DASWM: DISENTANGLING DYNAMIC AND STATIC FACTORS FOR LATENT WORLD MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce DASWM, a world model representation that softly disentangles dynamic, action-relevant factors from static ones, allowing the allocation of prediction capacity to the dynamic parts. Latent world models make visual planning efficient by encoding observations (e.g. pixels) into compact latent representations. However, not all encoded information needs to be subject to prediction. We show in this work that recently proposed JEPA-like latent world models fail when static visual factors are introduced. We identify the cause of this failure as static-dynamic entanglement in the representation. Our DASWM latent world model encodes each observation as a sequence of tokens, ordered with respect to their dynamic responsiveness. This leads to a representation where dynamic information lies in the early tokens, and static information in the latter ones, minimizing the entanglement. We evaluate DASWM on background-modified versions of several datasets and demonstrate clear improvements in planning, in addition to other benefits such as flexible budget planning at test-time.

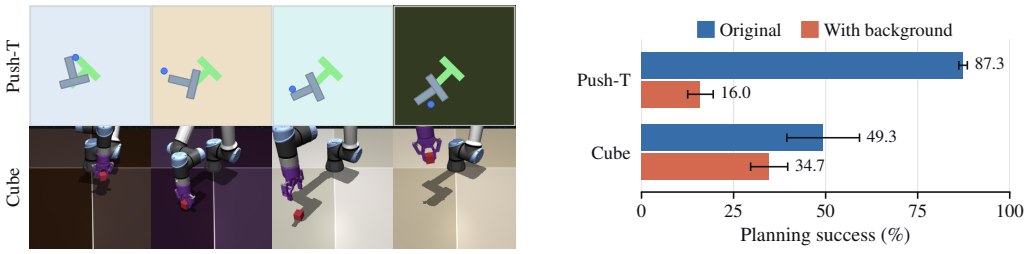
1 INTRODUCTION

Physical intelligence requires the ability to imagine the consequences of certain interventions in an environment. This allows an agent to plan its actions based on imagination instead of expensive execution (Hafner et al., 2019; 2025; Hansen et al., 2024). Recent latent world models, such as LeWM (Maes et al., 2026b), make this possible by learning an action-conditioned predictor that modifies a state based on a given action and an observation. Imagining the future and planning in a compressed latent space can make this process more efficient compared to observation space (e.g. pixels). However, recent embedded world models (Assran et al., 2025; Zhou et al., 2025; Sobal et al., 2025; Terver et al., 2025; Maes et al., 2026b) do not consider that the environment usually consists of both dynamic elements, which are affected by actions, and static elements, which are not. In this work, we introduce a representation that allows soft factorization of static and dynamic elements and enables the predictor to efficiently allocate its capacity to the dynamic parts only.

LeWM (Maes et al., 2026b) learns the encoder and the predictor end-to-end from pixels and prevents collapse with the SIGReg regularizer (Balestriero & LeCun, 2025), eliminating the need to use asymmetric encoders or stop gradient operations. It provides reasonable planning performance in several 2D and 3D simulated environments in which a single action-controlled agent moves itself and potentially other objects. However, we find that its success is heavily tied to the simplicity of the evaluation environments, which contain static elements that stay constant over the full dataset (e.g a simple white background). We augment datasets such as Push-T (Chi et al., 2023) and OGBench-Cube (Park et al., 2025) with episode-level varying background colors while preserving their trajectories and physics (Fig. 1a) and train LeWM on both static and varying background setups. When varying the backgrounds in such a way, the model quality drastically degrades (Fig. 1b).

We conduct an analysis to identify the cause of the failure and find that it is due to a *static-dynamic entanglement in the representation*. For this, we track how the latent representation changes between two timesteps over several identical physical transitions rendered with different backgrounds and decompose the variance of the latent change into parts explained by the physical transition only, the background only, and the combination of both. As shown in Fig. 2, we find that in the varying background case, most of the variance is explained by the combination of the transition and background

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.



(a) Dataset samples.

(b) Planning success.

Figure 1: (a) We introduce static variation (e.g. varying background color over episodes) on episode level, here shown on examples from Push-T (Chi et al., 2023) and Cube (Park et al., 2025) datasets. LeWM planning success falls after training with this variation. (b) Results of LeWM trained on the original data and our variant with varying background and evaluated on the same filtered Moderate start–goal pairs, showing a drastic performance decrease; mean ± s.d. over three evaluation seeds.

and not the transition alone. Thus, the same motion is represented differently under each background, signifying an entanglement between the static background and the transitions.

Therefore, we propose Dynamic-and-Static World Models (DASWM), a latent world model paradigm that softly disentangles action-relevant from static information by learning a set of tokens ordered by dynamic responsiveness, where early tokens carry the information that changes under the actions, and later tokens carry more temporally invariant information. This allows the transition model to focus on a subset of tokens and leads to generally higher success rates under (now disentangled) static shifts.

To induce the ordering during training, we split a token set of size K into a prefix of k tokens, where k is sampled randomly per transition. Then, only the prefix tokens are passed through the predictor to obtain the prefix of the next state, while the remaining suffix tokens are simply copied. We carefully design an objective with different components that allow DASWM to minimize the static-dynamic entanglement by placing the dynamics in early tokens and static information in later ones. The early tokens are supervised with the next-state prediction and inverse dynamics loss, while the later tokens are supervised with a persistence loss that encourages invariance. Additionally, to avoid representation collapse, we adapt the SIGReg regularizer (Balestriero & LeCun, 2025) to our nested latent representation by applying it to concatenations of tokens for all possible prefixes.

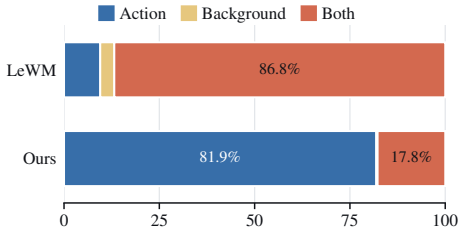


Figure 2: We report the decomposition of the explained variance of the latent changes across 500 episodes. At the top, we see that for LeWM trained on Push-T background, the action and background together explain most of the variance, signaling an entanglement. DASWM reduces the entanglement and allows the actions to control the latent change.

We evaluate DASWM on our proposed versions of background varying datasets (TwoRooms-B, Push-T-B, and OGBench-Cube-B), and OGBench Scene, which already has statically varying components across episodes. DASWM achieves consistent improvements against the original LeWM and an inverse-dynamics enhanced baseline across all datasets. It improves planning success rate by up to 33.8% on some datasets under background variation. We demonstrate how DASWM minimizes the static-dynamic entanglement, and additionally allows for flexible-budget planning during inference as a side benefit. In summary, our contributions are:

- We create datasets that expose a failure mode of recently proposed latent world models under static information variation and identify the reason for the failure as static-dynamic representation entanglement.
- As a solution, we propose DASWM, a latent world model representation paradigm that learns a set of tokens that is ordered from dynamic to static content.
- We evaluate DASWM and demonstrate how it reduces the entanglement issue and achieves consistent improvements over several datasets with varying background elements.

2 RELATED WORK

Latent world models for visual planning. Latent world models replace pixel prediction with future-state prediction in a learned feature space (Hafner et al., 2019; 2025; Hansen et al., 2024). The main difference between the methods in this line of work is the latent encoder and the anti-collapse technique. DINO-WM and V-Jepa 2 use frozen foundation models (Oquab et al., 2024; Assran et al., 2025)as the encoder representation and they train a predictor on top(Zhou et al., 2025). LeWM instead learns the encoder and the dynamics model end-to-end from pixels (Maes et al., 2026b). Training the encoder and predictor jointly can lead to a representation collapse, therefore they use the SIGReg loss introduced by LeJEPA to prevent collapse (Balestriero & LeCun, 2025). PLDM trains latent dynamics with a JEPA objective for reward-free planning (Sobal et al., 2025). Delta-JEPA decodes the executed action from latent differences to prevent collapse (Zhang et al., 2026b). DINO-Bisim also uses DINOv2 as an encoder and aims to improve robustness to test-time background changes and distractors (Toso et al., 2026). Different from previous works, we focus on improving latent world modeling by proposing a design that inherently induces dynamic-static disentanglement.

