Slot-JEPA: Object-Structured Prediction Interfaces for Latent World Models

Anonymous authors
Paper under double-blind review

Abstract

As a prototypical latent world model, Joint-Embedding Predictive Architectures (JEPAs) provide strong semantic representations. However, dense patch representations expose a redundant interface to downstream predictors, which must implicitly infer spatial organization and temporal correspondence, presenting three challenges. First, object information is distributed across patches, forcing each downstream predictor to redundantly recover object structure that could otherwise be reused across tasks before modeling interactions. Second, the lack of explicit entity correspondence across time makes historical transitions expensive, restricting the predictor's flexibility in utilizing context. Third, limited by the lack of explicit transport structure, existing JEPA models typically use pointwise ℓ_1/ℓ_2 distances, which are insensitive to spatial displacement when predicted and goal supports are largely disjoint. We propose the **Slot-based Joint-Embedding Predictive Architecture (Slot-JEPA)**, which decomposes dense representations into stable semantic slots and time-varying spatial masks. Pretrained once and frozen downstream, this decomposition exposes a reusable object-structured interface. Building on it, our **Transition Memory Predictor (TMP)** stores visual transitions as compact masks and retrieves useful context using statistics of corresponding slots. Its two branches separate slot-specific dynamics from shared cross-slot dynamics. We further use Unbalanced Optimal Transport (UOT) to supervise mask prediction and evaluate candidate plans by transporting mass between aligned predicted and goal masks, providing a signal better aligned with spatial progress. Across seven visual-understanding benchmarks, the slot-reconstructed features remain transferable. Slot-JEPA also improves goal-conditioned planning across six planar environments and four robotic arm planning tasks on the REALM-JEPA benchmark.

1 Introduction

Building agents that can anticipate the consequences of their actions is a long-standing goal of artificial intelligence. World models pursue this goal by predicting how an environment evolves, allowing agents to evaluate possible behaviors before acting (Ha & Schmidhuber, 2018). Early visual world models operate in pixel space (Finn et al., 2016; Finn & Levine, 2017), but reconstructing every pixel requires modeling details that may be unpredictable. Latent world models instead predict action-conditioned changes in learned representations (Hafner et al., 2019; 2020) without rendering pixels.

Joint-Embedding Predictive Architectures (JEPAs) provide a promising foundation for this approach. Typically, JEPAs learn to predict masked image/video representations from visible context (Assran et al., 2023; Bardes et al., 2024). DINO-WM predicts representations changing from offline interactions (Zhou et al., 2025), while V-JEPA 2-AC post-trains an action-conditioned predictor for planning (Assran et al., 2025). Recent work further improves representation density and temporal consistency (Mur-Labadia et al., 2026), or adapts the model online (Wang et al., 2026).

Despite this progress, we argue that dense patch representations might not be ideal for downstream dynamics learning and planning for three reasons: **1)** An object's information is distributed across patches, leaving its structure implicit. Each task-specific predictor must recover this structure, model interactions, and decode the result back into patches. The object organization encoded during pretraining is not exposed as a reusable interface. **2)** A predictor needs relevant and diverse context to balance local and global views of dynamics. Yet patch-based history is highly redundant, making

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.

transitions expensive to store and their relevance costly to compute. **3)** Patch representations form dense spatial grids without explicit entity correspondence or transport structure. Existing methods therefore compare predicted and goal grids with pointwise ℓ_1 or ℓ_2 distance (Zhou et al., 2025; Assran et al., 2025). When their supports are largely disjoint, these losses can become insensitive to spatial displacement and rank progress poorly.

We therefore propose ***Slot**-based **J**oint-**E**mbedding **P**redictive **A**rchitecture (**Slot-JEPA**)*, which turns dense patches into a reusable object-structured interface. A task-agnostic decomposition module addresses problem **1)** by factorizing the dense feature field into stable semantic slots (Locatello et al., 2020; Seitzer et al., 2023; Nam et al., 2026) and time-varying spatial masks. Pretrained once and frozen for downstream tasks, it preserves the original feature field through reconstruction while exposing its object organization.

