LATENT GOAL PREDICTION FROM LANGUAGE FOR MODEL-BASED PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Joint-Embedding Predictive Architectures (JEPAs) enable agents to plan in latent space by imagining the outcomes of candidate actions, yet task specification remains a bottleneck. Visual targets provide precise local gradients but poor distant guidance, while language is flexible yet limited by noisy cross-modal alignment or dependence on distinct large generative models. We introduce LAGO (Latent Goal Prediction from Language), a hierarchical world model in which a single predictor both forecasts action-conditioned dynamics and grounds language instructions as sequences of intermediate latent subgoals, training both modes with a single regression objective over a shared latent space. At each planning step, LAGO predicts a sequence of latent subgoals from a language instruction and optimizes an action sequence using a soft-minimum alignment cost that rewards subgoal proximity without enforcing a rigid path. Subgoals are repredicted as the agent's state evolves, turning a single long-horizon instruction into a sequence of locally tractable objectives. Across environments spanning navigation and manipulation, LAGO outperforms flat and hierarchical image-goal planners, with the largest gains on long-horizon navigation with curved, hazard-constrained paths, where it more than doubles the success rate of a goal-conditioned behavior cloning policy trained on the same demonstrations. Grounding language in the world model's own latent space also outperforms vision-language reward models, including one that plans with ground-truth dynamics. Code available here.

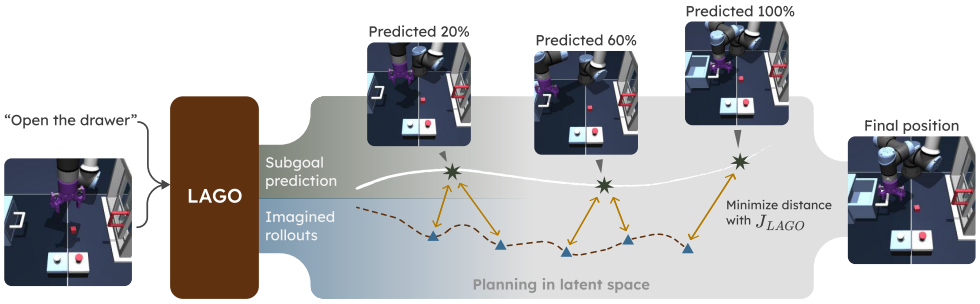


Figure 1: **Latent Goal Prediction from Language for Model-Based Planning.** LAGO trains a single latent predictor to forecast both language-conditioned subgoals (green stars) and action-conditioned state rollouts (blue triangles), unifying task specification and dynamics in a shared latent space. Given a language instruction, CEM optimizes action sequences to reach the predicted subgoals using J_{LAGO}. Re-predicting subgoals from the agent's current state at each planning step enables adaptive, long-horizon planning toward the desired final state.

1 INTRODUCTION

Achieving complex tasks from high-dimensional sensory observations remains a fundamental challenge in embodied intelligence. An agent must learn how its actions affect the world while reasoning over long horizons to reach a desired outcome. World models (Hou et al., 2026) offer a promising framework for this problem by separating these two challenges: a model can learn the dynamics of the environment and then be used to plan action sequences through imagined interactions, rather than learning a direct mapping from observations to actions. Two broad approaches have emerged for

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.

learning such models. One line of work jointly learns the environment dynamics with task-specific rewards or value functions, allowing planning to optimize a learned notion of task progress (Sutton & Barto, 2018; Hafner et al., 2020; Hansen et al., 2022). While such methods have demonstrated strong data efficiency and multitask capabilities (Hafner et al., 2025), they require additional reward supervision and are therefore tied to the tasks for which such supervision is available. A second line of work instead learns the dynamics of the environment without specifying a task at training time. In particular, Joint-Embedding Predictive Architectures (JEPAs) (LeCun, 2022) learn to predict future representations from observations and actions through a self-supervised latent-regression objective. V-JEPA 2 (Assran et al., 2025) demonstrated the potential of this paradigm for learning visual dynamics at scale, while subsequent work has leveraged such representations for trajectory optimization in robotic environments (Zhou et al., 2025; Daniel et al., 2026; Zhang et al., 2026; Maes et al., 2026b; Masip et al., 2026).

Existing JEPA-based planners formulate the desired outcome as a goal image, optimizing an action sequence such that its predicted latent state approaches the representation of the target (Zhou et al., 2025). Such a goal image provides precise supervision and a well-defined objective, but is impractical to obtain and becomes limiting for long-horizon tasks when the target lies beyond the model's reliable prediction horizon. Recent work mitigates this either by encouraging straighter, more predictable latent trajectories (Wang et al., 2026; Maes et al., 2026b), or by decomposing the horizon hierarchically, predicting intermediate subgoals or planning across multiple temporal scales (Masip et al., 2026; Zhang et al., 2026). Despite these advances, all of these methods still require a visual goal to specify the task.

Language offers a flexible alternative for specifying tasks, allowing a user to describe the desired behavior rather than provide a target observation. The challenge is then to ground this abstract instruction in the physical state of the environment. Existing approaches have explored this through language-conditioned rewards, either by training completion classifiers (Nair et al., 2021) or leveraging pretrained vision-language models to score the alignment between language and visual states or trajectories (Nair et al., 2022; Sontakke et al., 2023; Ma et al., 2023; Rocamonde et al., 2024; Wang et al., 2024). However, these cross-modal objectives can be noisy and difficult to optimize (Fu et al., 2024; Roy et al., 2025). Other methods instead predict future visual states from language, using image-editing diffusion models to generate subgoals (Black et al., 2023) or video generation models to predict trajectories and intermediate states (Du et al., 2023a; Ajay et al., 2023; Ko et al., 2023; Du et al., 2023b). These approaches rely on large generative models and expensive pixel-space sampling, while their visual predictions remain separate from the latent space in which the world model learns dynamics, preventing goals and dynamics from being expressed in a common representation.

We introduce **LAGO** (*Latent Goal prediction from language*), a hierarchical world model that plans directly from natural-language instructions. As presented in Figure 1, a single predictor operates in two modes: forecasting state dynamics conditioned on action, and grounding a language instruction as latent subgoals in the current scene. These language-conditioned subgoals give planning concrete, reachable targets, turning a long-horizon instruction into a sequence of locally tractable objectives that can be repredicted at every planning step. We summarize our contributions as follows:

- To bridge the gap between image-based and language-based goal specification, we introduce LAGO, a world model trained to map natural language instructions to target states within its own latent space, combining the precision of visual goals with the flexibility of language.
- Conditioned on language and an arbitrary completion value, LAGO is capable of predicting many subgoals for one instruction. Coupled with a soft-minimum cost function, this creates a navigable valley guiding the action-conditioned rollouts towards the objective.
- We show that planning with subgoals inferred from language approaches the performance of oracle subgoals at short and medium horizons and substantially outperforms flat and hierarchical image-goal planners. Additionally, grounding language in the world model's own latent space outperforms dedicated vision-language reward models, including when the latter plan with ground-truth dynamics.

2 Related Work

World Models. WMs were popularized in the context of model-based reinforcement learning (MBRL) (Sutton & Barto, 2018; Ha & Schmidhuber, 2018), where an agent learns a model of its

environment through which it can plan a policy before executing it. These methods learn latent dynamics from interaction, and address goal specification by learning task-specific reward and value functions from experience (Hafner et al., 2019; 2020; Hansen et al., 2022). The method proposed by Hafner et al. (2025) was the first to collect diamonds in Minecraft without human data or task-specific tuning, and recent work has scaled MBRL across a large and heterogeneous suite of environments (Hansen et al., 2025). Yet, these methods remain bottlenecked not by model capacity but by reward specification. A parallel line of work instead formulates planning as optimizing actions to minimize the distance between predicted rollouts and a goal state in a learned latent space, eliminating the need for task-specific reward and value learning. The representation is either inherited from a pretrained vision encoder (Zhou et al., 2025) or learned from pixels end to end (Assran et al., 2025).

To improve long-horizon planning, the latent space can be explicitly regularized so that distance in the space approximates planning cost (Wang et al., 2026). Maes et al. (2026b) shows that a similar straightening of latent trajectories emerges naturally from end-to-end next-state prediction training, without an explicit temporal regularizer. Long horizons have also been addressed through hierarchical planning, either by predicting intermediate subgoals conditioned on a goal image (Masip et al., 2026) or by learning world models at different temporal scales, with a high-level model generating intermediate states for a low-level planner (Zhang et al., 2026). Common to all these methods is the use of a goal image, which gives precise, well-shaped objectives near the target, but is impractical to obtain for complex tasks and offers little guidance when the goal lies beyond the planning horizon.

LAGO addresses this limitation by automatically generating latent subgoals that decompose long-horizon planning into a sequence of short-horizon prediction problems, substantially reducing the rollout lengths required to obtain meaningful optimization signals. Rather than optimizing directly toward a distant objective, LAGO *dynamically predicts intermediate latent targets* that maintain local tractability while preserving long-horizon semantic consistency.

Language as Task Specification. Vision-language-action (VLA) models map observations and instructions directly to actions by scaling behavior cloning (Brohan et al., 2023; Black et al., 2025), but they must implicitly infer the intermediate behaviors of a long-horizon instruction and scale only with action-labeled data. Reward-based methods instead ground language in a reward signal: Nair et al. (2021) trains a language-conditioned completion classifier for model-based planning, and later work repurposes pretrained VLMs as rewards via state–language embedding similarity or trajectory preference labels (Nair et al., 2022; Sontakke et al., 2023; Rocamonde et al., 2024). This removes hand-designed rewards but yields brittle, noisy objectives (Fu et al., 2024; Roy et al., 2025). A third line grounds language in future states, using image-editing or video generation models to synthesize subgoals or goal-reaching trajectories, with actions recovered by inverse dynamics (Black et al., 2023; Du et al., 2023a; Ajay et al., 2023). Because supervision is a future state, this scales to action-less video datasets orders of magnitude larger than action-labeled ones (Grauman et al., 2022; Chen et al., 2026), but tying planning to generative pixel decoding is expensive and forfeits scoring trajectories in the space where dynamics are modeled.

LAGO instead maps language instructions directly to target states *within a world model's latent space* and plans by minimizing latent distance to these targets. Because supervision is a future latent state, LAGO avoids the noisy cross-modal objectives of VLM-based rewards and the cost of decoding imagined rollouts into pixels, requiring action labels only to fit a task-agnostic world model and remaining free to scale with action-less demonstration data.