Sparse and budget-flexible visual representations. There exists some world model methods that follow also propose budget-flexible tokenization methods, but they use them to address different problems. Sparse Imagination accelerates planning by predicting random subsets of spatial patch tokens with randomized grouped attention (Chun et al., 2026). CompACT learns a compact discrete tokenizer for planning with a fixed small token set (Kim et al., 2026). Outside the area of world modeling, Matryoshka multimodal models and the Matryoshka Query Transformer learn nested visual token sets for flexible vision-language inference (Cai et al., 2025; Hu et al., 2024). Nested dropout learns ordered representations by randomly truncating latent units during training (Rippel et al., 2014). FlexTok and Semanticist apply nested dropout to causally ordered tokens to obtain coarse-to-fine sequences of flexible length for reconstruction(Bachmann et al., 2025; Wen et al., 2025). These methods motivate flexible token computation, but do not learn an ordered state whose prefixes are explicitly supervised as action-relevant transitions. Our tokens are not selected spatial patches: token order defines a nested update hierarchy.

Structured and disentangled dynamics. Object-centric methods represent scenes as sets of entities, from Slot Attention (Locatello et al., 2020) and structured world models (Kipf et al., 2020) to planning systems such as OP3 and PlaySlot (Veerapaneni et al., 2020; Villar-Corrales & Behnke, 2025). Recent work explicitly separates content from motion or static from dynamic object features (Zhang et al., 2026a; Wang et al., 2025). In contrast, DASWM does not assume object masks, object-bound slots, known nuisance variables, or a fixed content–motion partition. Its separation is budget-relative: information enters the predicted update through nested prefixes as more transition capacity is requested.

3 DASWM: FACTORIZING STATIC AND DYNAMIC INFORMATION

We aim to design a latent representation that disentangles action-dependent dynamic information from static information. Therefore, we propose DASWM, a latent world model paradigm that encodes each observation into a set of tokens ordered by their action responsiveness. Fig. 3 provides an overview of the method.

3.1 BACKGROUND: LATENT PREDICTIVE WORLD MODELS

Let o_t be an observation, a_t an action block, E_θ an encoder, and F_ϕ a latent dynamics model. A JEPA world model encodes $z_t = E_\theta(o_t)$ and predicts

$$\widehat{z}_{t+1} = F_\phi(z_t, a_t), \qquad \mathcal{L}_{\text{pred}} = \|\widehat{z}_{t+1} - z_{t+1}\|_2^2. \tag{1}$$

This objective alone can be satisfied with a constant representation. LeWM prevents collapse using the SIGReg regularizer, which matches random one-dimensional projections of the embeddings to a standard Gaussian (Balestriero & LeCun, 2025; Maes et al., 2026b). At test-time, given a goal observation o_{goal}, the cross-entropy method (CEM) (Rubinstein, 1999) is used to optimize an action sequence $(a_t)_{t=1}^T$ by minimizing the latent distance between an imagined terminal state $\hat{Z}_T$ and the encoded goal state $Z_{\text{goal}} = E_\theta(o_{\text{goal}})$.

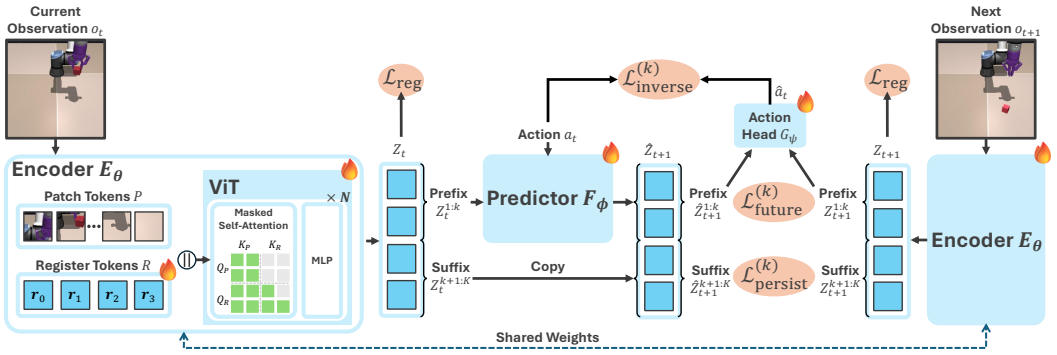


Figure 3: **Overview of DASWM**. During training, a shared causal register encoder E_θ encodes the current o_t and future observation o_{t+1} into ordered token sets Z_t and Z_{t+1}, respectively. For a sampled update budget k, the predictor F_ϕ updates the k prefix tokens $Z_t^{1:k}$ to $\widehat{Z}_{t+1}^{1:k}$, conditioned on the action a_t, while a persistence loss discourages unnecessary change in the remaining suffix. An auxiliary action head enforces that the temporal prefix difference retains action information.

3.2 DASWM ARCHITECTURE

Given observation o_t, our encoder produces a set of K tokens (Fig. 3)

$$Z_t = E_\theta(o_t) = [z_t^1, \ldots, z_t^K], \qquad z_t^j \in \mathbb{R}^d. \tag{2}$$

We adopt the register transformer encoder architecture from (Bachmann et al., 2025), which consists of a ViT with K learnable registers. Register tokens attend to image patches but not vice versa. We further employ causal inter-register attention to create an asymmetry, discouraging token redundancy and inducing an ordering. We use $K = 4$ tokens of dimension $d = 48$, matching the 192 channels of the single LeWM embedding.

We adopt the same predictor architecture used by LeWM (Maes et al., 2026b), which consists of 6 transformer blocks with frame-causal temporal attention, with the following modifications. We employ token-level causal attention like in the encoder and add token-level positional embeddings in addition to temporal positional encoding.

3.3 TRAINING PROCEDURE

For every transition during training, we choose a set of k prefix tokens $Z_t^{1:k}$, and we denote the remaining $Z_t^{k+1:K}$ as the persistence tokens . The predictor F_ϕ receives only the prefix tokens $Z_t^{1:k}$ and action block a_t and updates the prefix only, while the persistence tokens are simply copied

$$\widehat{Z}_{t+1}^{1:k} = F_\phi(Z_t^{1:k}, a_t), \qquad \widehat{Z}_{t+1}^{k+1:K} = Z_t^{k+1:K}. \tag{3}$$

We first warm up with the complete budget $k = K$ for 10,000 training steps. Afterwards, we sample $k \in \mathcal{K}$ uniformly for each mini-batch, where $\mathcal{K} = \{1, 2, 4, \ldots, K\}$.

To induce dynamic-static factorization among the tokens, we design the objective to consist of the following losses:

Future prediction Loss. The predictor learns to estimate the next state's prefix $\widehat{Z}_{t+1}^{1:k}$, given the current prefix $Z_t^{1:k}$ and action a_t, to match the future ground truth $Z_{t+1}^{1:k}$

$$\mathcal{L}^{(k)}_{\text{future}} = \frac{1}{kd} \left\| \widehat{Z}_{t+1}^{1:k} - Z_{t+1}^{1:k} \right\|_2^2. \tag{4}$$

Inverse Dynamics Loss. Future prediction alone may still favor visual variation over action-induced variation. Therefore, inspired by Delta-JEPA (Zhang et al., 2026b), we add an inverse dynamics loss that encourages the encoded transition of every sampled prefix to retain action information. An auxiliary action head G_ψ regresses the action block from the temporal difference of the prefix as

$$\widehat{a}_t = G_\psi\big(Z_{t+1}^{1:k} - Z_t^{1:k}\big), \qquad \mathcal{L}^{(k)}_{\text{inverse}} = \|\widehat{a}_t - a_t\|_2^2. \tag{5}$$

The head acts on encoded observations rather than on the predictor output, so it shapes the encoder representation directly. It is discarded at inference.

Persistence Loss. For the persistence tokens, we require them to stay invariant from one transition to another. Therefore, we propose a persistence loss that penalizes this change

$$\mathcal{L}_{\text{persist}}^{(k)} = \frac{1}{(K-k)d} \left\| Z_{t+1}^{k+1:K} - Z_t^{k+1:K} \right\|_2^2, \tag{6}$$

with the term omitted when $k = K$. This loss is the most critical component for inducing the specialization and dynamic ordering of the tokens. The weight of this loss modulates the dynamic energy distribution among tokens (Fig. 8, Section 5).