For problem **2)**, the temporal stability of slots allows visual history to be stored and reused as compact masks rather than high-dimensional patch representations. Inspired by memory-based Video Object Segmentation (Oh et al., 2019; Cheng et al., 2024), our proposed ***T**ransition **M**emory **P**redictor (**TMP**)* uses statistics of masks associated with the same slots to assess relevance to the current query and retrieve relevant and diverse, potentially non-contiguous transitions. We further design two branches that separate slot-specific dynamics from inter-slot interaction dynamics, reducing the burden on dynamics learning.

Finally, problem **3)** motivates our Unbalanced Optimal Transport (UOT) loss for planning. Using the temporal correspondence of slots, it replaces pointwise comparison between predicted and goal patch representations with transport between their aligned masks. Moving mask mass across space provides a smooth planning signal that remains informative over long displacements.

Slot decomposition turns a perception-oriented patch field into a reusable interface for prediction, memory maintenance, and goal evaluation. Across seven visual-understanding benchmarks, slot-reconstructed features retain most of the semantic information. Across six planar tasks and four Robotic Arm Planning tasks from the REALM-JEPA dataset, Slot-JEPA improves goal-conditioned planning. The contributions are:

- We introduce Slot-JEPA, which compresses patch representations into shared semantic slots and time-varying masks, separating reusable visual organization from task-specific dynamics and easing predictor learning.
- We develop TMP to retrieve relevant, non-contiguous transitions from memory instead of relying only on a continuous recent-history window.
- We predict action effects in mask space and use UOT for goal evaluation, providing a signal better aligned with spatial progress when ℓ_1/ℓ_2 losses become insensitive to displacement.

2 Related Work

Due to space constraints, we briefly review the most relevant work here and point each topic to its corresponding extended discussion in Appendix A.

Joint-Embedding Predictive World Models. Slot-JEPA builds on JEPA representation learning (Assran et al., 2023; Bardes et al., 2024) and action-conditioned latent planning in DINO-WM and V-JEPA 2 (Zhou et al., 2025; Assran et al., 2025). It retains a pretrained visual encoder while replacing dense patch prediction with a slot-and-mask interface for dynamics and goal evaluation.

Object-Centric Representation and Dynamics. SlotFormer and Causal-JEPA improve the object awareness of learned representations: SlotFormer models dynamics over object-slot sequences, while Causal-JEPA applies masking based on slots (Wu et al., 2023; Nam et al., 2026). Slot-JEPA instead inserts a task-agnostic slot decomposition layer between the patch representation and downstream planning, which exposes an object-centric interface for prediction, memory, and goal evaluation.

Memory-Based VOS and Transition Memory. Memory-based video object segmentation retrieves earlier observations to guide mask propagation (Oh et al., 2019; Cheng & Schwing, 2022; Cheng et al., 2024). TMP adapts this idea to action-conditioned dynamics by retrieving observed state–action–next-state transitions as context for a new prediction.

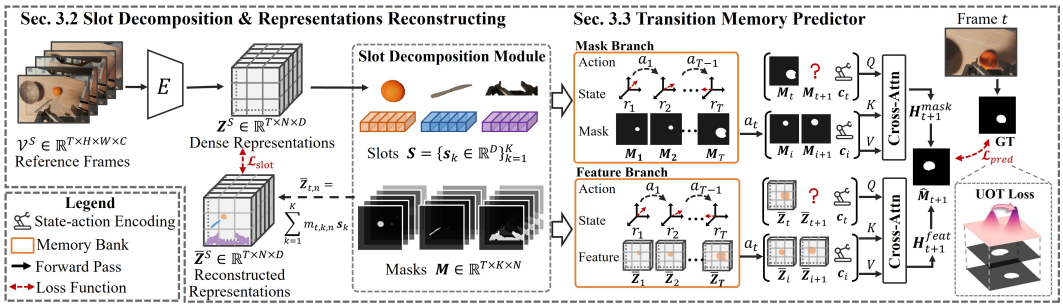

Figure 1: Overview of Slot-JEPA. The slot decomposition module factorizes patch representations into stable slots $\boldsymbol{S}$ and time-varying masks $\boldsymbol{M}$. TMP predicts $\hat{\boldsymbol{M}}_{t+1}$ with sampled transition memory. The prediction is supervised against the next-frame mask by $\mathcal{L}_{\text{pred}}$, including UOT loss.