Hierarchical Planners. A growing line of work addresses long-horizon planning by decomposing tasks into intermediate subgoals that are easier to optimize and execute. An early approach by Nair & Finn (2019) trains a generative model to propose candidate subgoal images and optimizes these subgoals to minimize the worst-case planning cost across segments. However, the subgoals are decoded as full images, their number is limited by the cost of optimization, and they are computed once before execution rather than updated as the plan unfolds. Chen et al. (2024) generates coarse high-level trajectories subsequently refined into executable plans, while Hao et al. (2025) introduces coupled diffusion processes to improve temporal consistency. However, these methods plan over low-level environment states, rather than high-dimensional observations, and rely on behavior cloning to execute the resulting plans. Huang et al. (2024) uses a diffusion model to generate coarse-to-fine latent subgoal sequences that are executed with model predictive control (MPC), enabling iterative replanning rather than a fixed hierarchical decomposition. However, it assumes access to privileged geometric states, namely signed distance fields and full 3D object models, which restricts

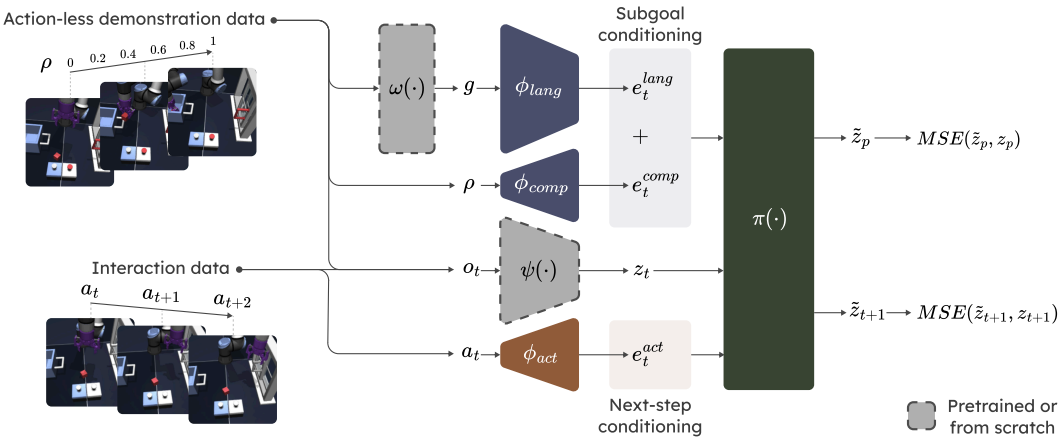


Figure 2: **The LAGO architecture.** LAGO is a Joint Embedding Predictive Architecture operating in two modes over a shared latent space. Given a current latent observation z_t, a conditioned latent predictor $\pi(\cdot)$ forecasts a target latent state: either the next state $\tilde{z}_{t+1}$ when conditioned on a low-level action, or a subgoal state $\tilde{z}_\rho$ when conditioned on a language instruction and a completion scalar ρ. Both modes share a single predictor, and the architecture is agnostic to the choice of latent space.

its application in standard visual environments. More recently, Haramati et al. (2026) explored object-centric subgoal decomposition for compositional planning, but relies on a separately trained value function to filter generated subgoals for reachability at test time. Concurrently, FF-JEPA (Masip et al., 2026) also keeps subgoal prediction in the world model's latent space, pairing an action-conditioned dynamics model with an action-free planner that proposes the next subgoal for a CEM optimizer. LAGO differs from FF-JEPA in three ways: **(i)** it uses a *single predictor* with shared parameters and a unified training objective for both dynamics and subgoal prediction, rather than relying on separate models and an explicit target image; **(ii)** it conditions each subgoal on a language instruction and a continuous completion variable ρ to predict *task-specific* intermediate states; **(iii)** it predicts a *sequence of subgoals*, jointly optimized through soft-minimum trajectory alignment and updated as the state evolves, rather than a single subgoal at each timestep.

Unlike these approaches, LAGO doesn't require reward supervision, expert action supervision, or access to low-level state information. It can predict an arbitrary number of subgoals and continuously updates its latent subgoal predictions online as the world state and language instruction evolve. Combined with our soft minimum trajectory cost formulation, which enables smooth progress evaluation without brittle discrete completion criteria, this yields more robust long-horizon planning under model uncertainty.

3 Methodology

3.1 LAGO

LAGO is a Joint Embedding Predictive Architecture (Assran et al., 2023) supporting two operating modes: **(1)** one-step transition prediction conditioned on actions, and **(2)** subgoal prediction conditioned on language instructions. Figure 2 gives an overview of LAGO's architecture.

Vision Encoder. Given an RGB observation $o_t \in \mathbb{R}^{H\times W\times 3}$, the encoder produces a sequence of patch tokens:

$$z_t = \psi(o_t) \in \mathbb{R}^{N\times D_z} \tag{1}$$

where N is the number of patches and D_z is the embedding dimension. LAGO is *agnostic to the choice of vision encoder*, and can leverage either pretrained or jointly learned representations.

Multi-Modal Conditioning. All conditioning signals are projected into a shared condition space of dimension D_c. A learned condition-type embedding $\tau \in \mathbb{R}^{D_c}$ is added to each conditioning vector to inform the predictor of the operating mode.

Action conditioning. For next-step transition prediction, low-level actions are encoded via a shared two-layer MLP $\phi_{act} : \mathbb{R}^{D_a} \to \mathbb{R}^{D_c}$. Continuous actions $a_t \in \mathbb{R}^{D_a}$ are passed to ϕ_{act} directly, while

discrete actions are first one-hot encoded into $\mathbb{R}^{D_a}$ (with D_a the number of discrete actions) and passed through the same MLP. The resulting conditioning vector is:

$$e_t^{act} = \phi_{act}(a_t), \qquad c_t^{act} = e_t^{act} + \tau_{act} \in \mathbb{R}^{D_c} \tag{2}$$

Subgoal conditioning. For subgoal prediction, a language instruction is first embedded by a language embedding model $\omega(\cdot)$ to obtain $g \in \mathbb{R}^{D_L}$, then projected to $\mathbb{R}^{D_c}$ via a two-layer MLP $\phi_{lang} : \mathbb{R}^{D_L} \to \mathbb{R}^{D_c}$. A completion scalar $\rho_t \in [0, 1]$ estimating task progress is projected independently to $\mathbb{R}^{D_c}$ via a separate two-layer MLP $\phi_{comp} : \mathbb{R} \to \mathbb{R}^{D_c}$. The subgoal conditioning vector combines both signals:

$$e_t^{lang} = \phi_{lang}(g), \quad e_t^{comp} = \phi_{comp}(\rho_t) \tag{3}$$

$$c_t^{sg} = e_t^{lang} + e_t^{comp} + \tau_{sg} \in \mathbb{R}^{D_c} \tag{4}$$

Conditioning injection. The final condition vector $c_t \in \mathbb{R}^{D_c}$ (either c_t^{act} or c_t^{sg} depending on the operating mode) is broadcast across all N patch token positions and concatenated with the latent $z_t \in \mathbb{R}^{N \times D_z}$, yielding the conditioned input $z_t \oplus c_t \in \mathbb{R}^{N \times (D_z + D_c)}$.

Latent Predictor. The predictor $\pi(\cdot)$ operates over the conditioned token sequence $z_t \oplus c_t \in \mathbb{R}^{N \times (D_z + D_c)}$. Before returning the processed tokens, we remove the broadcasted condition tokens from the patch tokens to recover the original embedding dimension of the visual encoder D_z:

$$\tilde{z} = \pi(z_t \oplus c_t) \in \mathbb{R}^{N \times D_z} \tag{5}$$

Training Objectives. The model is trained with a unified mean-squared error objective.

Next-step transition prediction. Transition samples are drawn from an action-labeled interaction dataset. Each sample consists of a state observation o_t, the executed action a_t and the subsequent observation o_{t+1}. Using a_t, we construct c_t^{act} as specified in (2), and train the predictor to minimize:

$$\mathcal{L}_{\text{act}} = \|\tilde{z}_{t+1} - z_{t+1}\|_2^2, \quad \tilde{z}_{t+1} = \pi(z_t \oplus c_t^{act}) \tag{6}$$

Subgoal prediction. Subgoal samples are drawn from an action-less demonstration dataset. Given a demonstration episode of length G, we sample an initial state observation o_j with index $j \in \{0, \ldots, G-2\}$, and a subgoal state observation o_k with index $k \in \{j+1, \ldots, G-1\}$. From these, we set the completion scalar as

$$\rho = \frac{k - j}{G - 1 - j} \in (0, 1]. \tag{7}$$

Intuitively, this encodes how far along the episode the subgoal lies relative to o_j. With this completion scalar, we construct c^{sg} as described in (3) and (4), and train the predictor to minimize:

$$\mathcal{L}_{\text{subgoal}} = \|\tilde{z}_\rho - z_\rho\|_2^2, \quad \tilde{z}_\rho = \pi(z_j \oplus c^{sg}) \tag{8}$$

Because both tasks optimize the same latent-space prediction objective under different conditioning inputs, transition and subgoal samples can be freely intermixed within a single batch and optimized jointly. A single forward pass through π thus learns both environment dynamics and subgoal structure, with no separate heads, losses, or training stages required.

3.2 Planning with LAGO

Subgoal Generation. Given the current observation o_t and a language-specified goal, we generate K latent subgoals $\{z_1^{sg}, \ldots, z_K^{sg}\}$ at uniformly spaced completion scalars $\rho_k = \frac{k}{K}$, $k \in \{1, \ldots, K\}$ using the subgoal conditioning described in Section 3.1. All subgoals are predicted independently from the current z_t rather than autoregressively, avoiding compounding prediction errors. Subgoals are repredicted at every MPC step, so the sequence continuously adapts to the agent's evolving state.

Action Optimization and Cost Formulation. At each planning step, the CEM solver samples candidate action sequences over a fixed horizon H, generating latent state rollouts $\tilde{z}_{1:H}$. To evaluate

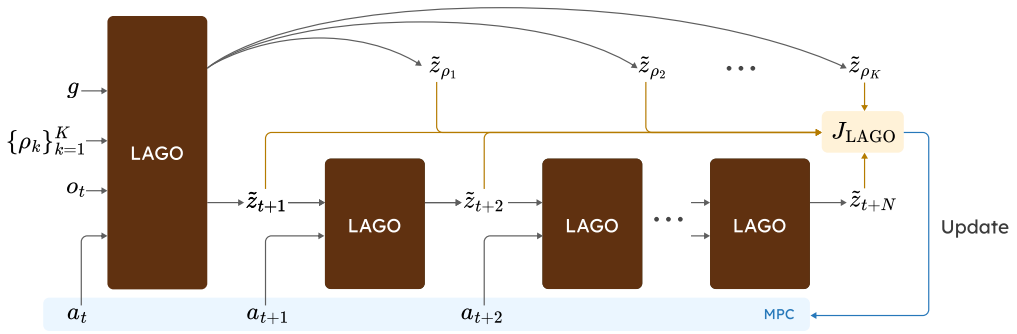


Figure 3: **Planning with LAGO.** At each step, LAGO generates a sequence of K latent subgoals from an arbitrary list of completion scalars $\{\rho_k\}_{k=1}^K$, thereby connecting the current state to the language-specified goal. A CEM planner then optimizes action sequences by evaluating candidate rollouts against the subgoals via J_{LAGO}, a soft minimum cost that measures the best alignment achieved between each subgoal and any timestep along the rollout. Subgoals are repredicted at every planning step, allowing the agent to continuously adapt to its evolving state.

these rollouts against the predicted subgoals, we introduce a soft minimum trajectory cost that measures, for each subgoal, the best alignment achieved at any point along the rollout:

$$J_{\text{LAGO}} = \sum_{k=1}^{K} \lambda^k \cdot \left(-\tau \log \sum_{h=1}^{H} \exp\left(-\frac{1}{\tau} \text{MSE}\left(\tilde{z}_h,\, z_k^{sg}\right) \right) \right) \tag{9}$$

where $\lambda \in (0, 1]$ is a subgoal decay factor and $\tau > 0$ is a temperature parameter controlling the hardness of the temporal aggregation. For $\tau \to 0$, the soft-min approaches a hard minimum, selecting only the closest timestep to each subgoal. As τ increases, timesteps near the minimum contribute proportionally more, favoring trajectories that converge to and remain near a subgoal over those that only transiently visit it. The decay λ^k prioritizes earlier subgoals over distant ones. A visualization of the loss landscape produced by J_{LAGO} can be found in Appendix D.

4 Experiments

4.1 Experimental Setup

Implementation. We use a frozen DINOv2 ViT-S/14 (Oquab et al., 2024) as our vision encoder ψ, which has been shown to provide a rich representation space for model-based planning (Zhou et al., 2025). Language instructions are embedded using EmbeddingGemma 300M (Vera et al., 2025), yielding fixed-length representations for all language tasks. The latent predictor π is implemented as a ViT (Dosovitskiy et al., 2021). To stress-test long-horizon planning, we train all methods with no frameskip and fix the planning horizon to 10 rollout steps. Full architectural details, hyperparameters, and optimization settings are provided in Appendix C.

Data Collection. We rely on the SWM library (Maes et al., 2026a) for data collection in all environments except Kingdom. The suboptimal interaction dataset used to learn next-step transition dynamics is comprised of 50k environment transitions. While the majority of these transitions are collected using a random policy, we also include noisy expert policy rollouts. This mixture is necessary to provide coverage of both object manipulation in control tasks and room traversal in navigation tasks (Wang et al., 2026; Maes et al., 2026b). The demonstration dataset used to learn subgoal prediction is comprised of 10k language-labeled expert trajectories.