Regularization Loss. An essential component of latent world models is the anti-collapse regularization mechanism. In our case, we also utilize the SIGReg loss (Balestriero & LeCun, 2025). However, we adapt it to our nested latent representation by applying it to concatenations of tokens for all possible prefixes to maintain the diversity across all dynamic sets. We show in the ablations that this choice is critical for all tokens to maintain a healthy variance. The loss is defined as

$$\mathcal{L}_{\text{reg}} = \frac{1}{|\mathcal{K}|} \sum_{b \in \mathcal{K}} \text{SIGReg}(\text{concat}(Z_t^{1:b})). \tag{7}$$

The complete training objective is the weighted sum of all loss terms above

$$\mathcal{L} = \mathbb{E}_{k \sim \text{Unif}(\mathcal{K})} \left[\mathcal{L}_{\text{future}}^{(k)} + \lambda_{\text{persist}} \mathcal{L}_{\text{persist}}^{(k)} + \lambda_{\text{inverse}} \mathcal{L}_{\text{inverse}}^{(k)} \right] + \lambda_{\text{reg}} \mathcal{L}_{\text{reg}}, \tag{8}$$

where the weights λ_{persist}, λ_{inverse}, λ_{reg} are hyperparameters. By default, we use $\lambda_{\text{reg}} = 0.09$, and choose the rest for every dataset via a grid search. The persistence and inverse dynamics loss weights govern how strongly transition information is concentrated in early tokens. We study their interaction rather than treating either value as universal.

3.4 Planning at Inference

At inference, we choose an update budget k and combine the predicted prefix with the persistent suffix to form the next latent state:

$$\widehat{Z}_{t+1}^{(k)} = \text{concat} \left[\widehat{Z}_{t+1}^{1:k},\ Z_t^{k+1:K} \right]. \tag{9}$$

We use the cross-entropy method (CEM) (Rubinstein, 1999) to plan in this state. Appendix A.2 defines the planning objective and gives the complete planner configuration.

4 Experiments

After defining the datasets and protocol, we first isolate the failure caused by persistent visual variation and analyze its cause (Sec. 4.3). We then test whether the proposed DASWM improves full-budget planning across environments (Sec. 4.4) and provide further analysis, e.g., whether that improvement is accompanied by less background-conditioned latent change in Sec. 5.

4.1 Datasets

Controlled background interventions. We construct Push-T Background, Cube Background, and TwoRoom Background from Push-T (Chi et al., 2023), OGBench-Cube (Park et al., 2025), and TwoRoom (Sobal et al., 2025), respectively. Each variant preserves the source trajectories, actions, and simulator setup, but assigns one of 20 background colors to each episode. The color remains fixed within an episode. These datasets therefore introduce persistent visual variation without changing the underlying control problem. Their construction and palettes are documented in Appendix A.3.

Naturally heterogeneous environment. We also evaluate on the OGBench-Scene (Park et al., 2025), which contains an effector, cubes, a drawer, and a window. Unlike our controlled interventions, Scene already combines moving objects with components that remain fixed for many transitions. It tests whether the same representation is useful beyond color-based background variation.

4.2 Experimental Setup

Our default DASWM model uses four 48-dimensional tokens and samples $k \in \{1, 2, 4\}$. Training begins with 10,000 full-budget iterations before prefix-budget sampling is activated. We use $\lambda_{\text{reg}} = 0.09$ throughout and select the persistence and inverse-loss weights per dataset. The pairs $(\lambda_{\text{persist}}, \lambda_{\text{inverse}})$ are $(0.1, 0.1)$ on OGBench-Scene, $(6, 0.3)$ on Cube Background and TwoRoom Background, and $(3, 8)$ on Push-T Background. Appendix A.1 gives the full configuration details.

Baselines. We report the original LeWM objective when diagnosing the failure and also train LeWM with the same inverse-dynamics supervision used by DASWM. The latter is the stronger baseline for the main planning comparison, separating the benefit of the ordered token state from the benefit of action supervision alone.

Evaluation. We evaluate the models' planning in closed-loop simulation using the CEM algorithm (Rubinstein, 1999), which given a goal observation, optimizes a sequence of actions using the world models' rollouts. The selected actions are then executed in the simulator. We report the success rate as the main metric. We follow the success criteria defined in Sun et al. (2026), which exclude episodes where start-goal pairs are already successful and use their Moderate and Strict success criteria for TwoRoom Background, Push-T Background, and Cube Background. We apply the same initial-success filter to OGBench-Scene. Each result averages 50 episodes for each of three evaluation seeds. Appendix A.2 specifies the success rules and planner settings.

4.3 Identifying Static/Dynamic Entanglement

Background Intervention. We train the default LeWM setup once on the original Push-T and OGBench-Cube datasets, and again on their background-varying versions and evaluate the planning performance to see the effect of the background intervention.

Failure Detection. We see in Fig. 1b that the intervention alone sharply reduces LeWM planning success on both environments. We inspect further by training probes on the representation of each model to predict planning-relevant factors, such as positions and orientations. As shown in Fig. 4, the prediction errors increase when predicting the pusher position, block position, and block orientation from the Push-T state. The representation is therefore failing to retain this information with different backgrounds.

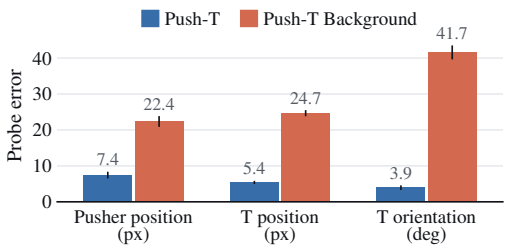


Figure 4: **State Prediction Probes on Push-T**. We show the error of LeWM trained on Push-T and on Push-T Background; whiskers are 95% bootstrap intervals. When varying static backgrounds, LeWM's dynamics prediction deteriorates significantly, causing planning failures.

Finding the Cause. To better understand the failure, we analyze the latent representation by decomposing variance of its change to see what parts of the input explain it. We setup 500 held-out physical transitions under all 20 backgrounds on Push-T and compute the latent change $\Delta z_{i,b} = E_\theta(o_{t+1}^{i,b}) - E_\theta(o_t^{i,b})$, where i indexes a physical transition and b its background (see Appendix A.4 for details). The decomposition computation in Fig. 2 attributes 9.5% of the squared variation to the physical transition, 3.7% to the background effect, and 86.8% to the transition-background interaction. This suggests that for every background the encoder learns different dynamic representations which we identify as static-dynamic entanglement. Notably, in the same figure we observe our proposed DASWM reduces this variance explained by the interaction and thereby reducing the entanglement.

4.4 Planning Performance Across Environments

Tab. 1 reports planning success across all datasets, averaged over 3 evaluation seeds. DASWM outperforms both LeWM and the inverse-dynamics-enhanced LeWM baseline on every dataset and

Table 1: **Planning Success across Benchmarks**. Following prior works, we report the mean over three evaluation seeds. DASWM outperforms LeWM consistently across all benchmarks with large improvements especially on the robotic simulation datasets OGBench-Scene and Cube Background. Detailed breakdowns including standard deviations are provided in Appendix A.5.

	Scene	TwoRoom Background		Cube Background		Push-T Background	
Method	Success	Moderate	Strict	Moderate	Strict	Moderate	Strict
LeWM	24.7	25.3	8.7	48.0	17.3	16.0	20.0
LeWM + $\mathcal{L}_{\text{inverse}}$	36.7	98.7	87.3	80.7	55.3	72.7	42.0
DASWM (Ours)	**45.3**	**100.0**	**93.3**	**95.3**	**74.0**	**74.7**	**42.7**

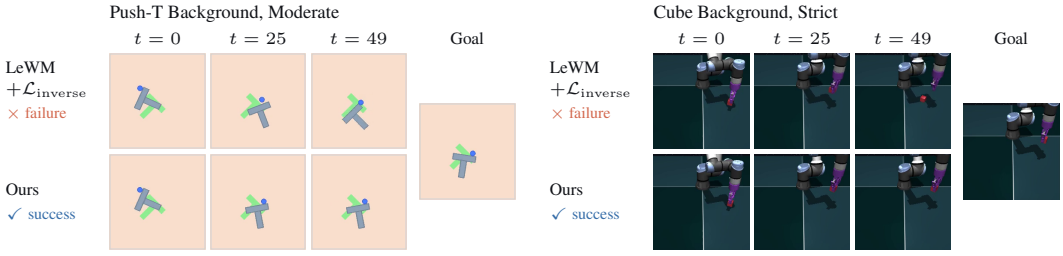


Figure 5: **Executed Plans on Push-T and Cube**. The frames show the states resulting from a plan that was optimized using the baseline (top) and DASWM (bottom) and the right column is the goal observation. We highlight episodes where DASWM leads to better plans.

success criterion, showing that the gains stem from the disentangled token ordering itself, not merely from the shared inverse-dynamics supervision.