3 Methodology

3.1 Overview

At each time step, the multimodal observation $o_t = (v_t, r_t)$ contains a visual observation v_t and, when available, a low-dimensional state observation r_t, such as the robot end-effector pose and gripper state; a_t denotes the action. Given previously observed sequences, a world model predicts how the world will evolve under candidate future action sequences. At test time, given a current observation o_0 and a visual goal v_g, model-predictive control (MPC) (Chua et al., 2018) selects the action sequence whose predicted visual observations are closest to v_g, as measured by a goal distance $d(\hat{v}, v_g)$.

An overview of Slot-JEPA is shown in Fig. 1. Slot-JEPA inserts a reusable object-structured interface between a general visual encoder and task-specific predictors. We therefore initialize the visual encoder with V-JEPA 2.1 (Mur-Labadia et al., 2026), and jointly pretrain the slot decomposition module and Transition Memory Predictor (TMP). The module converts patch representations into stable slots and spatial masks, which serve as the basic TMP inputs and outputs. After pretraining, we freeze the visual encoder and slot decomposition module, and fine-tune the transferred TMP on each downstream control task. A UOT-based loss is applied during training and planning.

3.2 Pretrained Slot Decomposition Module

Let $\mathcal{V}^S$ denote the reference frames for slot extraction, and let $\boldsymbol{Z}^S = E(\mathcal{V}^S)$ be their dense patch representations. The slot decomposition module extracts a shared set of K slots $\boldsymbol{S} = \{\boldsymbol{s}_k \in \mathbb{R}^D\}_{k=1}^K$ from $\boldsymbol{Z}^S$. Holding $\boldsymbol{S}$ fixed, it maps any encoded sequence $\boldsymbol{Z} = E(v_{1:T}) \in \mathbb{R}^{T\times N\times D}$ from the same environment to a time-varying spatial assignment $\boldsymbol{M} = [m_{t,k,n}] \in [0,1]^{T\times K\times N}$. The slots summarize visual content shared by the reference frames, while $m_{t,k,:} \in [0,1]^N$ indicates where the content of slot k appears. Ideally, they reconstruct each patch representation through

$$\boldsymbol{Z}_{t,n} \approx \bar{\boldsymbol{Z}}_{t,n} = \sum_{k=1}^{K} m_{t,k,n}\boldsymbol{s}_k. \tag{1}$$

Whereas the dense representation $\boldsymbol{Z}$ contains TND scalars, the factorized representation $(\boldsymbol{S}, \boldsymbol{M})$ contains $KD + TKN$ scalars. The latter is therefore more compact since $K \ll D$ in practice. To obtain $\boldsymbol{S}$, we use Slot Attention (Locatello et al., 2020), which iteratively aggregates the reference patches into competing slots. Let $\boldsymbol{s}_{j,r}^{\text{loc}}$ denote local slot j at iteration r, the local update is

$$\begin{aligned} \alpha_{n,j,r} &= \text{softmax}_j\Big(k(\boldsymbol{z}_n)^\top q(\boldsymbol{s}_{j,r}^{\text{loc}})/(\tau\sqrt{D})\Big), \quad \boldsymbol{u}_{j,r} = \sum_n \frac{\alpha_{n,j,r}}{\sum_{n'}\alpha_{n',j,r}} v(\boldsymbol{z}_n), \\ \tilde{\boldsymbol{s}}_{j,r+1}^{\text{loc}} &= \text{GRU}(\boldsymbol{u}_{j,r}, \boldsymbol{s}_{j,r}^{\text{loc}}), \quad \boldsymbol{s}_{j,r+1}^{\text{loc}} = \tilde{\boldsymbol{s}}_{j,r+1}^{\text{loc}} + \text{MLP}\big(\text{LN}(\tilde{\boldsymbol{s}}_{j,r+1}^{\text{loc}})\big). \end{aligned} \tag{2}$$