Environments. We evaluate on the following environments. **(1)** TwoRoom (Sobal et al., 2025) is a 2D continuous-control navigation task with two rooms joined by a narrow doorway. The agent must navigate from a starting position to a target one. **(2)** Kingdom is a 2D grid-world we introduce, with randomized layouts and irreversible water hazards that stress subgoal generation and long-horizon planning. The agent must navigate to a desired location. **(3)** OGBench Cube (Park et al., 2025) is a manipulation task where a robotic arm moves a cube to a target location. We apply a distance-based gripper heuristic to all planners. **(4)** OGBench Scene (Park et al., 2025) is a fixed-layout manipulation task where the agent must interact with a drawer, window, and cube. While image goal baselines are

provided an image of the agent in the desired location to solve the task, we adapt all environments to support language goals (e.g., "place the red dot at position $[x, y]$" or "reach the left of the dragon"). Prior work (Zhou et al., 2025; Wang et al., 2026; Maes et al., 2026b) evaluated methods in settings where targets are placed randomly within a fixed distance, often yielding near-solved or trivial instances (see Appendix A.2.1 for examples). In contrast, we set each level to a fixed start-to-target distance, eliminating the risk of such degenerate cases and stress-testing long-horizon planning. All environment-generation seeds used to construct the training datasets are stored with the corresponding data. Evaluation uses held-out environment seeds that do not overlap with any training seed. We report success rate, averaged over 10 seeds per difficulty level. Methods that plan with CEM are additionally repeated 5 times per seed, since their planning is stochastic. Full details can be found in Appendix A.

Image goal baselines. We compare against the following recent world-model baselines: **(1)** DINO-WM (Zhou et al., 2025) is a JEPA architecture using a frozen DINOv2 (Oquab et al., 2024) as its vision encoder. **(2)** Temporal Straightening (Wang et al., 2026) (T.S.) extends DINO-WM by adding a learned projection over the frozen patch tokens and a temporal straightening regularization term. This encourages the latent space to reduce curvature across successive states along expert trajectories. **(3)** LeWM (Maes et al., 2026b) is a fully end-to-end trained JEPA that learns both encoder and predictor from pixels, relying exclusively on the SIGReg regularizer (Balestriero & LeCun, 2025) to prevent representation collapse. **(4)** HWM (Zhang et al., 2026) adds a high-level latent world model on top of a low-level one, operating over learned latent macro-actions at a coarser timescale. At test time, it plans macro-actions toward the goal image with the high-level model, and the first predicted latent state serves as a subgoal for low-level planning over primitive actions. **(5)** We introduce two oracle-subgoal (O.S.) variants, in which subgoals are replaced at every planning step by oracle ones, obtained by recomputing the shortest path to the goal and rendering equally spaced waypoints along it. LAGO (O.S.) uses the same checkpoint as LAGO, but its predictor is used only for next-state prediction. DINO-WM (O.S.) uses the same oracle subgoals and the same J_{LAGO} planning objective, but obtains next-state predictions from the DINO-WM baseline.

To ensure a fair comparison with LAGO, all image-goal baselines are trained using next-step transition samples from both the interaction and the demonstration datasets. *LAGO ignores the demonstration actions, using the action-less demonstrations solely for subgoal learning*. All methods are provided an equal compute budget and the evaluated checkpoints are selected based on validation accuracy. We provide visualizations of the decoded latent space for LeWM and Temporal Straightening in Appendix F.

Language Goal Baselines. We compare against the following language-conditioned baselines: **(1)** LIV (Ma et al., 2023) is a contrastive vision-language model trained on robotic interaction data with additional supervision for value estimation. We finetune LIV on the expert trajectories. **(2)** PE-Core (Bolya et al., 2025) is a general-purpose vision-language encoder trained with a contrastive objective on large-scale multimodal data. It provides a strong off-the-shelf baseline for language grounding without robotics-specific adaptation. We use frozen representations without finetuning. **(3)** Robo-Dopamine (Tan et al., 2025) is a general-purpose reward modeling framework for robotic manipulation, built around a generative vision-language General Reward Model (GRM) that estimates task progress by conditioning on state images and language instructions. We use frozen representations without finetuning. **(4)** GCBC is a goal-conditioned behavior cloning policy trained to directly imitate the demonstration actions from the current state and language embedding, without a world model, subgoals, or any planning at test time.

LIV, PE-Core, and Robo-Dopamine are evaluated as reward models for MPC planning. Candidate rollouts are generated with the true environment dynamics and the resulting renders are scored directly by each method's vision-language reward.

4.2 Performance of Planning with Self-Predicted Subgoals

We evaluate LAGO against image-goal and language-conditioned baselines on three environments of increasing planning difficulty, asking: **(i)** "*does LAGO's subgoal prediction improve long-horizon planning over baselines?*" **(ii)** "*how much is lost because subgoals are predicted rather than perfect?*" **(iii)** "*does the dual-objective design help beyond the subgoals themselves?*" To answer (ii) and (iii), we use the (O.S.) variants. LAGO and DINO-WM use the same frozen DINOv2 representation space and predictor architecture, making DINO-WM (O.S.) a controlled comparison in which the subgoal

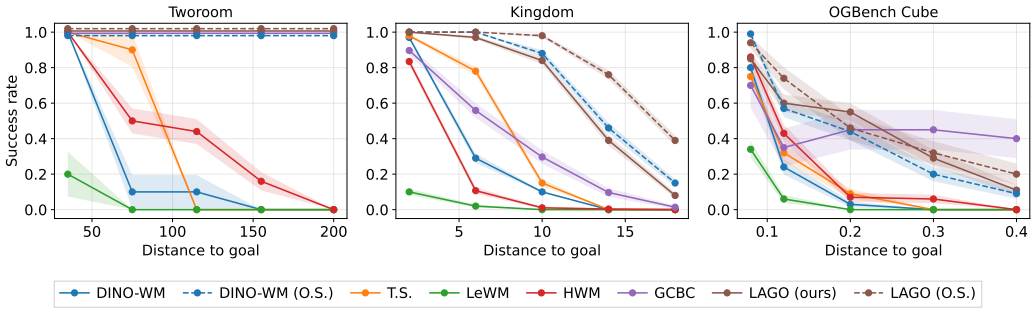


Figure 4: **Performance of Planning with Self-Predicted Subgoals.** Success rate vs. distance-to-goal on TwoRoom, Kingdom (averaged over all goals), and OGBench Cube. (O.S.): oracle subgoals.

branch is replaced by oracle subgoals and the two prediction modes are trained separately. Thus, comparing LAGO with LAGO (O.S.) measures the performance cost associated with predicting subgoals, while comparing LAGO (O.S.) with DINO-WM (O.S.) tests whether the joint training of subgoal and transition prediction improves the learned planning dynamics. Figure 4 reports success rate as a function of environment difficulty; Kingdom is averaged over all possible goals.

LAGO matches GCBC and both oracle-subgoal variants on TwoRoom, where they all achieve 100% success at every difficulty. It substantially exceeds Temporal Straightening (T.S.) and LeWM at the three longest distances, where both methods reach zero success, while HWM degrades more gradually but also reaches zero. Kingdom provides the clearest long-horizon test as shortest paths are often highly curved around obstacles, and successful trajectories must anticipate terminal hazards such as water tiles before reaching them. LAGO outperforms every non-oracle baseline at every difficulty on Kingdom, while GCBC degrades from 90% to 1% success as difficulty increases. At difficulty 10, for example, LAGO achieves 84% success compared with 30% for GCBC. This gap is consistent with the advantage of explicitly planning through intermediate states rather than directly predicting actions from the current observation and language goal. On OGBench Cube, the gap between methods is smaller because the task mainly requires reaching the cube and then moving it to the target, with little of the detour planning and terminal-hazard avoidance that distinguishes Kingdom. LAGO nevertheless outperforms GCBC at the two lowest difficulties and is comparable at 0.2, while GCBC is stronger at the two largest (45% vs 29% and 40% vs 11%), likely because grasp dynamics are hard for action-conditioned rollouts to model.

The oracle-subgoal comparisons locate LAGO's remaining performance gap. On TwoRoom and OGBench Cube, LAGO matches LAGO (O.S.), indicating that predicted subgoals are sufficiently accurate on these tasks. On Kingdom, LAGO performs comparably to LAGO (O.S.) at difficulties 2 and 6, but the gap widens from difficulty 10 onward. This suggests that subgoal quality, rather than the planning procedure, is the primary bottleneck at the longest horizons. Separately, LAGO (O.S.) exceeds DINO-WM (O.S.) in most unsaturated settings, most clearly at long Kingdom horizons (76% vs 46% at difficulty 14 and 39% vs 15% at difficulty 18). This supports the hypothesis that joint *dual-objective training improves the transition model used for planning*, and the benefit is not explained by subgoal prediction alone. On OGBench Cube, LAGO with predicted subgoals matches or exceeds DINO-WM (O.S.) at the four longest difficulties, showing that the jointly trained model can compensate for imperfect subgoals.

4.3 Language Grounding Performance

We evaluate the performance of grounding language directly in the world model's representation space, as opposed to offloading progress estimation to a dedicated vision-language model. We compare LAGO against language-goal baselines evaluated on Kingdom and OGBench Scene across all available tasks at fixed difficulty: Kingdom is set to 6 steps, and OGBench Scene fixes the starting position to the center of the environment. Results are reported as success rates in Figure 5, showing that PE-Core provides no meaningful signal for language-conditioned planning (6% in Kingdom, 0% in OGBench Scene), and LIV exhibits a similar trend (8.8% and 12.5%, respectively). LIV-finetuned is stronger due to its direct adaptation to each environment's expert demonstrations, yet still falls short of LAGO ($+44.4\%$ in Kingdom and $+6.6\%$ in OGBench Scene), indicating that grounding language within the world model's representation space outperforms delegating this grounding to a separate,

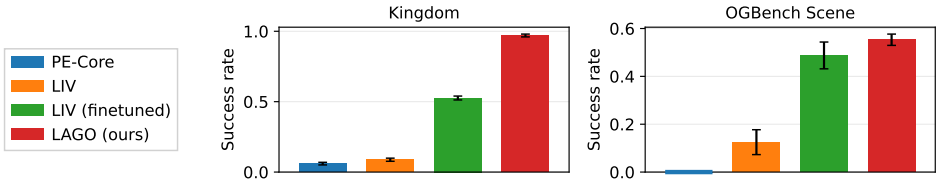


Figure 5: **LAGO vs. vision-language reward baselines.** Success rate averaged across all language goals on Kingdom (difficulty 6) and OGBench Scene. Baselines score rollouts simulated with true dynamics; LAGO plans in latent space.

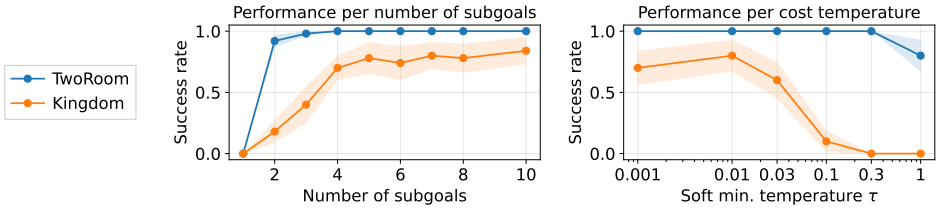


Figure 6: **Ablations of LAGO's planning components.** Success rates on Kingdom and TwoRoom as a function of **(left)** the number of subgoals K and **(right)** the cost temperature τ.

dedicated VLM. We further study the effect of scaling language understanding independently from next-state prediction in Appendix H.

Robo-Dopamine (Tan et al., 2025) is a large generative reward model, poorly suited to the high-sampling demands of model-based planning. Over 200 GPU-synced trials at batch size 1, it takes 451 ± 13ms per observation, versus 3.7 ± 1.4ms for LIV-finetuned and 11.0 ± 0.7ms for LAGO ($K = 5$). This $121\times$ slowdown rules it out as a reward signal for MPC under practical time constraints, so we instead assess its reward quality with the VOC metric (Ma et al., 2024) in Appendix G.