Improvements are largest where static-dynamic entanglement is strongest. On Cube Background, Moderate success rises from 80.7% to 95.3% and Strict success from 55.3% to 74.0%. On Scene, whose static structure (drawer, window, effector) arises naturally rather than from an injected background, success improves from 36.7% to 45.3%, indicating the benefit generalizes beyond the controlled color interventions. TwoRoom Background Strict success improves from 87.3% to 93.3%. Push-T Background shows more modest gains ($+2.0$ points Moderate, $+0.7$ Strict), suggesting its entanglement is harder to remove through token reordering alone.

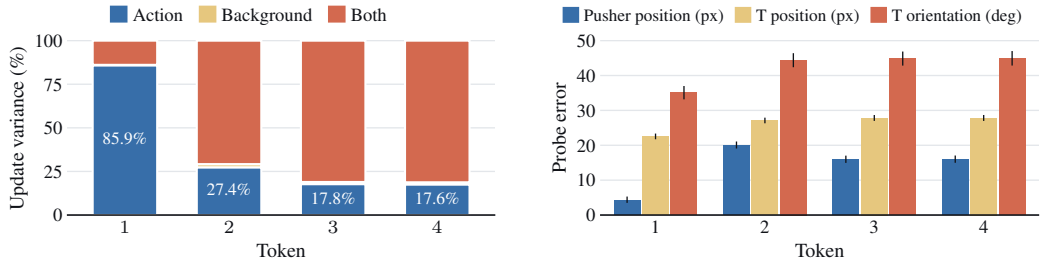
These results support the overall claim: ordering tokens by dynamic responsiveness lets the predictor allocate capacity to the action-relevant subset of the representation, rather than absorbing static variation into the same latent used for control. Fig. 5 shows planned episodes where this distinction determines success or failure; Appendix A.6 provides additional examples.

5 Analysis and Ablations

We next examine how information is distributed across the tokens learned by DASWM to see if it exhibits the intended ordering. We then isolate the roles of different design components and ablate sensitive parameters. Finally, we demonstrate the multi-budget planning advantage of DASWM.

What does each token learn? In Fig. 6a, we apply the variance decomposition independently to each token in DASWM on Push-T. We see that the variance of the first token is mostly explained by the transition, and this effect falls going to the right, where background information dominates more. Additionally we train probes on each token individually to predict planning relevant factors and report the errors in Fig. 6b. We see that the early token achieves the lowest error, showing that it specializes in carrying the dynamics relevant information.

Role of different design choices. We report in Tab. 2 the planning performance of DASWM resulting from removing the core design objectives on OGBench-Scene. Removing the inverse loss leads to the greatest drop in performance. Remove the persist loss also leads to a clear degradation in planning performance. By removing the prefix sampling, and simply using all tokens for future



(a) Per-token variance decomposition

(b) Per-token physical-state probes.

Figure 6: Per-token analysis of DASWM on Push-T Background. (a) Each bar partitions one token's explained variance into action, background, and the interaction of components; bar labels give the physical-transition share. (b) Median error of nonlinear probes for the pusher position, T position, and T orientation from each 48-dimensional token, on original-background Push-T frames. We see that the earliest token is mostly explained by the action (left) and it achieves the lowest probe error (right), while other tokens gradually contribute less to the dynamics relevant roles.

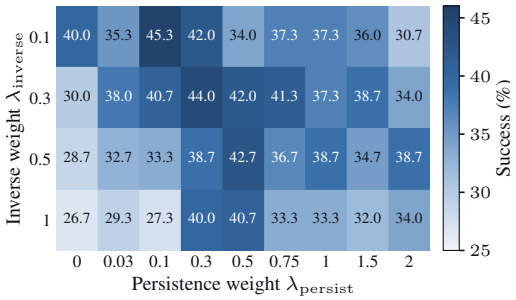


Figure 7: Success (%) on OGBench-Scene over a joint sweep of the persistence and inverse weights, with all other choices fixed. We see that the both weights are not fully independent. There eixsts a region $\lambda_{\text{inverse}} \in [0.1 - 0.3] \times \lambda_{\text{persist}} \in [0.1 - 0.5]$ that provides reliable models.

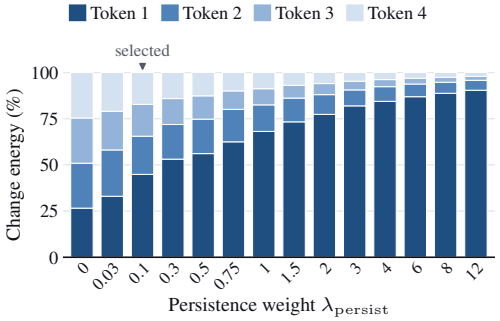


Figure 8: **Allocation of Change Energy.** We analyze how much each of DASWM's four tokens changes over time on Scene as the persistence weight varies. The larger λ_{persist}, the stronger is the concentration of change energy in the first tokens, indicating a disentanglement of dynamic and static information.

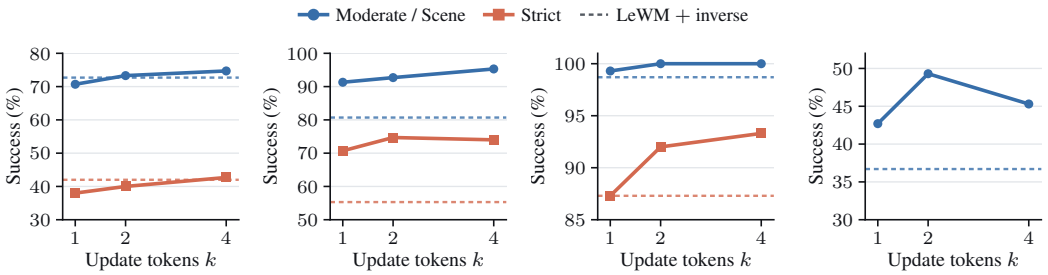
prediction during training, the planning performance also considerably drops. This demonstrates that our proposed design choices together lead to achieving the best planning performance.

Sensitivity to Loss Weights. We report in Fig. 7 the planning success rate on OGBench-Scene across different combinations of inverse loss weights (λ_{inverse}) and persistence loss weights (λ_{persist}). The heatmap shows that both loss weights are not fully independent. However, we can identify a region $\lambda_{\text{inverse}} \in [0.1 - 0.3] \times \lambda_{\text{persist}} \in [0.1 - 0.5]$ that leads to reasonable planning success values.

Table 2: **Ablation on OGBench-Scene**. Dropping any component decreases planning success significantly with the inverse dynamics loss and the prefix sampling being the most impactful.

Training variant	Success %
DASWM	**45.3**
$-\mathcal{L}_{\text{inverse}}$	23.3
$-\mathcal{L}_{\text{persist}}$	40.0
− prefix sampling	32.7

Role of Persistence loss. We examine the effect of the persistence loss weight on the latent change of each token in DASWM's token set. As shown in Fig. 8, the strength of the persistence weight determines where the change energy is placed among the tokens. A $\lambda_{\text{persist}} = 0$ choice leads to all tokens changing at the same rate, as shown in the left-most column. Increasing λ_{persist} focuses the change heavily on the first token only. We notice in



(a) Push-T Background (b) Cube Background (c) TwoRoom Background (d) Scene

Figure 9: Planning success of one selected DASWM checkpoint per dataset when predicting the first $k \in \{1, 2, 4\}$ tokens. Dashed lines show the corresponding 192-dimensional LeWM $+\mathcal{L}_{\text{inverse}}$ baseline. Full-budget ($k = 4$) performance is the primary result; smaller prefixes are optional views of the same learned state. We see that in almost all datasets, planning with lower budget still achieves better or comparable success rate compared to the baselines, showing that DASWM successfully allocates dynamics relevant information at earlier tokens.

Fig. 7b), the first row corresponds to the success rates achieved by the different λ_{persist} values. The best planning is achieved when a moderate distribution of energy is reached.