Here, $\tau > 0$ controls assignment softness, and GRU, LN, and MLP denote the gated recurrent unit, layer normalization, and multilayer perceptron, respectively. For initialization, we partition the reference frames into non-overlapping windows, extract local slots from each window, and use farthest-point sampling to select diverse initial slots. Importantly, Slot-JEPA does not assume that slots corre-

spond to predefined semantic categories. Instead, the shared reference-derived slots provide a fixed latent basis within each planning context, and masks encode temporal variation relative to this basis.

On top of the extracted slots $\boldsymbol{S}$, we compute $\boldsymbol{M}$ in the pretrained latent space (Kabra et al., 2021). Patch features cross-attend to the slots, and an MLP g produces slot-assignment logits:

$$\boldsymbol{H} = \mathrm{CA}(\boldsymbol{Z}; \boldsymbol{S}), \qquad A_{t,k,n} = g([\boldsymbol{H}_{t,n} \| \boldsymbol{s}_k]), \qquad m_{t,k,n} = \mathrm{softmax}_k(A_{t,k,n}). \tag{3}$$

Here, CA denotes the cross-attention stack. Competition across slots encourages object-aligned grouping but does not require one slot per physical instance: a slot may cover multiple entities, a part, or background. Increasing K permits a finer decomposition but increases the prediction cost.

Having obtained $\boldsymbol{S}$ and $\boldsymbol{M}$ for frame t, we reconstruct the patch representation $\bar{\boldsymbol{Z}}_t$ with Eq. (1). Then we train the decomposition module by reconstructing the original dense representation $\boldsymbol{Z}_t$, while encouraging different slots to encode distinct content:

$$\mathcal{L}_{\text{slot}} = \frac{1}{T}\sum_{t=1}^{T} \|\bar{\boldsymbol{Z}}_t - \boldsymbol{Z}_t\|_1 + \frac{\lambda_{\text{orth}}}{K(K-1)} \sum_{k \neq j} \left(\frac{\boldsymbol{s}_k^\top \boldsymbol{s}_j}{\|\boldsymbol{s}_k\|_2 \|\boldsymbol{s}_j\|_2} \right)^2. \tag{4}$$

3.3 Transition Memory Predictor

The temporally stable slots make compact, query-dependent reuse of the observed trajectory possible: visual states can be stored as masks and compared through statistics of corresponding slots. During training, the observed transitions in each sampled video clip provide the context directly. At inference, TMP samples relevant and diverse context transitions $\mathcal{C}_t$ based on the query state and mask geometry. It then uses the same symmetric context–query structure to predict the next mask in both settings.

Transition memory bank. Each memory bank contains transitions from one environment that are decoded against the same slots $\boldsymbol{S}$. Visual states can therefore be stored solely as masks $\boldsymbol{M}_i$:

$$\mathcal{B}_t = \{(\boldsymbol{M}_i, r_i, a_i, \boldsymbol{M}_{i+1}, r_{i+1})\}_{i \in \mathcal{I}_t^{\text{mem}}}, \tag{5}$$

where r_i is the optional low-dimensional state, a_i is the action, and $\mathcal{I}_t^{\text{mem}}$ indexes the transitions available before the query. Let $\boldsymbol{\rho}_i \in \mathbb{R}^K$, $\boldsymbol{\mu}_i \in \mathbb{R}^{2K}$ collect the areas of the K masks and 2-D centroids in $\boldsymbol{M}_i$ respectively, and $\boldsymbol{u}_i = [r_i \| \sqrt{\lambda_v}\boldsymbol{\rho}_i \| \sqrt{\lambda_v}\boldsymbol{\mu}_i]$ summarizes each state. At inference, we sample transitions $\mathcal{C}_t \subset \mathcal{B}_t$ without replacement according to

$$d_{\text{mem}}(t,i) = \|\boldsymbol{u}_t - \boldsymbol{u}_i\|_2^2, \qquad P_t(i) = (1-\gamma)\,\mathrm{softmax}_i(-d_{\text{mem}}(t,i)) + \gamma/|\mathcal{B}_t|. \tag{6}$$