4.4 Model Analysis

For this analysis, we fix difficulty at 155 for TwoRoom and 10 for Kingdom. Complete results tables can be found in Appendix B.

Number of Subgoals. We vary $K \in \{1, \ldots, 10\}$ with all other hyperparameters fixed (Figure 6). A single subgoal fails in both environments, and performance is poor at low K because the planner must bridge large latent distances between subgoals. TwoRoom recovers almost immediately, while Kingdom recovers gradually. From $K = 5$ onward, performance plateaus (TwoRoom at 100%, Kingdom within error), so LAGO is insensitive to K once moderately large. We use $K = 5$ throughout our experiments.

Cost Function Analysis. We sweep the soft-min temperature τ of J_{LAGO} (Eq. 9), which controls how broadly each subgoal is matched across rollout timesteps (Figure 6). Low τ focuses on the best-aligned timestep, so each subgoal need only be visited at some point; high τ approaches a uniform sum that pulls every step toward every subgoal, implicitly enforcing straight-line interpolation. Lower temperatures perform substantially better: Kingdom peaks at 80% for $\tau = 0.01$ and degrades monotonically as temperature increases, falling to 0% at $\tau = 0.3$. Meanwhile, TwoRoom remains saturated (100%) across $\tau \leq 0.3$ and drops only mildly to 80% at $\tau = 1.0$. This is consistent with Kingdom's shortest paths curving around obstacles, which requires rollout flexibility to avoid terminal water tiles. We select $\tau = 0.001$ throughout.

5 Conclusion

We introduced LAGO, a hierarchical world model in which a single latent predictor forecasts both action-conditioned dynamics and language-conditioned subgoals. Planning against these subgoals with a soft-minimum alignment cost turns a long-horizon instruction into a sequence of locally tractable objectives. The gains are largest where optimal paths are non-trivial. On Kingdom, whose shortest paths curve around terminal hazards, LAGO reaches 84% success against 30% for GCBC. Grounding language in the world model's own latent space also outperforms dedicated vision-language reward models without access to ground-truth dynamics.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods, edit a research paper to improve readability, sourcing/searching for information, identify relevant literature. We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results. We have reviewed all AI-assisted work. All experiments were hand reviewed and tested by two of the authors to ensure correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

AgiBot-World-Contributors, Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, Shu Jiang, Yuxin Jiang, Cheng Jing, Hongyang Li, Jialu Li, Chiming Liu, Yi Liu, Yuxiang Lu, Jianlan Luo, Ping Luo, Yao Mu, Yuehan Niu, Yixuan Pan, Jiangmiao Pang, Yu Qiao, Guanghui Ren, Cheng Ruan, Jiaqi Shan, Yongjian Shen, Chengshi Shi, Mingkang Shi, Modi Shi, Chonghao Sima, Jianheng Song, Huijie Wang, Wenhao Wang, Dafeng Wei, Chengen Xie, Guo Xu, Junchi Yan, Cunbiao Yang, Lei Yang, Shukai Yang, Maoqing Yao, Jia Zeng, Chi Zhang, Qinglin Zhang, Bin Zhao, Chengyue Zhao, Jiaqi Zhao, and Jianchao Zhu. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems, 2025. URL https://arxiv.org/abs/2503.06669.

Anurag Ajay, Seungwook Han, Yilun Du, Shuang Li, Abhi Gupta, Tommi Jaakkola, Josh Tenenbaum, Leslie Kaelbling, Akash Srivastava, and Pulkit Agrawal. Compositional foundation models for hierarchical planning, 2023. URL https://arxiv.org/abs/2309.08587.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture, 2023. URL https://arxiv.org/abs/2301.08243.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba, Komeili, Matthew 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, 2025. URL https://arxiv.org/abs/2506.09985.

Randall Balestriero and Yann LeCun. Lejepa: Provable and scalable self-supervised learning without the heuristics, 2025. URL https://arxiv.org/abs/2511.08544.

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, 2023. URL https://arxiv.org/abs/2310.10639.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

Daniel Bolya, Po-Yao Huang, Peize Sun, Jang Hyun Cho, Andrea Madotto, Chen Wei, Tengyu Ma, Jiale Zhi, Jathushan Rajasegaran, Hanoona Rasheed, Junke Wang, Marco Monteiro, Hu Xu, Shiyu Dong, Nikhila Ravi, Daniel Li, Piotr Dollár, and Christoph Feichtenhofer. Perception

encoder: The best visual embeddings are not at the output of the network, 2025. URL https://arxiv.org/abs/2504.13181.

Anthony Brohan et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023.

Chang Chen, Fei Deng, Kenji Kawaguchi, Caglar Gulcehre, and Sungjin Ahn. Simple hierarchical planning with diffusion. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=kXHEBK9uAY.

Delong Chen, Tejaswi Kasarla, Yejin Bang, Mustafa Shukor, Willy Chung, Jade Yu, Allen Bolourchi, Theo Moutakanni, and Pascale Fung. Action100m: A large-scale video action dataset, 2026. URL https://arxiv.org/abs/2601.10592.

Tal Daniel, Carl Qi, Dan Haramati, Amir Zadeh, Chuan Li, Aviv Tamar, Deepak Pathak, and David Held. Latent particle world models: Self-supervised object-centric stochastic dynamics modeling, 2026. URL https://arxiv.org/abs/2603.04553.

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, 2021. URL https://arxiv.org/abs/2010.11929.

Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation, 2023a. URL https://arxiv.org/abs/2302.00111.

Yilun Du, Mengjiao Yang, Pete Florence, Fei Xia, Ayzaan Wahid, Brian Ichter, Pierre Sermanet, Tianhe Yu, Pieter Abbeel, Joshua B. Tenenbaum, Leslie Kaelbling, Andy Zeng, and Jonathan Tompson. Video language planning, 2023b. URL https://arxiv.org/abs/2310.10625.

Yuwei Fu, Haichao Zhang, Di Wu, Wei Xu, and Benoit Boulet. Furl: Visual-language models as fuzzy rewards for reinforcement learning, 2024. URL https://arxiv.org/abs/2406.00645.

Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, Miguel Martin, Tushar Nagarajan, Ilija Radosavovic, Santhosh Kumar Ramakrishnan, Fiona Ryan, Jayant Sharma, Michael Wray, Mengmeng Xu, Eric Zhongcong Xu, Chen Zhao, Siddhant Bansal, Dhruv Batra, Vincent Cartillier, Sean Crane, Tien Do, Morrie Doulaty, Akshay Erapalli, Christoph Feichtenhofer, Adriano Fragomeni, Qichen Fu, Abrham Gebreselasie, Cristina Gonzalez, James Hillis, Xuhua Huang, Yifei Huang, Wenqi Jia, Weslie Khoo, Jachym Kolar, Satwik Kottur, Anurag Kumar, Federico Landini, Chao Li, Yanghao Li, Zhenqiang Li, Karttikeya Mangalam, Raghava Modhugu, Jonathan Munro, Tullie Murrell, Takumi Nishiyasu, Will Price, Paola Ruiz Puentes, Merey Ramazanova, Leda Sari, Kiran Somasundaram, Audrey Southerland, Yusuke Sugano, Ruijie Tao, Minh Vo, Yuchen Wang, Xindi Wu, Takuma Yagi, Ziwei Zhao, Yunyi Zhu, Pablo Arbelaez, David Crandall, Dima Damen, Giovanni Maria Farinella, Christian Fuegen, Bernard Ghanem, Vamsi Krishna Ithapu, C. V. Jawahar, Hanbyul Joo, Kris Kitani, Haizhou Li, Richard Newcombe, Aude Oliva, Hyun Soo Park, James M. Rehg, Yoichi Sato, Jianbo Shi, Mike Zheng Shou, Antonio Torralba, Lorenzo Torresani, Mingfei Yan, and Jitendra Malik. Ego4d: Around the world in 3,000 hours of egocentric video, 2022. URL https://arxiv.org/abs/2110.07058.

David Ha and Jürgen Schmidhuber. World models. *Zenodo*, 2018. doi: 10.5281/ZENODO.1207631. URL https://zenodo.org/record/1207631.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning (ICML)*, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *ICLR*, 2020.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *International Conference on Machine Learning (ICML)*, 2022.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Learning massively multitask world models for continuous control, 2025. URL https://arxiv.org/abs/2511.19584.

Ce Hao, Anxing Xiao, Zhiwei Xue, and Harold Soh. Chd: Coupled hierarchical diffusion for long-horizon tasks. In *Conference on Robot Learning*, pp. 1420–1451. PMLR, 2025.

Dan Haramati, Carl Qi, Tal Daniel, Amy Zhang, Aviv Tamar, and George Konidaris. Hierarchical entity-centric reinforcement learning with factored subgoal diffusion. *arXiv preprint arXiv:2602.02722*, 2026.

Bohan Hou, Gen Li, Jindou Jia, Tuo An, Xinying Guo, Sicong Leng, Haoran Geng, Yanjie Ze, Tatsuya Harada, Philip Torr, Oier Mees, Marc Pollefeys, Zhuang Liu, Jiajun Wu, Pieter Abbeel, Jitendra Malik, Yilun Du, and Jianfei Yang. World model for robot learning: A comprehensive survey, 2026. URL https://arxiv.org/abs/2605.00080.

Zixuan Huang, Yating Lin, Fan Yang, and Dmitry Berenson. Subgoal diffuser: Coarse-to-fine subgoal generation to guide model predictive control for robot manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16489–16495. IEEE, 2024.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, Vitor Guizilini, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Muhammad Zubair Irshad, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. Droid: A large-scale in-the-wild robot manipulation dataset, 2025. URL https://arxiv.org/abs/2403.12945.

Po-Chen Ko, Jiayuan Mao, Yilun Du, Shao-Hua Sun, and Joshua B. Tenenbaum. Learning to act from actionless videos through dense correspondences, 2023. URL https://arxiv.org/abs/2310.08576.

Yann LeCun. A path towards autonomous machine intelligence, 2022. Version 0.9.2, June 27, 2022.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. Interactive language: Talking to robots in real time, 2022. URL https://arxiv.org/abs/2210.06407.

Yecheng Jason Ma, William Liang, Vaidehi Som, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. Liv: Language-image representations and rewards for robotic control, 2023. URL https://arxiv.org/abs/2306.00958.

Yecheng Jason Ma, Joey Hejna, Ayzaan Wahid, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, Jonathan Tompson, Osbert Bastani, Dinesh Jayaraman, Wenhao Yu, Tingnan Zhang, Dorsa Sadigh, and Fei Xia. Vision language models are in-context value learners, 2024. URL https://arxiv.org/abs/2411.04549.

Lucas Maes, Quentin Le Lidec, Dan Haramati, Nassim Massaudi, Damien Scieur, Yann LeCun, and Randall Balestriero. stable-worldmodel-v1: Reproducible world modeling research and evaluation, 2026a. URL https://arxiv.org/abs/2602.08968.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworldmodel: Stable end-to-end joint-embedding predictive architecture from pixels, 2026b. URL https://arxiv.org/abs/2603.19312.

Sergi Masip, Jonathan Swinnen, Yutong Hu, Renaud Detry, and Tinne Tuytelaars. Ff-jepa: Long-horizon planning in world models with latent planners, 2026. URL https://arxiv.org/abs/2606.09311.

Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips, 2019. URL https://arxiv.org/abs/1906.03327.

Suraj Nair and Chelsea Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation, 2019. URL https://arxiv.org/abs/1909.05829.

Suraj Nair, Eric Mitchell, Kevin Chen, Brian Ichter, Silvio Savarese, and Chelsea Finn. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation, 2021. URL https://arxiv.org/abs/2109.01115.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation, 2022. URL https://arxiv.org/abs/2203.12601.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. URL https://arxiv.org/abs/2304.07193.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl, 2025. URL https://arxiv.org/abs/2410.20092.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Juan Rocamonde, Victoriano Montesinos, Elvis Nava, Ethan Perez, and David Lindner. Vision-language models are zero-shot reward models for reinforcement learning, 2024. URL https://arxiv.org/abs/2310.12921.