Planning with flexible budget. A further advantage of the ordered token representation is that DASWM can plan with any prefix subset of the tokens at inference time, without retraining. Fig. 9 shows planning success for budgets $k \in \{1, 2, 4\}$ against the LeWM+$\mathcal{L}_{\text{inverse}}$ baseline (dotted). In almost every dataset, a single token already matches or exceeds the full 192-dimensional baseline: on Cube Background, $k = 1$ reaches 91.3% Moderate success versus the baseline's 80.7%, and TwoRoom Background and Scene show the same pattern. This confirms that dynamics-relevant information is concentrated in the earliest tokens, and that the remaining tokens are largely redundant for control. Full-budget ($k = 4$) results remain the primary comparison in Tab. 1; complete numbers for all budgets appear in Appendix A.5.

Number of tokens. Tab. 3 varies the number of tokens for OGBench-Scene while holding the total latent dimension fixed at 192 and keeping $(\lambda_{\text{persist}}, \lambda_{\text{inverse}}) = (6, 0.3)$ fixed on Scene. Four tokens yield the best full-budget success (37.3%), and splitting further into eight tokens degrades performance across every budget. This suggests that for the specific given task, a small number of tokens may be sufficient.

Table 3: **Number of tokens on OGBench-Scene.** Success (%) when planning with the first k tokens, at a fixed total latent dimension of 192 and $(\lambda_{\text{persist}}, \lambda_{\text{inverse}}) = (6, 0.3)$.

Tokens	$k = 1$	$k = 2$	$k = 4$	$k = 8$
2×96	31.3	34.7	–	–
4×48	31.3	34.7	**37.3**	–
8×24	22.7	29.3	34.7	33.3

6 Conclusion and Future Work

We demonstrate in this work that LeWM latent world model formulation can drastically fail when a static variation, such as different background color, is introduced into its training environments. We identify the reason of this failure as static-dynamic entanglement, where the representation encodes dynamics differently for each background. To address this failure we propose DASWM, a latent world model paradigm that learns a set of tokens ordered such that prefix tokens encode dynamics and suffix tokens encode static information, thereby achieving soft dynamic-static disentanglement. This is achieved by employing a set of objectives and training pipeline that allow for disentanglement and improving the planning performance. We show through experiments and ablations the efficacy of DASWM and the important role of each of our components.

Limitations and Future Work. However, there is potential for future improvements. We can see in Fig. 2 that there is still some entanglement remaining that our method does not fully resolve. Moreover, we evaluate DASWM mainly on simulation data, which does not yet establish the robustness of the method in real-life data with uncertainty. Therefore, we plan to investigate other sources of static variations, scaling up to real-world problems, and generative solutions for dealing with uncertainty.

REPRODUCIBILITY STATEMENT

Appendix A.1 specifies the architecture, objectives, optimization schedule, training seed, and checkpoint used for evaluation. Appendix A.3 documents the construction of the background variants, and Appendix A.2 defines the planner, filtering rules, success criteria, and evaluation seeds. Complete results for the models reported in the paper, including standard deviations and smaller update budgets, appear in Appendix A.5. We explicitly identify which datasets use episode-level or transition-level splits and disclose that loss coefficients were selected using the reported evaluation protocol. Anonymous code, configurations, dataset-generation scripts, and result manifests will be released with the submission.

AI USE STATEMENT

We used generative AI tools to assist with research ideation, experimental design, implementation, diagnostic analysis, figure preparation, and drafting and editing this manuscript. The authors reviewed AI-assisted code, analyses, claims, citations, and text, and take responsibility for the final content.

REFERENCES

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. *arXiv preprint arXiv:2506.09985*, 2025.

Roman Bachmann, Jesse Allardice, David Mizrahi, Enrico Fini, Oğuzhan Fatih Kar, Elmira Amirloo, Alaaeldin El-Nouby, Amir Zamir, and Afshin Dehghan. FlexTok: Resampling images into 1D token sequences of flexible length. In *International Conference on Machine Learning*, 2025.

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

Mu Cai, Jianwei Yang, Jianfeng Gao, and Yong Jae Lee. Matryoshka multimodal models. In *International Conference on Learning Representations*, 2025.

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. In *Robotics: Science and Systems*, 2023.

Junha Chun, Youngjoon Jeong, and Taesup Kim. Sparse imagination for efficient visual world model planning. In *International Conference on Learning Representations*, 2026.

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

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

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, 2024.

Wenbo Hu, Zi-Yi Dou, Liunian Harold Li, Amita Kamath, Nanyun Peng, and Kai-Wei Chang. Matryoshka Query Transformer for large vision-language models. In *Advances in Neural Information Processing Systems*, 2024.

Dongwon Kim, Gawon Seo, Jinsung Lee, Minsu Cho, and Suha Kwak. Planning in 8 tokens: A compact discrete tokenizer for latent world model. *arXiv preprint arXiv:2603.05438*, 2026.

Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. In *International Conference on Learning Representations*, 2020.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems*, 2020.

Lucas Maes, Quentin Le Lidec, Luiz Facury, Nassim Massaudi, Ayush Chaurasia, Francesco Capuano, Richard Gao, Taj Gillin, Dan Haramati, Damien Scieur, Yann LeCun, and Randall Balestriero. stable-worldmodel: A platform for reproducible world modeling research and evaluation. *arXiv preprint arXiv:2605.21800*, 2026a.

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

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 Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

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

Oren Rippel, Michael A. Gelbart, and Ryan P. Adams. Learning ordered representations with nested dropout. In *International Conference on Machine Learning*, 2014.

Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999.

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

Junhan Sun, Hao Zhao, and Guofeng Zhang. INTACT: Isomorphic intent-to-action learning for search-free world models. *arXiv preprint arXiv:2607.26056*, 2026.

Basile Terver, Tsung-Yen Yang, Jean Ponce, Adrien Bardes, and Yann LeCun. What drives success in physical planning with joint-embedding predictive world models? *arXiv preprint arXiv:2512.24497*, 2025.

Leonardo F. Toso, Davit Shadunts, Yunyang Lu, Nihal Sharma, Donglin Zhan, Nam H. Nguyen, and James Anderson. Learning invariant visual representations for planning with joint-embedding predictive world models. *arXiv preprint arXiv:2602.18639*, 2026.

Rishi Veerapaneni, John D. Co-Reyes, Michael Chang, Michael Janner, Chelsea Finn, Jiajun Wu, Joshua Tenenbaum, and Sergey Levine. Entity abstraction in visual model-based reinforcement learning. In *Conference on Robot Learning*, 2020.

Angel Villar-Corrales and Sven Behnke. PlaySlot: Learning inverse latent dynamics for controllable object-centric video prediction and planning. In *International Conference on Machine Learning*, 2025.

Zizhao Wang, Kaixin Wang, Li Zhao, Peter Stone, and Jiang Bian. Dyn-O: Building structured world models with object-centric representations. In *Advances in Neural Information Processing Systems*, 2025.

Xin Wen, Bingchen Zhao, Ismail Elezi, Jiankang Deng, and Xiaojuan Qi. "Principal Components" enable a new language of images. In *International Conference on Computer Vision*, 2025.

Tianqiu Zhang, Muyang Lyu, Yufan Zhang, Fang Fang, and Si Wu. DiLA: Disentangled latent action world models. *arXiv preprint arXiv:2605.15725*, 2026a.

Zhenghao Zhang, Yuanxiang Wang, Zhenyu Guan, Yujia Yang, Bingkang Shi, Tianyu Zong, Hongzhu Yi, Guoqing Chao, Xingchen Chen, Tiankun Yang, Chenxi Bao, Tao Yu, Jingjing Zhou, and Jungang Xu. Delta-JEPA: Learning action-sensitive world models via latent difference decoding. *arXiv preprint arXiv:2606.31232*, 2026b.

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

A SUPPLEMENTARY MATERIAL

CONTENTS

A.1 Implementation Details 13
A.2 Evaluation Details 13
A.3 Background dataset construction and validation 14
A.4 Analysis Experiment Details 15
A.5 Additional Results 16
A.6 Additional Qualitative Results 17

A.1 IMPLEMENTATION DETAILS

Encoder and predictor. The encoder is a Vision Transformer with 14×14 patches on 224×224 images, width 192, depth 12, and three heads, trained from random initialization. $K = 4$ learned register tokens attend to image patches and causally to earlier registers; patch tokens do not attend to registers. A linear layer maps each register to $d = 48$ dimensions. The dynamics model is a six-layer transformer with hidden size 192, 16 heads of dimension 64, MLP dimension 2048, and dropout 0.1. It receives three context frames (except in TwoRoom it's a single frame) and attends causally across register positions. As in LeWM, encoder targets and predictions pass through BatchNorm MLP projectors with hidden dimension 2048.

