ into an input token, and a latent proprioceptive head, a $\text{Linear}(1024, 1024)$, regresses the detached embedding of the next pose, its loss added to the latent one:

$$\mathcal{L}_{\text{JEPA-WM}_{\text{prop}}} = \big\| \hat{z}_{t+1} - z_{t+1} \big\|_2^2 + \lambda \big\| \hat{\zeta}_{t+1} - \text{sg}\big(\zeta(p_{t+1})\big) \big\|_2^2, \qquad \zeta(p) = Wp, \quad \lambda = 1. \tag{12}$$

**Navigation World Model (NWM, Bar et al., 2025).** NWM and RAE-NWM are the two diffusion world models built for first-person navigation, planned here with the image cost. NWM's official implementation is trained on our data. A frozen Stable Diffusion VAE encodes each 224×224 frame into a 4×28×28 latent, and a CDiT-M/2 backbone (depth 12, width 512, 8 heads, the size closest to JEPath in trainable parameters) denoises the latent of one future frame, conditioned by cross-attention on the four most recent frames. Two things change from upstream: the action conditioning, a continuous relative goal pose there, is here the embedding of PADoom's 19 discrete actions; and the model is trained to predict the immediately next frame rather than one $k$ steps ahead with $k$ drawn over a horizon, since that is how the planner rolls it out, one action per forward pass. Training runs 117k steps at batch 128 with the upstream optimizer at learning rate $8 \times 10^{-5}$, and each rollout step is one sampling call of 25 denoising steps, kept low for compute.

**RAE-NWM (Zhang et al., 2026).** NWM with its tokenizer replaced: a frozen DINOv2-B with registers takes the place of the VAE and maps each frame to a 768×16×16 latent, the CDiT-M/2 backbone is kept, and a stage-2 denoising head (width 2048, depth 2) reads its output and holds three fifths of the trainable parameters on its own. The discrete action embedding and the next-frame regime of NWM apply here too; training runs 117k steps at batch 64 with the upstream optimizer at learning rate $1 \times 10^{-4}$, and each rollout step is one sampling call of the 50 Euler steps of the flow-matching ODE.

**Temporal Straightening (TS, Wang et al., 2026).** TS is the other answer to aliasing: rather than changing the cost, it straightens the latent trajectory so that feature distance grows along a path, and Tab. 2 compares it with JEPath under both observation modes. The official implementation is trained on our data, three trainable parts on top of a frozen DINOv2 ViT-S/14: a convolutional projector ($384 \to 192 \to 8$ channels) that compresses the 14×14 patch tokens of a frame into the features $E_\phi(o_t)$; DINO-WM's causal ViT at token dimension 28, which predicts the next features from the current ones, the pose $p_t$ and the action $a_t$; and a three-layer MLP head $E^s$ that pools each frame into a 128-d vector $\bar{z}_t$ on which the trajectory is straightened. The predictor is an order of magnitude smaller than DINO-WM's, two thirds of the trainable parameters sitting in the projector and the pooling head. As in DINO-WM, the token $z_t = \big[\, E_\phi(o_t)\,;\, h(p_t)\,;\, e(a_t) \,\big] \in \mathbb{R}^{28}$ holds 8 visual, 10 proprioceptive and 10 action channels, and the action channels are predicted but not scored. Training adds a curvature penalty on consecutive steps of the pooled trajectory to the next-feature regression of DINO-WM:

$$\mathcal{L}_{\text{TS}} = \mathcal{L}_{\text{DINO-WM}} + \lambda\, \mathbb{E}_{\mathcal{D}} \Big[ 1 - \frac{v_t \cdot v_{t+1}}{\|v_t\|_2 \, \|v_{t+1}\|_2} \Big], \qquad v_t = \bar{z}_{t+1} - \bar{z}_t, \tag{13}$$

Table 5: **Aggregating tokens for the motion head.** SR and SPL of JEPath on the Hard seen pairs of Tab. 1, for the tokens the motion head reads. The first row regresses the absolute pose instead of the ego-motion. Bold/underline: best/second best.

| | SR | SPL |
|---|---|---|
| Absolute pose target | $63.1_{\pm 2.2}$ | $40.5_{\pm 1.3}$ |
| Learnable token | $69.3_{\pm 0.5}$ | $47.7_{\pm 0.9}$ |
| Three learnable, concatenated | $69.7_{\pm 0.4}$ | $48.1_{\pm 0.3}$ |
| Mean-pooled patches | $72.0_{\pm 0.4}$ | $51.4_{\pm 0.3}$ |
| Three learnable, one per DoF | $\underline{72.5}_{\pm 1.0}$ | $\underline{51.7}_{\pm 0.7}$ |
| `[CLS]` token (JEPath) | $\mathbf{73.9}_{\pm 1.2}$ | $\mathbf{51.8}_{\pm 1.1}$ |

with $\lambda = 0.1$, the curvature averaged over the two interior frames of each four-frame window. One checkpoint is trained per observation mode, differing only in the view and its resize, for 234k steps at batch 32, with Adam at learning rate $10^{-5}$ on its encoder and AdamW at $5 \times 10^{-4}$ elsewhere.

# E Additional results

## E.1 Aggregating tokens for the motion head

The motion head is a linear readout, so the architectural choice left open is which tokens it reads. Tab. 5 compares five aggregations on the Hard seen pairs, everything else held fixed: a learnable token prepended to every frame, three such tokens concatenated, three with one per degree of freedom, the mean of the frame's patch tokens, and DINOv2's own `[CLS]` token, which is what JEPath uses. The ordering is consistent, an aggregation reading the frame's own tokens beating any added register: mean pooling is 2.7% above a single learnable token, and the predicted `[CLS]` token is best, 4.6% above it. The first row changes the target rather than the aggregation, prepending a token that encodes the absolute ego-state and regressing the next absolute pose from it (normalized as in App. D.2), and sits 10.8 points below the `[CLS]` readout.

## E.2 Planning algorithm

JEPath plans with the CEM of App. B, scores an imagined path by the distance of its final position to the goal and executes the whole planned sequence before replanning. Replacing the CEM with a Monte Carlo tree search (MCTS) over the same 750 rollouts per plan reaches 72.2% SR against 73.9%, and 54.8 SPL against 51.8. The tree search scores a path by the smallest distance to the goal reached along it rather than at its end: a branch that rounds an obstacle first moves away from the goal, and scoring its end alone prunes it early. Therefore, this row changes both the search and the cost. Neither variant improves the SR of the CEM, and the tree search gains 3.0 points of SPL, which rewards shorter paths.

Table 6: **CEM vs. MCTS.** SR and SPL of JEPath on the Hard seen pairs of Tab. 1, mean $\pm$ std over 3 seeds. Search: the CEM of App. B, or an MCTS with the same 750 imagined rollouts per plan, expanded in batches of 25 with exploration constant 0.1, each node valued by the best cost below it.

| Search | SR | SPL |
|---|---|---|
| CEM | $\mathbf{73.9}_{\pm 1.2}$ | $51.8_{\pm 1.1}$ |
| MCTS | $72.2_{\pm 0.5}$ | $\mathbf{54.8}_{\pm 0.4}$ |