Simon Roy, Samuel Barbeau, Giovanni Beltrame, Christian Desrosiers, and Nicolas Thome. Revisiting the learning objectives of vision-language reward models, 2025. URL https://arxiv.org/abs/2512.20675.

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, 2025. URL https://arxiv.org/abs/2502.14819.

Sumedh A Sontakke, Jesse Zhang, Sébastien M. R. Arnold, Karl Pertsch, Erdem Bıyık, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. Roboclip: One demonstration is enough to learn robot policies, 2023. URL https://arxiv.org/abs/2310.07899.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. The MIT Press, Cambridge, MA, 2nd edition, 2018.

Huajie Tan, Sixiang Chen, Yijie Xu, Zixiao Wang, Yuheng Ji, Cheng Chi, Yaoxu Lyu, Zhongxia Zhao, Xiansheng Chen, Peterson Co, Shaoxuan Xie, Guocai Yao, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. Robo-dopamine: General process reward modeling for high-precision robotic manipulation, 2025. URL https://arxiv.org/abs/2512.23703.

Henrique Schechter Vera, Sahil Dua, Biao Zhang, Daniel Salz, Ryan Mullins, Sindhu Raghuram Panyam, Sara Smoot, Iftekhar Naim, Joe Zou, Feiyang Chen, Daniel Cer, Alice Lisak, Min Choi, Lucas Gonzalez, Omar Sanseviero, Glenn Cameron, Ian Ballantyne, Kat Black, Kaifeng Chen, Weiyi Wang, Zhe Li, Gus Martins, Jinhyuk Lee, Mark Sherwood, Juyeong Ji, Renjie Wu, Jingxiao Zheng, Jyotinder Singh, Abheesht Sharma, Divyashree Sreepathihalli, Aashi Jain, Adham Elarabawy, AJ Co, Andreas Doumanoglou, Babak Samari, Ben Hora, Brian Potetz, Dahun Kim, Enrique Alfonseca, Fedor Moiseev, Feng Han, Frank Palma Gomez, Gustavo Hernández Ábrego, Hesen Zhang, Hui Hui, Jay Han, Karan Gill, Ke Chen, Koert Chen, Madhuri Shanbhogue, Michael Boratko, Paul Suganthan, Sai Meher Karthik Duddu, Sandeep Mariserla, Setareh Ariafar, Shanfeng Zhang, Shijie Zhang, Simon Baumgartner, Sonam Goenka, Steve Qiu, Tanmaya Dabral, Trevor Walker, Vikram Rao, Waleed Khawaja, Wenlei Zhou, Xiaoqi Ren, Ye Xia, Yichang Chen, Yi-Ting Chen, Zhe Dong, Zhongli Ding, Francesco Visin, Gaël Liu, Jiageng Zhang, Kathleen Kenealy, Michelle Casbon, Ravin Kumar, Thomas Mesnard, Zach Gleicher, Cormac Brick, Olivier Lacombe, Adam Roberts, Qin Yin, Yunhsuan Sung, Raphael Hoffmann, Tris Warkentin, Armand Joulin, Tom Duerig, and Mojtaba Seyedhosseini. Embeddinggemma: Powerful and lightweight text representations, 2025. URL https://arxiv.org/abs/2509.20354.

Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning (CoRL)*, 2023.

Ying Wang, Oumayma Bounou, Gaoyue Zhou, Randall Balestriero, Tim G. J. Rudner, Yann LeCun, and Mengye Ren. Temporal straightening for latent planning, 2026. URL https://arxiv.org/abs/2603.12231.

Zihan Wang et al. Rl-vlm-f: Reinforcement learning with vision-language foundation models. In *International Conference on Machine Learning (ICML)*, 2024.

Wancong Zhang, Basile Terver, Artem Zholus, Soham Chitnis, Harsh Sutaria, Mido Assran, Randall Balestriero, Amir Bar, Adrien Bardes, Yann LeCun, and Nicolas Ballas. Hierarchical planning with latent world models, 2026. URL https://arxiv.org/abs/2604.03208.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning, 2025. URL https://arxiv.org/abs/2411.04983.

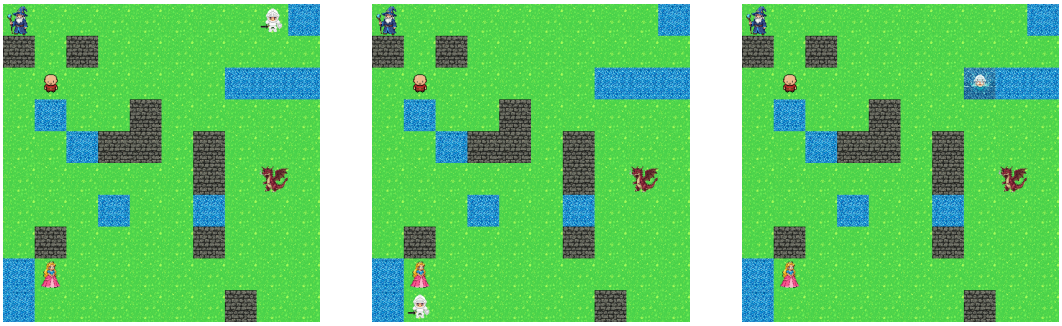
A ENVIRONMENTS

A.1 ENVIRONMENT DETAILS

TwoRoom (Sobal et al., 2025) is a 2D continuous-control navigation task consisting of two rooms connected by a narrow doorway. The agent is initialized in one room and tasked with reaching a target location in the other. Although goals are typically specified as images of the target state, we instead support latent goal prediction from language by expressing targets as "place the red dot at position [x, y]". To stress-test long-horizon planning, we group test seeds into five difficulty levels based on expert trajectory length. Prior work (Zhou et al., 2025; Wang et al., 2026; Maes et al., 2026b) evaluated settings where targets lay within 25 steps of the start, often yielding near-solved or trivial instances (see Appendix A.2.1 for examples). In contrast, we set each level to a fixed start-to-target distance, eliminating the risk of such degenerate cases.

Kingdom is a 2D navigation environment that we introduce to stress-test model-based planning and subgoal generation under dynamic spatial layouts and irreversible environmental effects. The environment is represented as a discrete grid of cells, and water cells act as terminal hazards: if the agent enters a water cell, the episode ends immediately. This introduces irreversible failure states and strengthens the need for precise long-horizon planning. At each episode reset, the positions of all entities, including the agent (a knight), four non-player characters (villager, princess, dragon, wizard), static obstacles, and water tiles, are randomly reconfigured. This ensures that neither goal locations nor feasible paths are consistent across episodes, so successful behavior cannot be memorized and must instead be inferred from the current scene configuration at each rollout.

Kingdom defines 30 language goals with relational objectives such as "reach the left of the dragon" whose satisfaction depends jointly on the current arrangement of entities and obstacles. As in TwoRoom, we partition evaluation into multiple difficulty levels based on expert trajectory length, enabling a controlled assessment of long-horizon planning (see Appendix A.2.2 for examples).



(a) Randomized initial state. (b) "Reach below the princess". (c) Drowning failure

Figure 7: **Kingdom environment**. Example frames from randomized grid-world layouts containing walls, NPCs (villager, princess, dragon, wizard), and water. (a) The agent at rest. (b) A goal-satisfying state for the task "reach below the princess", where the agent stands on the tile directly beneath the princess. (c) Water tiles are walkable but terminal, so a single misstep ends the episode, which makes planning difficult even at short horizons.

OGBench Cube (Park et al., 2025) is a manipulation task where a robotic arm must move a cube to a target location on a tabletop. Each episode randomizes the positions of the robot, cube, and goal, preventing reliance on fixed spatial configurations. We enable latent goal prediction from language by expressing targets as "move the cube to position [x, y]". Task difficulty is controlled by categorizing episodes according to the required displacement, defined as the sum of the end-effector-to-cube and cube-to-goal distances. Since LAGO and all planning baselines learn exclusively from pixels, grasping is difficult to predict reliably. We therefore use a simple distance-based heuristic that automatically closes the gripper when the end-effector is within a specified distance of the cube. This heuristic is applied to all model-based planning methods.

OGBench Scene (Park et al., 2025) is a manipulation environment with three interactable objects: a drawer (which can be opened, closed, locked, or unlocked via a button), a window (similarly

button-controlled), and a cube that can be rearranged or placed inside the drawer. Unlike Kingdom or OGBench Cube, the scene layout is largely fixed so goal localization is static and the planning challenge lies primarily in task ordering and execution rather than spatial reasoning.

A.2 Evaluation Examples

A.2.1 TwoRoom

To illustrate the difference between prior evaluation protocols and our difficulty-graded benchmark, we provide randomly selected test configurations from both settings below.

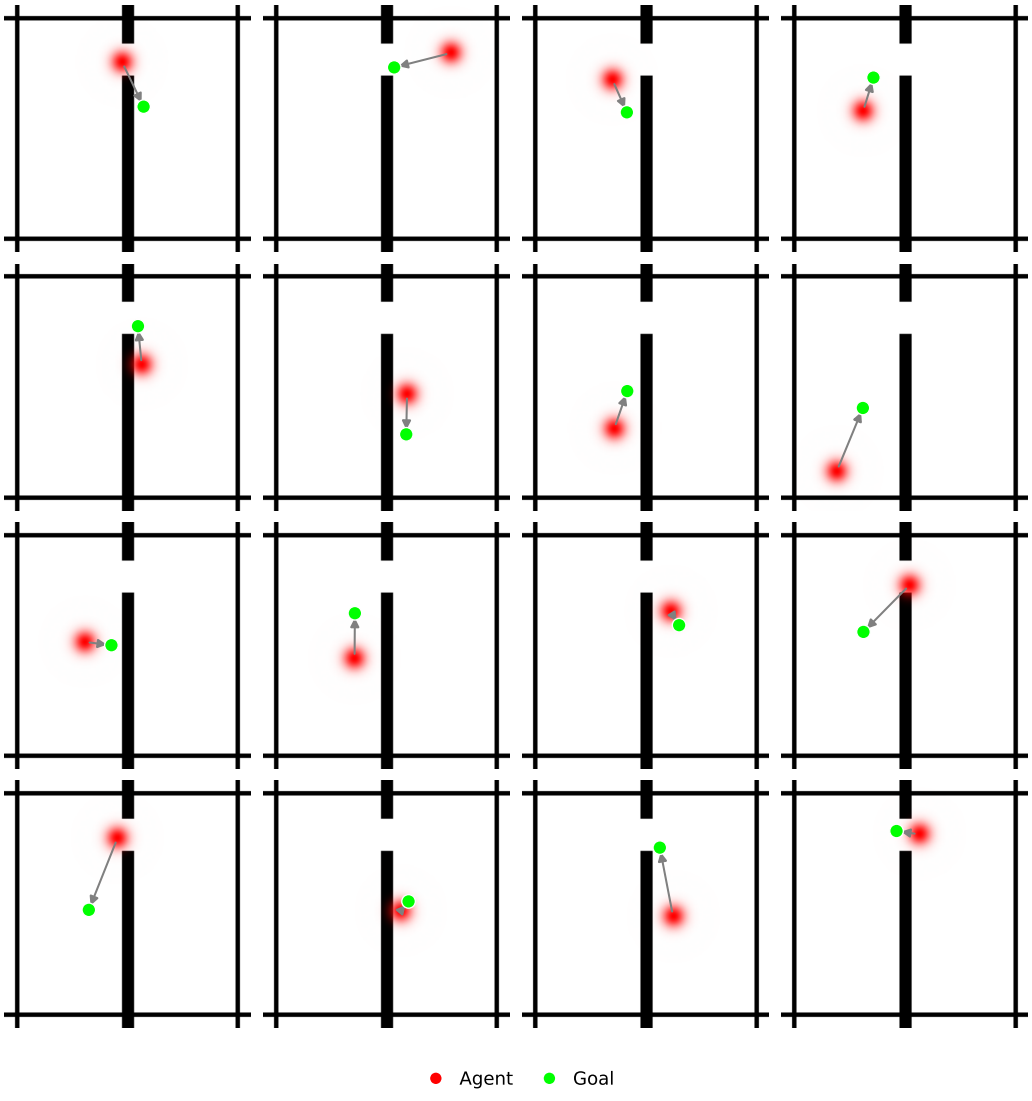


Figure 8: Randomly selected examples of prior TwoRoom test configurations, where start (*red*) and goal (*green*) states are configured to be at most 25 steps apart, often resulting in near-trivial or already-solved instances.