The mixing weight $\gamma \in [0,1]$ balances distance-weighted sampling for relevance with uniform sampling for diversity, which is essential for in-context examples (Huang et al., 2026).

Predictor with symmetric structure. TMP contains a slot-wise mask branch for spatial changes and a shared feature branch for interactions among slots. Both process observed transitions as context blocks and the transition to be predicted as a query block. For each context transition i, we encode its state and action as $\boldsymbol{c}_i = E_c([r_i \| a_i \| r_{i+1}])$. In the mask branch, slot k forms a spatial field $\boldsymbol{X}_{i,k,n} = m_{i,k,n}\boldsymbol{s}_k$. The current and next fields are paired to form a context block

$$\boldsymbol{B}_{i,k} = \left[P_c(\boldsymbol{c}_i), P_s(\boldsymbol{s}_k), P_m(\boldsymbol{X}_{i,k} \| \boldsymbol{X}_{i+1,k})\right] \in \mathbb{R}^{(N'+2)\times d'}. \tag{7}$$

Here P_c, P_s, and P_m project inputs to width d'. For the query, unknown future inputs are replaced by learnable tokens. Specifically, $\boldsymbol{q}^r$ replaces r_{t+1} in the state–action encoding: $\boldsymbol{c}_t = E_c([r_t \| a_t \| \boldsymbol{q}^r])$, and $\boldsymbol{q}_k$ replaces the future spatial field. Then we have

$$\boldsymbol{B}_{t,k} = \left[P_c(\boldsymbol{c}_t), P_s(\boldsymbol{s}_k), P_m(\boldsymbol{X}_{t,k} \| \boldsymbol{q}_k)\right], \quad \boldsymbol{B}_k = \left[\boldsymbol{B}_{i,k}\right]_{i \in \mathcal{I}_t}, \quad \boldsymbol{B}_k^+ = [\boldsymbol{B}_k; \boldsymbol{B}_{t,k}]. \tag{8}$$

where $\mathcal{I}_t$ indexes the selected transitions. Then we apply multiple attention blocks:

$$\boldsymbol{H}_{t,k}^{\text{mask}} = \mathrm{Attn}\left(\boldsymbol{B}_{t,k}W_Q, \boldsymbol{B}_k^+ W_K, \boldsymbol{B}_k^+ W_V\right), \quad \mathrm{Attn}(Q,K,V) = \mathrm{softmax}(QK^\top/\sqrt{d})V. \tag{9}$$

We select only the query positions $\mathcal{J}$ occupied by learnable tokens and concatenate $(\boldsymbol{H}_{t,k}^{\text{mask}})_{\mathcal{J}}$ into $\boldsymbol{H}_{t+1}^{\text{mask}}$. The feature branch follows the same process, but takes $[\bar{\boldsymbol{Z}}_i \| \boldsymbol{M}_i^\top]$ as its spatial input, yielding interaction features $\boldsymbol{H}_{t+1}^{\text{feat}}$. Finally, we broadcast $\boldsymbol{H}_{t+1}^{\text{feat}}$ across slots and combine both branches with an MLP f_{out} and slot-wise softmax to predict the future assignment map:

$$\hat{\boldsymbol{M}}_{t+1} = \mathrm{softmax}_{\text{slot}}\left(f_{\text{out}}\left([\boldsymbol{H}_{t+1}^{\text{mask}} \| \boldsymbol{H}_{t+1}^{\text{feat}}]\right)\right). \tag{10}$$