Inverse head. For a sampled budget k, the inverse-dynamics head jointly processes the encoded target difference $Z_{t+1}^{1:k} - Z_t^{1:k}$, padded to the maximum prefix width, and regresses the complete action block. The head is a two-layer MLP with hidden dimension 384 and is discarded after training.

Budget sampling and normalization. For the first 10,000 steps, $k = 4$ is always used as warm-up. Afterwards, each minibatch draws k uniformly from $\{1, 2, 4\}$.

Temporal context handling. In the main text, for clarity, we assume that a transition is from a single frame to another. In practice, however, extra previous frames are also encoded and used to predict the next state. The predictor receives N consecutive encoded frames $Z_{t-N+1:t}$ and their action blocks, attends causally over time, and predicts the next prefix at each temporal position. We average the future-prediction loss over these N transitions; the inverse loss is evaluated on the final transition. Context and target frames share the same encoder and are optimized jointly. We use $N = 3$ following LeWM.

Optimization. We use AdamW with $\beta = (0.9, 0.95)$, learning rate $5 \cdot 10^{-5}$, weight decay 10^{-3}, linear warm-up over the first 5% of training, cosine decay to 10^{-6}, gradient-norm clipping at 1, batch size 128, and bfloat16 mixed precision. Models train for 5 epochs on Scene, TwoRoom Background, and for 5 epochs on Push-T Background. The SIGReg weight is 0.09. The $(\lambda_{\text{persist}}, \lambda_{\text{inverse}})$ pairs are $(0.1, 0.1)$ on Scene, $(6, 0.3)$ on Cube and TwoRoom Background, and $(3, 8)$ on Push-T Background.

Baselines. LeWM uses our implementation of the official model with one 192-dimensional state vector, the same ViT and predictor depth, three context frames, SIGReg weight 0.09 with 1024 directions, and the same optimizer and training length as the corresponding DASWM model. When inverse supervision is enabled, its two-layer head has hidden dimension 512. Its selected inverse weights are 3 on Push-T Background, 1 on Cube and TwoRoom Background, and 0.3 on Scene.

A.2 EVALUATION DETAILS

Planner. All evaluations in the paper use the stable-worldmodel CEM planner (Rubinstein, 1999; Maes et al., 2026a) with 300 candidates, 30 elites, and 30 iterations over a horizon of five action blocks. Each planned action is repeated for five environment steps. The planned sequence is executed

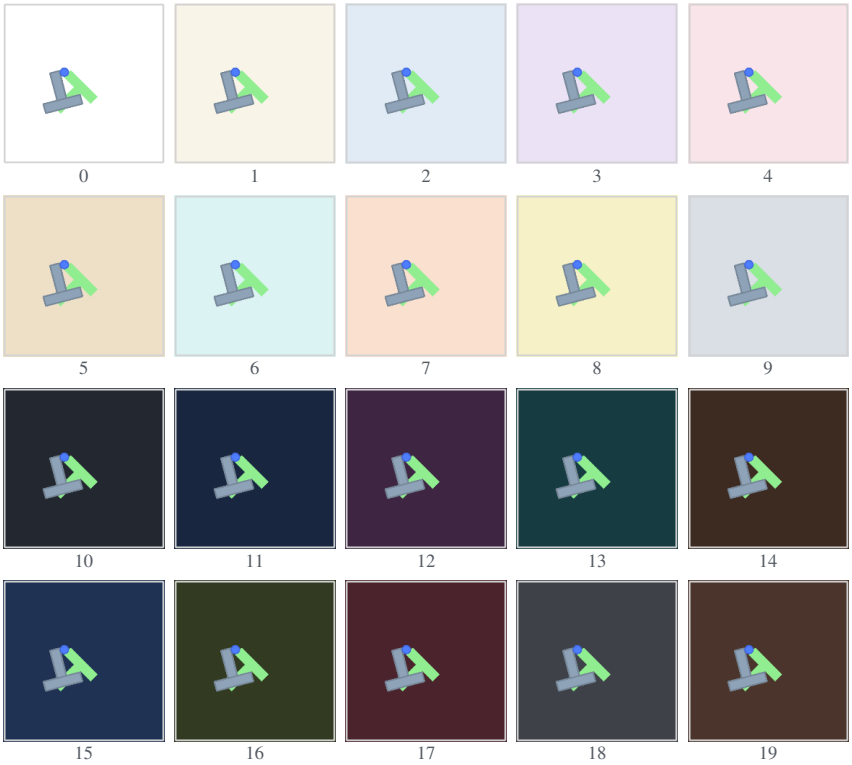


Figure A1: The 20 Push-T Background colors. One stored frame is re-rendered under every color. Rendering it with its assigned color reproduces the stored dataset frame exactly.

before replanning, and each episode has a budget of 50 environment steps. CEM minimizes the terminal latent cost

$$J(a_{t:t+H-1}; k) = \left\| \widehat{Z}^{(k)}_{t+H} - E_\theta(o_{\text{goal}}) \right\|_2^2 . \tag{10}$$

Filtered Moderate and Strict protocols. For Push-T, Cube, and TwoRoom, each evaluation seed samples 50 start–goal pairs, one per episode, with the goal 25 steps ahead, and excludes pairs satisfying the success criteria at the first state, following the principle used by CLEAR-LeWM (Sun et al., 2026). Push-T pairs come from validation episodes. Cube and TwoRoom pairs come from all episodes because their inherited LeWM data split is transition-based; TwoRoom pairs additionally start and end in different rooms. Moderate uses each task's original predicate: pusher and block within 20 position units and 20° on Push-T, cube position within 4 cm on Cube, and agent distance below 16 on TwoRoom. Strict requires the Push-T block within 10 units and 10° for three consecutive steps; the Cube within 3 cm and 15° using symmetry-aware orientation for three consecutive steps; and TwoRoom distance below 8 after routing through the door and reaching the goal side.

Scene protocol. Each seed samples 50 starts from held-out validation transitions, uses the frame 50 steps ahead as the goal, and removes initially successful starts.

Seeds and model selection. All models use training seed 3072. Evaluation seeds 42, 43, and 44 determine the sampled start-goalpairs. We report their mean and sample standard deviation in all experiments.

A.3 Background Dataset Construction and Validation

Each background dataset are generated from their original counterparts by preserving the Episodes, actions, simulator states, and all other non-pixel metata are copied unchanged. Each episode receives

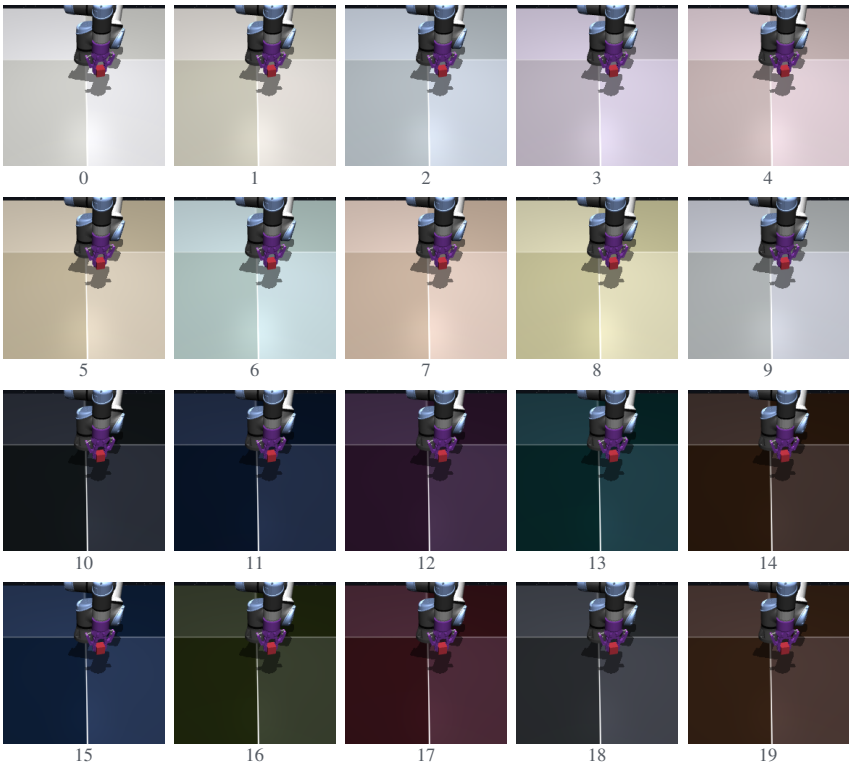


Figure A2: The 20 Cube Background colors applied to the two shades of the checkered floor.

one of 20 background colors, fixed for the entire episode. Frames are rendered using the stable-worldmodel environment(Maes et al., 2026a).