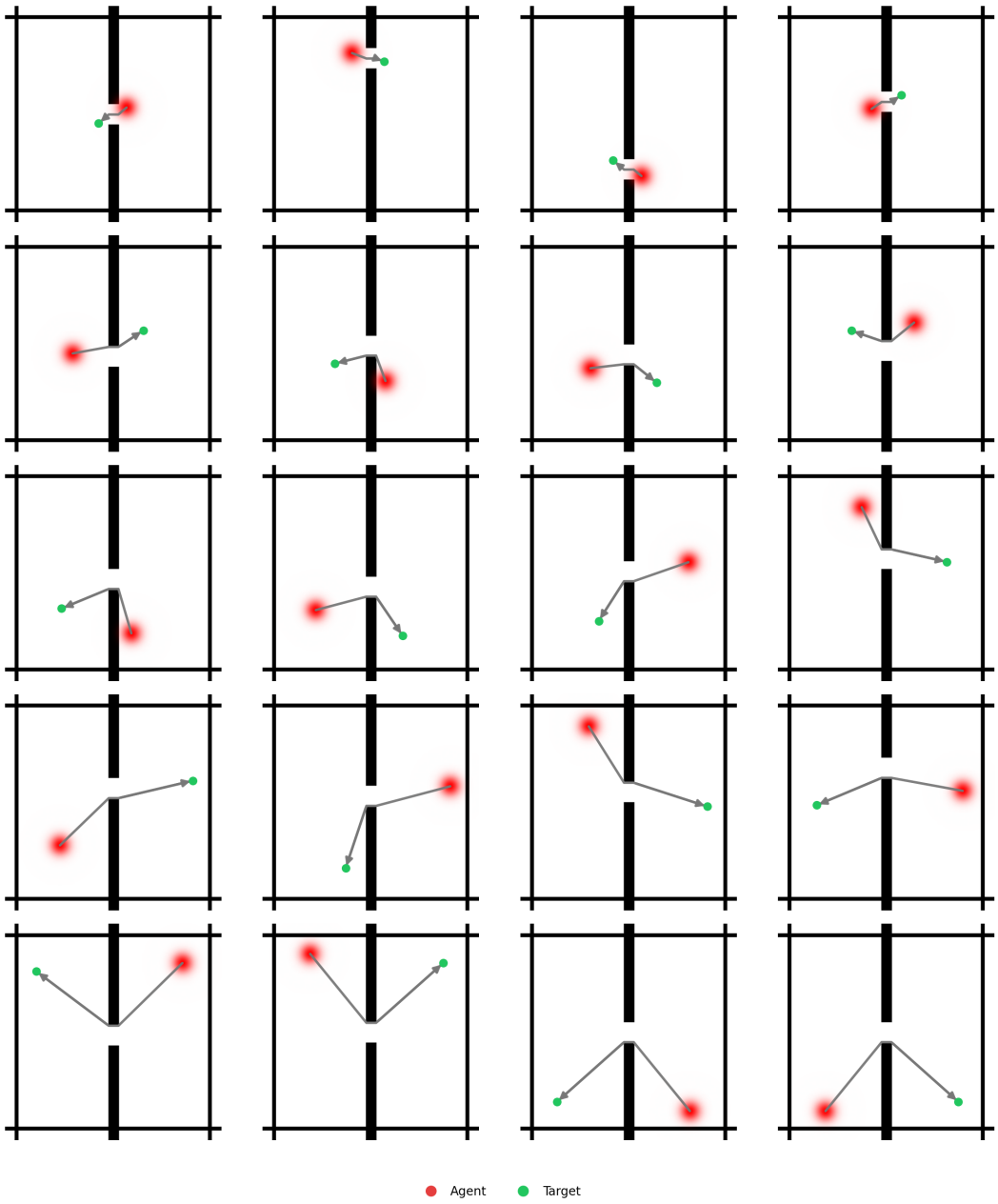


Figure 9: Randomly selected examples of our difficulty-graded TwoRoom test configurations. Each row represents a different difficulty level. While previous test suites only enforce a maximum distance, we enforce an exact distance between the starting and target positions to better evaluate the long-horizon capacity of model-based planning.

A.2.2 KINGDOM

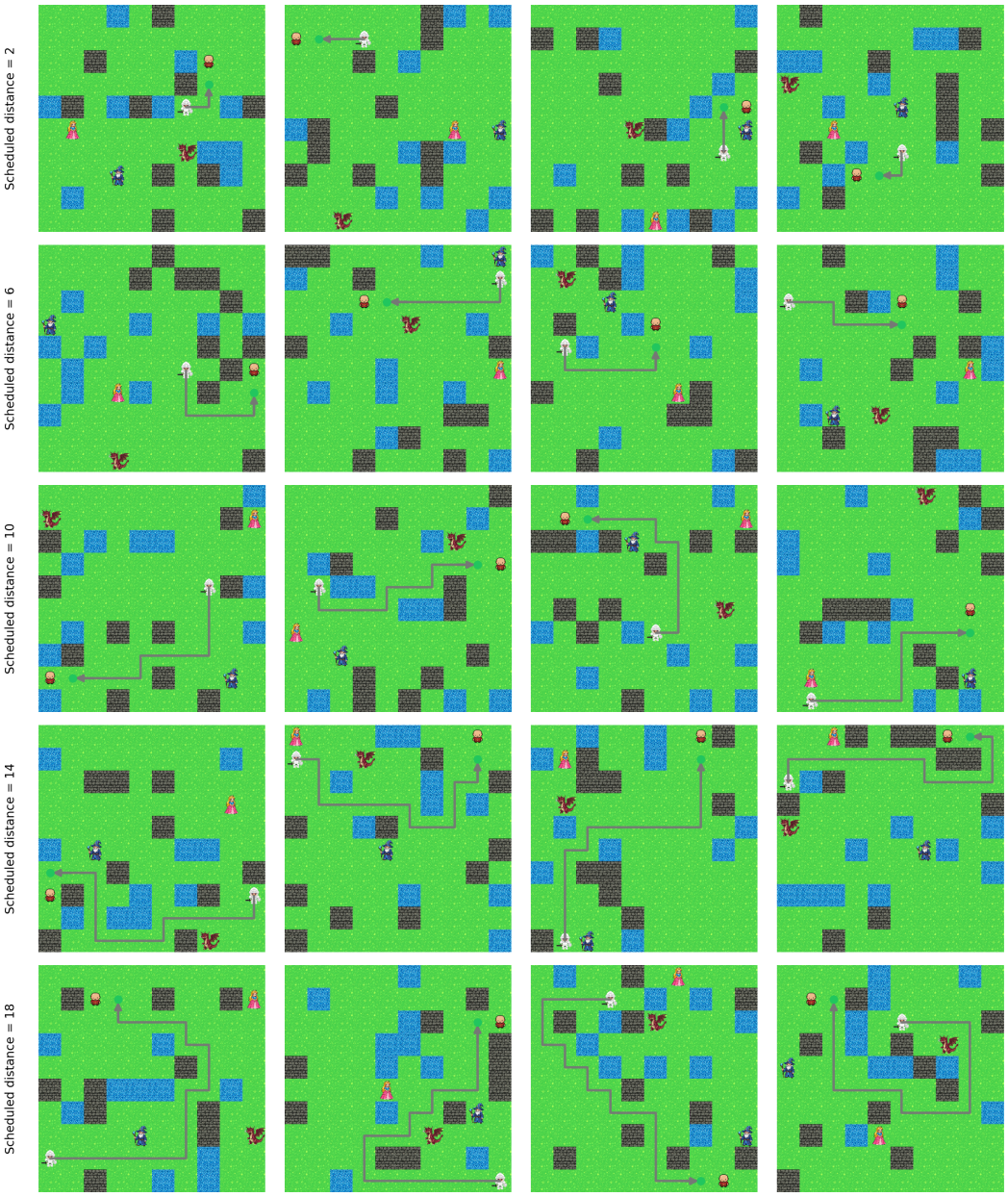


Figure 10: Randomly selected examples of our difficulty-graded Kingdom test configurations. Each row represents a different difficulty level based on the distance to travel to reach the target. In these examples, the language goal is "Reach the villager".

B COMPLETE RESULTS TABLES

B.1 SUBGOAL PERFORMANCE

Table 1: Complete results for all environments for increasing levels of difficulty, averaged across all possible tasks for an environment. Values represent mean success rate ± std, where higher is better.

Difficulty	DINO-WM	DINO-WM (O.S.)	LeWM	T.S.	HWM	LAGO (ours)	LAGO (O.S.)	GCBC
				TwoRoom				
35	100.0 ±0.0	100.0 ±0.0	20.0 ±12.6	100.0 ±0.0	100.0 ±0.0	100.0 ±0.0	100.0 ±0.0	100.0 ±0.0
75	10.0 ±9.5	100.0 ±0.0	0.0 ±0.0	90.0 ±9.5	50.0 ±7.1	100.0 ±0.0	100.0 ±0.0	100.0 ±0.0
115	10.0 ±9.5	100.0 ±0.0	0.0 ±0.0	0.0 ±0.0	44.0 ±7.0	100.0 ±0.0	100.0 ±0.0	100.0 ±0.0
155	0.0 ±0.0	100.0 ±0.0	0.0 ±0.0	0.0 ±0.0	16.0 ±5.2	100.0 ±0.0	100.0 ±0.0	100.0 ±0.0
200	0.0 ±0.0	100.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	100.0 ±0.0	100.0 ±0.0	100.0 ±0.0
				Kingdom				
2	97.0 ±1.0	100.0 ±0.0	10.0 ±1.7	98.0 ±0.8	83.4 ±2.2	100.0 ±0.0	100.0 ±0.0	89.7 ±2.5
6	29.0 ±2.7	100.0 ±0.0	2.0 ±0.9	78.0 ±2.4	10.7 ±1.8	97.0 ±1.0	100.0 ±0.0	55.9 ±4.1
10	10.0 ±0.5	88.0 ±1.9	0.0 ±0.0	15.0 ±2.1	1.0 ±0.6	84.0 ±2.2	98.0 ±0.9	29.7 ±3.8
14	0.0 ±0.0	46.0 ±2.9	0.0 ±0.0	0.0 ±0.0	0.3 ±0.3	39.0 ±2.9	76.0 ±2.5	9.7 ±2.5
18	0.0 ±0.0	15.0 ±2.1	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	8.0 ±1.6	39.0 ±2.9	1.4 ±1.0
				OGBench Cube				
0.08	80.0 ±4.0	99.0 ±1.0	34.0 ±4.7	75.0 ±4.3	86.0 ±3.5	85.0 ±3.6	94.0 ±3.4	70.0 ±12.6
0.12	24.0 ±4.3	57.0 ±5.0	6.0 ±2.4	32.0 ±4.7	43.0 ±5.0	60.0 ±4.9	74.0 ±6.2	35.0 ±10.7
0.2	3.0 ±1.7	44.0 ±5.0	0.0 ±0.0	9.0 ±2.9	7.0 ±2.6	55.0 ±5.0	46.0 ±7.0	45.0 ±11.1
0.3	0.0 ±0.0	20.0 ±4.0	0.0 ±0.0	0.0 ±0.0	6.0 ±2.4	29.0 ±4.5	32.0 ±6.6	45.0 ±11.1
0.4	0.0 ±0.0	9.0 ±2.9	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	11.0 ±3.1	20.0 ±6.0	40.0 ±10.9

B.2 LANGUAGE GROUNDING

Table 2: Language goal performance across environments. Values are mean success rate ± std, where higher is better.

Method	Success Rate (%)
Kingdom	
PE-Core	6.00 ±0.93
LIV	8.77 ±1.11
LIV (finetuned)	52.6 ±1.38
LAGO (ours)	97.0 ±1.0
OGBench Scene	
PE-Core	0.0 ±0.0
LIV	12.5 ±5.2
LIV (finetuned)	48.75 ±5.6
LAGO (ours)	55.31 ±2.36

B.3 PERFORMANCE PER NUMBER OF SUBGOALS

Table 3: Complete results of LAGO's subgoal-count ablation for each environment. TwoRoom is fixed at difficulty 155, and Kingdom at difficulty 10. Values are success rates in $[0, 1]$, where higher is better.

	Number of subgoals								
Environment	1	2	3	4	5	6	7	8	10
TwoRoom	0.00 ±0.00	0.92 ±0.05	0.98 ±0.02	**1.00 ±0.00**	**1.00 ±0.00**	**1.00 ±0.00**	**1.00 ±0.00**	**1.00 ±0.00**	**1.00 ±0.00**
Kingdom	0.00 ±0.00	0.18 ±0.09	0.40 ±0.15	0.70 ±0.10	0.78 ±0.13	0.74 ±0.14	0.80 ±0.11	0.78 ±0.12	**0.84 ±0.11**
Average	0.00	0.55	0.69	0.85	0.89	0.87	0.90	0.89	**0.92**

B.4 Performance per Cost Temperature

Table 4: Complete results of LAGO's cost function analysis for each environment. TwoRoom is fixed at difficulty 155, and Kingdom at difficulty 10. Soft-min is reported for temperatures $\tau \in \{0.001, 0.01, 0.03, 0.1, 0.3, 1\}$. Values are success rates in %, $\pm$ std, where higher is better.

Environment	Sum	Soft-Min (τ)					
		0.001	0.01	0.03	0.1	0.3	1
TwoRoom	50.0 $\pm$16.0	**100.0** $\pm$0.0	**100.0** $\pm$0.0	**100.0** $\pm$0.0	**100.0** $\pm$0.0	**100.0** $\pm$0.0	80.0 $\pm$13.0
Kingdom	0.0 $\pm$0.0	70.0 $\pm$14.0	**80.0** $\pm$13.0	60.0 $\pm$15.0	10.0 $\pm$9.0	0.0 $\pm$0.0	0.0 $\pm$0.0
Average	25.0 $\pm$8.0	85.0 $\pm$7.0	**90.0** $\pm$6.5	80.0 $\pm$7.5	55.0 $\pm$4.5	50.0 $\pm$0.0	40.0 $\pm$6.5

C HYPERPARAMETERS

C.1 LAGO

Table 5: Architectural hyperparameters of LAGO.

Parameter	Symbol	Value
Total parameters		15.0M
Vision & Language Encoders		
Patch token dimension	D_z	384
Language embedding dim.	D_L	768
Vision encoder backbone		DINOv2 ViT-S/14
Language encoder backbone		EmbeddingGemma-300M
ViT Predictor		
Depth		6
Number of heads		8
Conditioning dimension	D_c	64
Hidden dimension		448
Activation		GELU
Optimisation		
Optimizer		AdamW
Learning rate		3×10^{-4}
Batch size		128
Training steps		500 epochs
Early stopping patience		50 epochs

C.2 DINO-WM

Table 6: Architectural hyperparameters of DINO-WM.

Parameter	Symbol	Value
Total parameters		14.7M
Vision Encoder		
Patch token dimension	D_z	384
Vision encoder backbone		DINOv2 ViT-S/14
ViT Predictor		
Depth		6
Number of heads		8
Conditioning dimension	D_c	64
Hidden dimension		448
Activation		GELU
Optimisation		
Optimizer		AdamW
Learning rate		3×10^{-4}
Batch size		128
Training steps		500 epochs
Early stopping patience		50 epochs

C.3 TEMPORAL STRAIGHTENING

Table 7: Architectural hyperparameters of Temporal Straightening.

Parameter	Symbol	Value
Total parameters		17.9M
Vision Encoder		
Patch token dimension	D_z	384
Encoder backbone		DINOv2 ViT-S/14
ViT Predictor		
Depth		6
Number of heads		8
Conditioning dimension	D_c	64
Hidden dimension		448
Activation		GELU
Straightening Aggregator		
Output dimension		128
Hidden dimension		256
Depth		2
Straightening Loss		
Loss weight	λ_{str}	0.1
Optimisation		
Optimizer		AdamW
Predictor learning rate		3×10^{-4}
Projector/aggregator learning rate		1×10^{-5}
Batch size		128
Training steps		500 epochs
Early stopping patience		50 epochs

C.4 LEWM

Table 8: Architectural hyperparameters of LeWM.

Parameter	Symbol	Value
Total parameters		11.9M
Vision Encoder		
Patch token dimension	D_z	192
Encoder backbone		ViT-Tiny *
ViT Predictor		
Depth		6
Number of heads		8
Hidden dimension		256
Conditioning dimension	D_c	64
Activation		GELU
SIGReg		
Loss weight		0.09
Number of knots		17
Number of projections		1024
Optimisation		
Optimizer		AdamW
Learning rate		5×10^{-5} *
Batch size		128
Training steps		500 epochs
Early stopping patience		50 epochs

* We choose these values for LeWM based on the official manuscript and implementation.

C.5 GCBC

Table 9: Architectural hyperparameters of GCBC.

Parameter	Symbol	Value
Total parameters		15.0M
Vision & Language Encoders		
Patch token dimension	D_z	384
Language embedding dim.	D_L	768
Vision encoder backbone		DINOv2 ViT-S/14
Language encoder backbone		EmbeddingGemma-300M
ViT Predictor		
Depth		6
Number of heads		8
Hidden dimension		448
Conditioning dimension	D_c	64
Activation		GELU
Optimisation		
Optimizer		AdamW
Learning rate		3×10^{-4}
Batch size		128
Training steps		500 epochs
Early stopping patience		50 epochs
Frameskip		1

C.6 HWM

Table 10: Architectural hyperparameters of HWM.

Parameter	Symbol	Value
Total parameters		35.7M (sum of both levels)
Vision Encoder (shared, frozen)		
Patch token dimension	D_z	384
Encoder backbone		DINOv2 ViT-S/14
Level-1 Predictor (Conv)		
Depth		3
Channels		512
Kernel size		3
Activation		ReLU
Action conditioning		raw action (concat)
Level-2 Predictor (Conv)		
Depth		5
Channels		512
Kernel size		5
Activation		ReLU
Macro-action latent dim.	$D_z^{(2)}$	8
Action encoder		8–64–8 MLP
Level-2 Posterior		
Input dimension		25
Hidden dimension		32
Depth		2
Hierarchy		
Level-1 window length		15 steps
Level-2 macro-step skip		5
Level-2 macro-steps		5
Optimisation		
Optimizer		Adam
Learning rate		1.5×10^{-4} *
Batch size		128
Training steps		500 epochs
Early stopping patience		50 epochs
Frameskip		1

* The official implementation scales the configured base rate of 3×10^{-4} by batch size$/256$, after a linear warmup over the first 10% of steps.

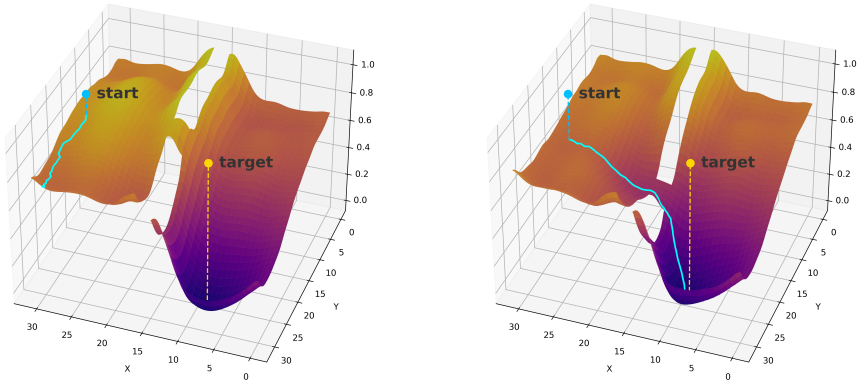
C.7 MPC AND CEM

Table 11: MPC and CEM hyperparameters used at planning time.

Parameter	Symbol	Value
Planning horizon		10
Number of samples		300
Number of elites		30
CEM iterations		10
Number of subgoals	K	5
Subgoal decay factor	λ	1
Subgoal decay factor (OGBench Cube)	λ	0.75
Temperature	τ	0.001

D VISUALIZING THE LOSS LANDSCAPES

Figure 11 visualizes the optimization landscape induced by the planning objective on a long-horizon TwoRoom task. With only the final goal as supervision, the landscape provides little useful gradient toward the goal, making optimization difficult. In contrast, LAGO's intermediate subgoals create a smooth sequence of lower-cost regions that forms a navigable valley toward the objective, providing CEM with progressively informative guidance throughout the rollout.

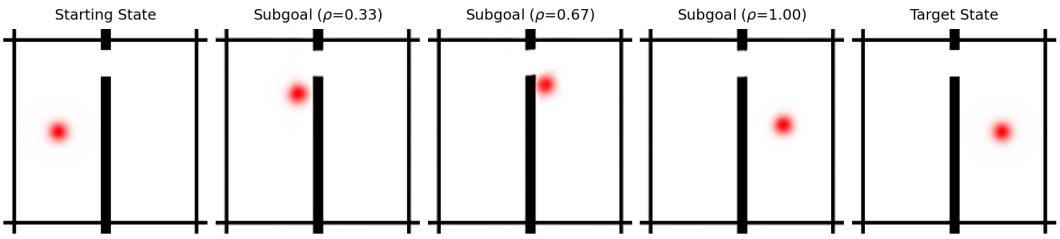


(a) Final goal image (b) LAGO

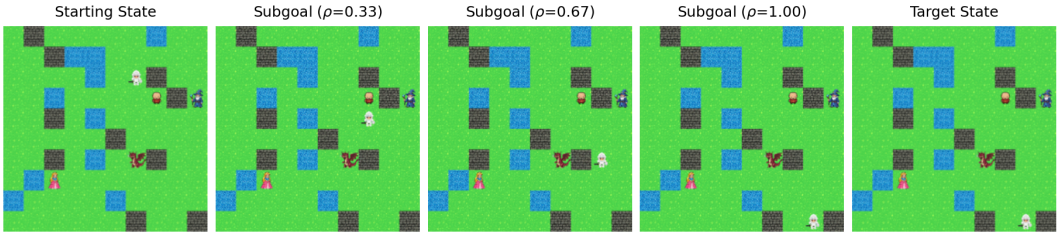
Figure 11: Loss landscapes for long-horizon TwoRoom tasks: single latent goal (left) vs. K language-predicted subgoals aggregated via J_{LAGO} (right). Planned paths are shown in blue. The single-goal landscape provides little gradient for the planner, yielding suboptimal trajectories. LAGO's subgoals carve a smooth valley that guides the CEM optimizer efficiently toward the objective.

E EXAMPLES OF DECODED SUBGOALS

We provide qualitative examples of decoded LAGO subgoals across all four environments. Each row shows the starting state (leftmost), the decoded subgoals at increasing completion values (middle), and the target state (rightmost), illustrating how LAGO decomposes a language goal into a smooth sequence of intermediate states.



(a) Example of predicted subgoals for the TwoRoom environment where the language goal passed to LAGO is "Place the red dot at position [0.732, 0.500]". The leftmost image shows the starting state, the rightmost image the target state, and middle images the decoded subgoals at different completion values.



(b) Example of predicted subgoals for the Kingdom environment where the language goal passed to LAGO is "Reach the bottom-right corner". The leftmost image shows the starting state, the rightmost image the target state, and middle images the decoded subgoals at different completion values.



(c) Example of predicted subgoals for the OGBench Cube environment where the language goal passed to LAGO is "Move the cube to position [0.425, -0.100]". The leftmost image shows the starting state, the rightmost image the target state, and middle images the decoded subgoals at different completion values.