Push-T Background. The colors replace the canvas; color 0 is the original background. Push-T uses an episode-level 90/10 split with seed 3072 (16,816 training and 1,869 validation episodes), and colors are balanced independently within each split. Frames are rendered at 512×512 and resized to 224×224, as in the original dataset.

OGBench-Cube Background. In the case of this dataset, each color modifies the two shades of the checkered floor in the environment; the original floor is not part of the palette. Colors are assigned by shuffling $(e \bmod 20)_{e=1}^{N}$ over episodes with seed 3072, which balances the palette. The same transition-level 90/10 split used by LeWM is adapted here as well.

TwoRoom Background. Each color replaces the canvas and door region; color 0 is the original white background. Assignment and the transition-level split follow Cube Background.

A.4 Analysis Experiment Details

We analyze whether the same action-induced change is represented consistently under different backgrounds. We sample $N = 500$ validation episodes of Push-T Background and select one transition from each. Both states are rendered under each of the $B = 20$ backgrounds, so every transition–background pair is observed and the design is balanced.

For transition i under background b, we compute the latent change:

$$\Delta z_{i,b} = E(o_{t+1}^{i,b}) - E(o_t^{i,b}) \in \mathbb{R}^D. \tag{11}$$

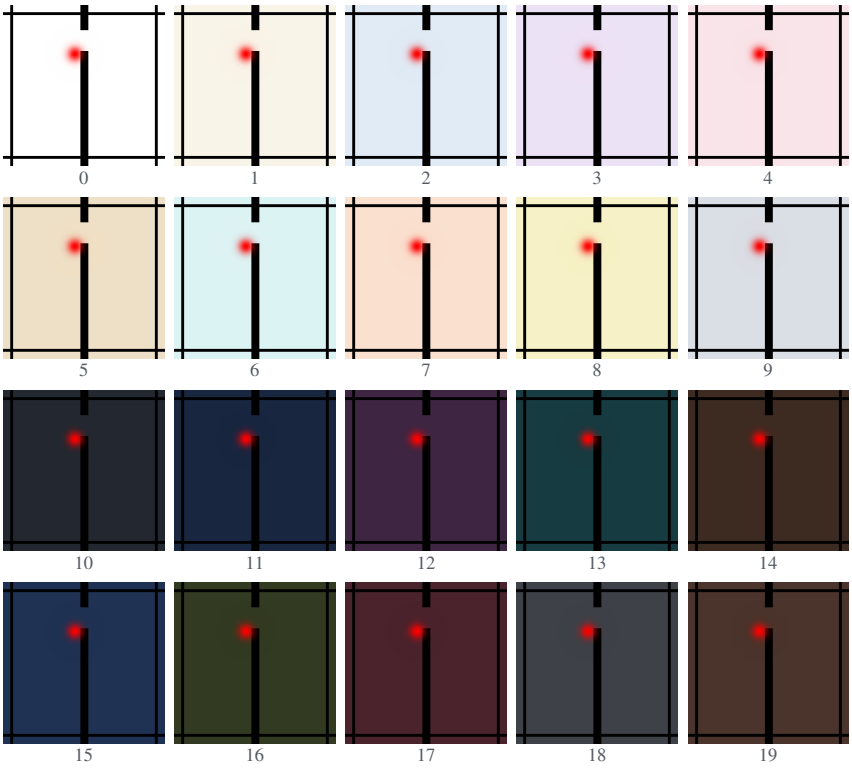


Figure A3: The 20 TwoRoom Background colors applied to the canvas and door region.

Let $\overline{\Delta z}$ be the grand mean and $\overline{\Delta z}_{i\cdot}$ and $\overline{\Delta z}_{\cdot b}$ the transition and background means. We define:

$$\begin{aligned} \mathrm{SS}_{\mathrm{total}} &= \sum_{i,b} \|\Delta z_{i,b} - \overline{\Delta z}\|_2^2, \\ \mathrm{SS}_{\mathrm{transition}} &= B \sum_{i} \|\overline{\Delta z}_{i\cdot} - \overline{\Delta z}\|_2^2, \\ \mathrm{SS}_{\mathrm{background}} &= N \sum_{b} \|\overline{\Delta z}_{\cdot b} - \overline{\Delta z}\|_2^2, \\ \mathrm{SS}_{\mathrm{interaction}} &= \sum_{i,b} \|\Delta z_{i,b} - \overline{\Delta z}_{i\cdot} - \overline{\Delta z}_{\cdot b} + \overline{\Delta z}\|_2^2. \end{aligned} \tag{12}$$

The three components sum exactly to $\mathrm{SS}_{\mathrm{total}}$. The transition component is the change across backgrounds, the background component is an additive offset shared across transitions, and the interaction is the background-dependent representation of a particular physical transition. We report each as a fraction of the total. Confidence intervals use 500 bootstrap resamples of complete transition blocks, retaining all 20 backgrounds.

A.5 Additional Results

We report in the main paper the mean success rate of 3 evaluation seeds. For completness we report here the full results with the standard deviations. Results are available in Tab. A1.

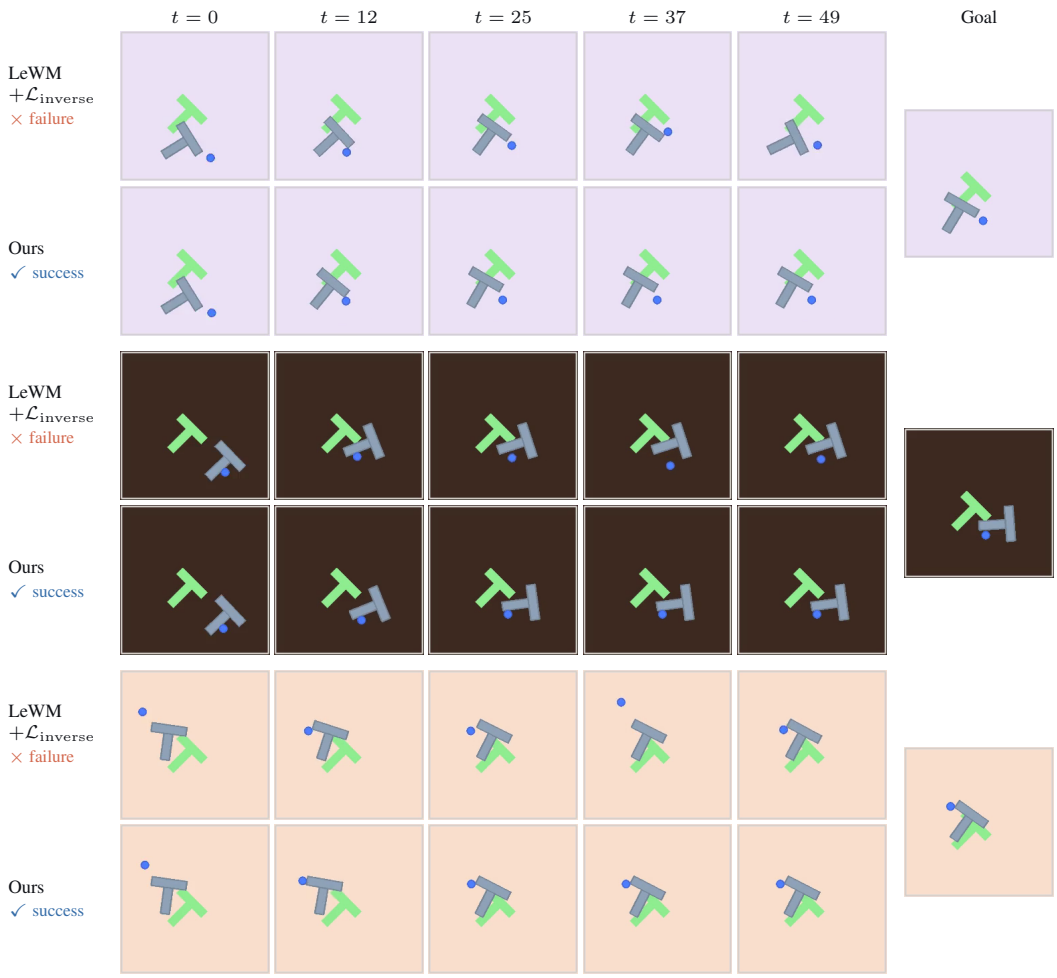


Figure A4: Further Push-T Background episodes (Moderate criterion), LeWM without the inverse loss (top) and DASWM (bottom) in each block. The green T is the environment's fixed target marker; the goal observation is on the right.