(d) Example of predicted subgoals for the OGBench Scene environment where the language goal passed to LAGO is "Close the drawer". The leftmost image shows the starting state, the rightmost image the target state, and middle images the decoded subgoals at different completion values.

F Latent Space Visualization

To evaluate whether the learned latent spaces of the considered baselines capture sufficient spatial and semantic information, we train a lightweight decoder network to reconstruct the original pixels from the frozen latent embeddings. Since JEPA-based architectures do not inherently possess a generative component, this probing mechanism allows us to qualitatively assess the fidelity of the world model's internal states.

F.1 LEWM

TwoRoom

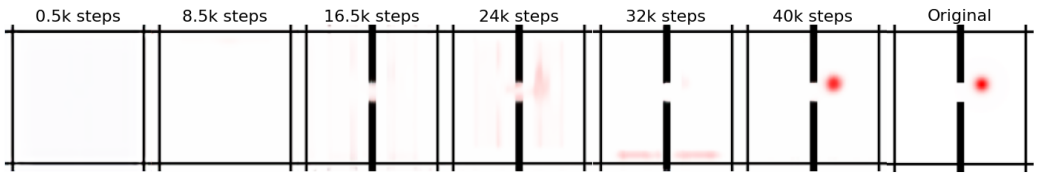


Figure 13: Decoder training progress in TwoRoom. We track a single held-out example across training and show its reconstruction at multiple checkpoints; reconstructions become increasingly accurate over time. The ground-truth observation is shown on the right.

Kingdom

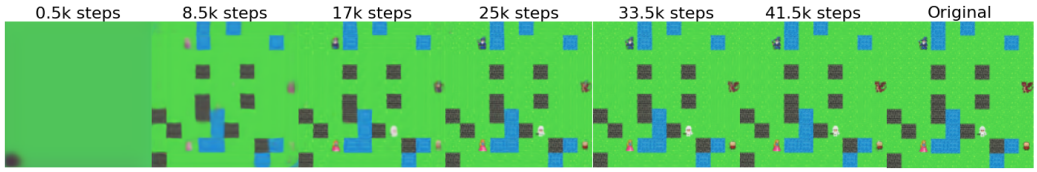


Figure 14: Decoder training progress in Kingdom. We track a single held-out example across training and show its reconstruction at multiple checkpoints; reconstructions become increasingly accurate over time. The ground-truth observation is shown on the right.

OGBench Cube



Figure 15: Decoder training progress in OGBench Cube. We track a single held-out example across training and show its reconstruction at multiple checkpoints; reconstructions become increasingly accurate over time. The ground-truth observation is shown on the right.

F.2 TEMPORAL STRAIGHTENING

TwoRoom

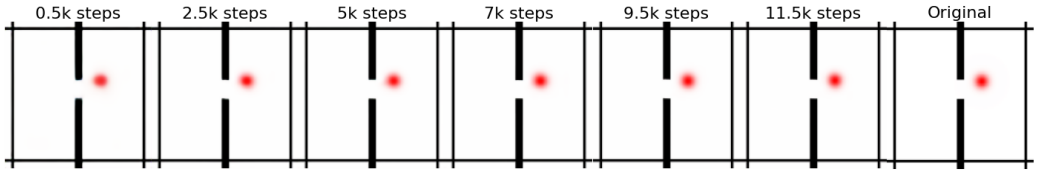


Figure 16: Decoder training progress in TwoRoom. We track a single held-out example across training and show its reconstruction at multiple checkpoints; reconstructions become increasingly accurate over time. The ground-truth observation is shown on the right.

Kingdom



Figure 17: Decoder training progress in Kingdom. We track a single held-out example across training and show its reconstruction at multiple checkpoints; reconstructions become increasingly accurate over time. The ground-truth observation is shown on the right.

OGBench Cube



Figure 18: Decoder training progress in OGBench Cube. We track a single held-out example across training and show its reconstruction at multiple checkpoints; reconstructions become increasingly accurate over time. The ground-truth observation is shown on the right.

G ROBO-DOPAMINE EVALUATION

While Robo-Dopamine's inference cost precludes its use in MPC-based planning, we assess the quality of its reward signal in isolation using the VOC (Value of Correlation) metric (Ma et al., 2024). This metric measures the Spearman rank correlation between the predicted reward and task progress, offering a direct comparison of reward monotonicity independent of planning performance.

As shown in fig. 19, LAGO produces a stronger and more monotonic reward signal than Robo-Dopamine across all OGBScene goals. On the representative goal "Lock the window", LAGO achieves a VOC of 0.97 versus 0.69 for Robo-Dopamine, whose reward remains near-flat through the middle of the episode and exhibits considerably more noise. This pattern holds on average across all goals, where LAGO consistently yields higher VOC scores. Taken together, these results suggest that LAGO not only satisfies the inference constraints of model-based planning, but also provides a higher-quality reward signal than a generative model with significantly more parameters.

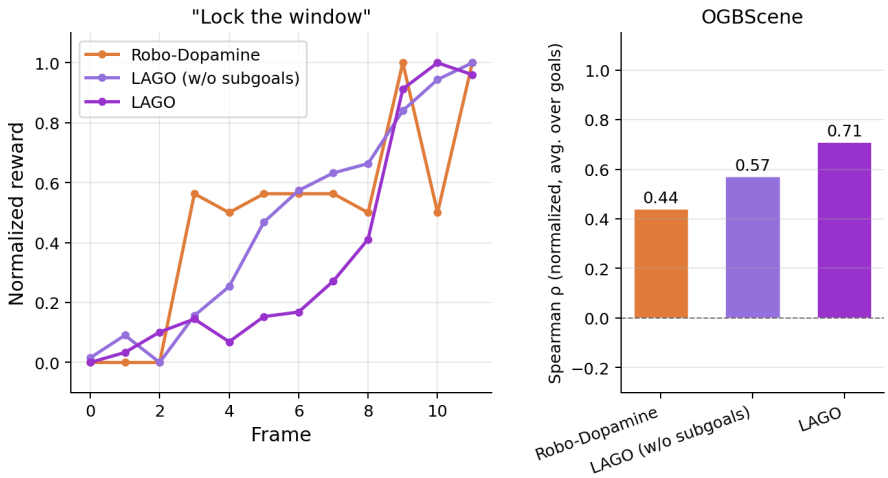


Figure 19: **Reward signal quality on OGBScene: LAGO vs. Robo-Dopamine.** *Left:* Normalized reward over frames for the goal "Lock the window". Despite a broadly increasing trend for both, Robo-Dopamine (VOC $= 0.69$) is noisier and assigns near-flat reward mid-task, while LAGO (VOC $= 0.97$) rises more consistently. *Right:* Average VOC across all OGBScene goals, where LAGO yields higher scores throughout.

H LAGOSET: Scaling Task Understanding Without Actions

LAGO learns dynamics from action-labeled transitions, but its subgoal predictor requires only video and language. To test whether the language-conditioned component benefits from data without actions, we train a LAGO model on interaction and demonstration data from LIBERO90 (Liu et al., 2023), but augment the demonstration stream with LAGOSET, an action-less video-language corpus we assemble from a collection of larger demonstration datasets. We call the resulting model LAGO++, which differs from LAGO only in its subgoal supervision. We then evaluate whether scaling the language stream leads to increased robustness to out-of-domain instructions on held-out LIBERO-90 starting configurations.

H.1 Dataset and Training

LAGOSET pools 310k language-annotated clips from BridgeData V2 (Walke et al., 2023), DROID (Khazatsky et al., 2025), AgiBot World (AgiBot-World-Contributors et al., 2025), and Language-Table (Lynch et al., 2022), where a clip is one camera view of one episode. The corpus spans about 165k episodes and 57.5M frames. Only video and instructions are used; actions are discarded. Language-Table clips retain their human annotations, while clips from the other sources are annotated with four instructions each, generated by an image-conditioned paraphraser (Qwen3.5-4B (Qwen Team, 2026)), of which one is sampled at each training visit. With LAGOSET, we simulate the difference in size between action-labeled robotic data and action-less demonstration data (Miech et al., 2019; Grauman et al., 2022; Chen et al., 2026).

H.2 Rewording Benchmark

The benchmark consists of 20 LLM-authored rewordings for each of 18 LIBERO-90 instructions, ranging from synonym substitutions to descriptions of the outcome only. Rewordings within cosine similarity 0.9 (EmbeddingGemma) of any LAGOSET instruction are removed, leaving 289. Each rewording is assigned to one of four bins by its distance $(1 - \cos)$ to the original instruction; Table 12 gives examples.

Evaluation uses held-out initial configurations (five per task, 90 in total). Given a rewording, the model predicts the final subgoal $\tilde{z}_{\rho=1}$ in DINOv2 latent space, which is scored as correct if its nearest candidate in latent MSE is the final state of the instructed task. Candidates are the final and halfway

states of all tasks in the same scene. Restricting candidates to the same scene keeps the test focused on instruction understanding, since final states of other scenes would be trivially far. The halfway states penalize predictions that stall part-way through the task. With about nine candidates per query (after filtering for leakage with LAGOSET as mentioned earlier), chance accuracy is $\approx 12\%$, while retrieving from the real final frame instead gives a ceiling of 100%. Accuracy is averaged over each instruction's rewordings and then over the 18 instructions.

Table 12: Example rewordings from each distance range, with their EmbeddingGemma distance to the original instruction.

Distance	"put the black bowl on the plate"	"put the moka pot on the stove"
0.01–0.06	place the black bowl on the plate (0.02)	place the moka pot on the stove (0.03)
0.06–0.12	put the black bowl on the dinner plate (0.06)	put the moka pot on the cooktop (0.07)
0.12–0.19	put the dark bowl on the dish (0.14)	put the espresso maker on the stove (0.16)
0.19–0.50	take the dark bowl and lower it onto the flat dish (0.22)	lift the espresso pot and rest it on the burner (0.30)

H.3 Results

Table 13: Final-subgoal retrieval accuracy (%) on LIBERO-90 under instruction rewordings, binned by distance to the original instruction. Columns give trainable parameters; LAGO++ adds action-less LAGOSET language. Mean $\pm$ s.e. over instructions; higher is better.

Distance	Rewordings	LAGO (15M)	LAGO++ (15M)	LAGO++ (30M)	LAGO++ (59M)
Original	–	100.0 $\pm$0.0	100.0 $\pm$0.0	100.0 $\pm$0.0	100.0 $\pm$0.0
0.01–0.06	73	93.4 $\pm$3.8	98.4 $\pm$1.0	98.6 $\pm$1.0	98.8 $\pm$1.0
0.06–0.12	72	92.0 $\pm$2.5	97.4 $\pm$1.2	98.7 $\pm$0.9	98.4 $\pm$1.2
0.12–0.19	72	77.9 $\pm$4.6	85.8 $\pm$3.6	86.9 $\pm$3.5	91.8 $\pm$3.0
0.19–0.50	72	56.9 $\pm$8.5	76.1 $\pm$5.9	85.2 $\pm$4.6	87.4 $\pm$3.8
All	289	79.2 $\pm$4.4	89.7 $\pm$2.3	91.5 $\pm$1.9	94.6 $\pm$1.3

Table 13 reports accuracy per distance bin. Every model grounds the original instructions perfectly, so differences appear only under rewording. Accuracy falls as rewordings move away from the original, yet even LAGO, which has only seen the LIBERO-90 instructions, stays well above chance on the most distant rewordings (56.9% against $\approx 12\%$). LAGO++ (15M) has the same size as LAGO and differs only in its language data. LAGOSET contains no actions, and rewordings close to its instructions were filtered out, so its gains (79.2% to 89.7% overall, largest on the most distant rewordings) show that LAGO can improve its task understanding from action-less demonstrations. Larger LAGO++ models improve further, again mostly on distant rewordings, so the most robust models combine action-less data with more capacity. Together, these results suggest that, on LIBERO-90, LAGO can use action-less language data to become more robust to rephrased instructions without additional action supervision.