Table A1: Planning with fewer update tokens, reported as mean and sample standard deviation over evaluation seeds 42–44.

Dataset	Criterion	LeWM $+\mathcal{L}_{\text{inverse}}$ (192)	DASWM $k = 1$ (48)	DASWM $k = 2$ (96)	DASWM $k = 4$ (192)
Push-T Background	Moderate	72.7±5.0	70.7±9.0	73.3±8.3	74.7±9.5
	Strict	42.0±5.3	38.0±10.6	40.0±5.3	42.7±9.0
Cube Background	Moderate	80.7±9.9	91.3±1.2	92.7±6.4	95.3±5.0
	Strict	55.3±9.9	70.7±8.3	74.7±9.2	74.0±5.3
TwoRoom Background	Moderate	98.7±1.2	99.3±1.2	100.0±0.0	100.0±0.0
	Strict	87.3±2.3	87.3±3.1	92.0±4.0	93.3±3.1
Scene	Success	36.7±9.9	42.7±11.7	49.3±11.0	45.3±7.6

A.6 ADDITIONAL QUALITATIVE RESULTS

Figures A4, A5, A6, and A7 show further episodes in the format of Fig. 5.



Figure A5: Further Cube Background episodes (Strict criterion), LeWM $+\mathcal{L}_{\text{inverse}}$ (top) and DASWM (bottom) in each block. The goal observation is on the right.

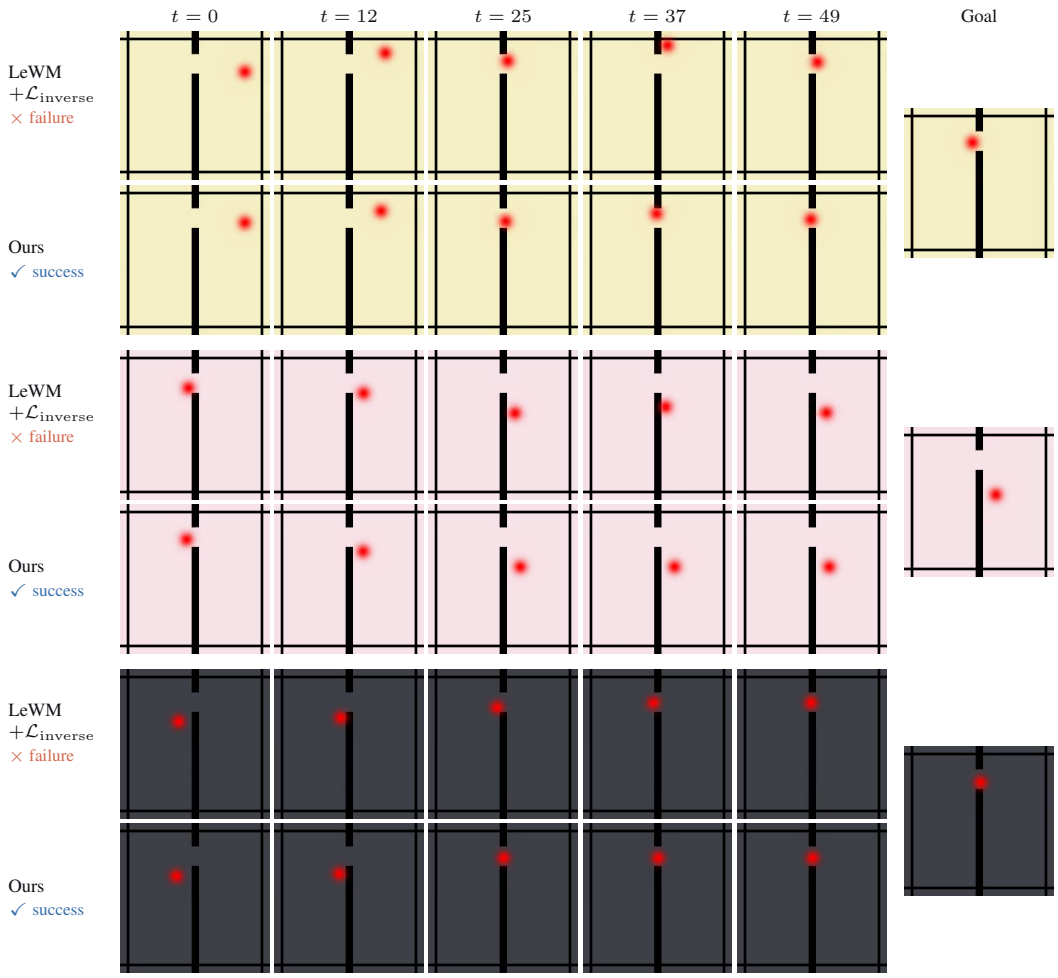


Figure A6: Evaluation Episodes on TwoRoom-Background Dataset.

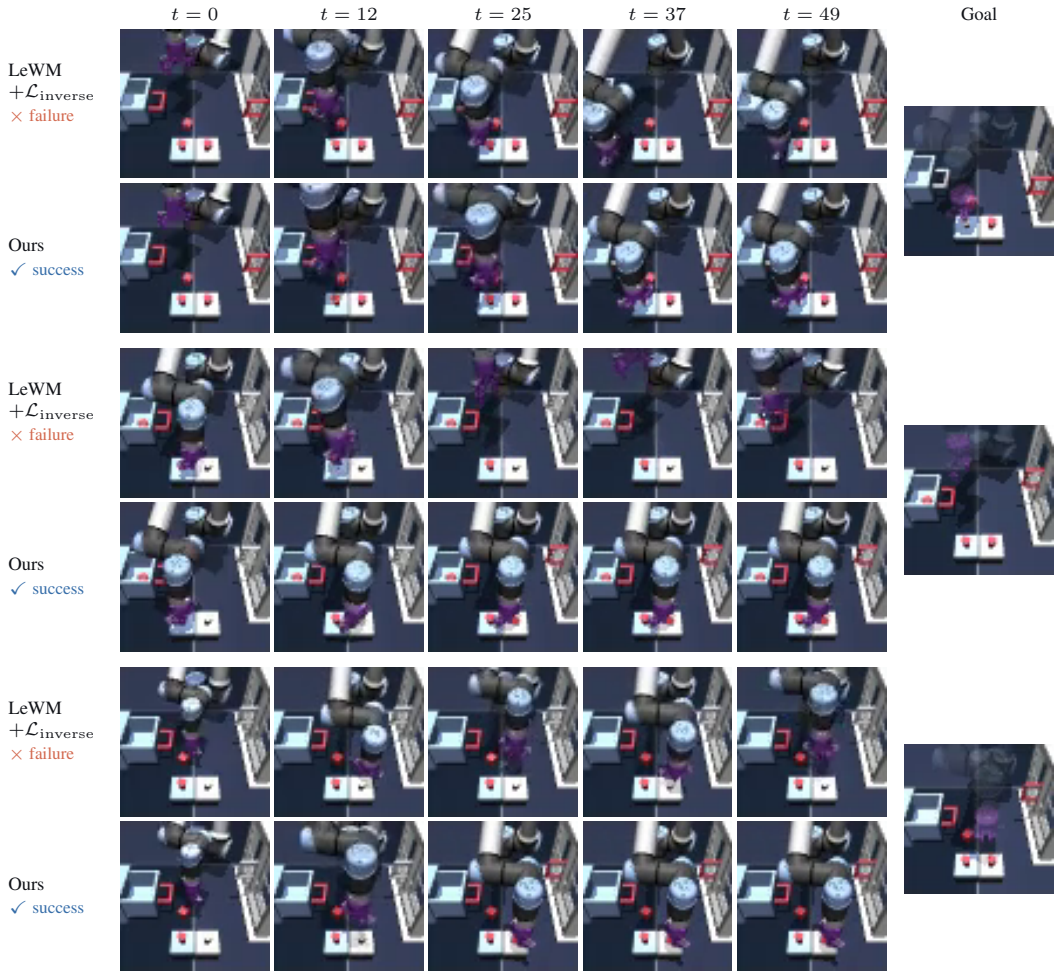


Figure A7: Evaluation Episodes on the OGBench-Scene Dataset.