Action-free pretraining. To pretrain TMP on natural videos without action annotations, an inverse transition encoder B_ψ compresses each observed change into a low-dimensional pseudo-action:

$$\tilde{a}_t = B_\psi(\boldsymbol{S}, \boldsymbol{M}_t, \boldsymbol{M}_{t+1}) \in \mathbb{R}^{d_b}, \qquad d_b \ll KN. \tag{11}$$

For the query transition, $\boldsymbol{M}_{t+1}$ reaches TMP only through the bottleneck $\tilde{a}_t$, limiting direct target leakage while preserving the transition information needed for prediction. After representation pretraining, we discard the inverse transition and state encoders, and replace them with task-specific encoders for r_t and a_t. The remaining TMP weights are fine-tuned on downstream interaction data.

3.4 Transport-Based Objective for Mask Prediction

We next specify how the mask predictions are trained and used for goal evaluation. During training, the fixed $\boldsymbol{S}$ decodes the held-out future representation $\boldsymbol{Z}_{t+1}$ into the target mask $\boldsymbol{M}_{t+1}$. Given action a_t and transition context $\mathcal{C}_t$, TMP produces $\hat{\boldsymbol{M}}_{t+1}$. The target and predicted masks reconstruct $\bar{\boldsymbol{Z}}_{t+1,n} = \sum_k m_{t+1,k,n} \boldsymbol{s}_k$ and $\hat{\boldsymbol{Z}}_{t+1,n} = \sum_k \hat{m}_{t+1,k,n} \boldsymbol{s}_k$, respectively.

Conventional latent predictors directly predict $\hat{\boldsymbol{Z}}_{t+1}$ and minimize its ℓ_1 distance to $\boldsymbol{Z}_{t+1}$ (Bardes et al., 2024; Assran et al., 2025). With our factorization, the same objective naturally transfers to $\|\hat{\boldsymbol{M}}_{t+1} - \boldsymbol{M}_{t+1}\|_1$. However, in either representation or mask space, pointwise ℓ_1 can become insensitive to further spatial displacement when predicted and target supports are largely disjoint. We therefore use ***Unbalanced Optimal Transport (UOT)*** to measure the cost of moving mask mass from a prediction to its target.

Consider masks $p, q \in \mathbb{R}_+^N$, let $\boldsymbol{\xi}_i \in [0,1]^2$ be the normalized coordinate of patch i, and set $c_{ij} = \|\boldsymbol{\xi}_i - \boldsymbol{\xi}_j\|_2^2$ in the ground-cost matrix $\boldsymbol{C}$. A transport plan $\boldsymbol{\pi} \in \mathbb{R}_+^{N \times N}$ moves π_{ij} units of mass from patch i to j. Because p and q may have different total mass, KL penalties on the plan's marginals allow mass to be created or removed (Chizat et al., 2018b). We obtain an entropy-regularized plan and evaluate its transport and mass-change costs:

$$\begin{aligned} \boldsymbol{\pi}^*_{\varepsilon,\rho} &= \arg\min_{\boldsymbol{\pi} \geq 0} \langle \boldsymbol{\pi}, \boldsymbol{C} \rangle + \varepsilon\, \Omega(\boldsymbol{\pi}) + \rho\, \mathrm{KL}(\boldsymbol{\pi}\mathbf{1} \,\|\, p) + \rho\, \mathrm{KL}(\boldsymbol{\pi}^\top \mathbf{1} \,\|\, q), \\ \mathcal{T}_{\varepsilon,\rho}(p,q) &= \langle \boldsymbol{\pi}^*_{\varepsilon,\rho}, \boldsymbol{C} \rangle + \rho\, \mathrm{KL}(\boldsymbol{\pi}^*_{\varepsilon,\rho}\mathbf{1} \,\|\, p) + \rho\, \mathrm{KL}((\boldsymbol{\pi}^*_{\varepsilon,\rho})^\top \mathbf{1} \,\|\, q). \end{aligned} \tag{12}$$

Here, Ω is the entropic regularizer, ε controls the smoothness of the computed plan, and ρ controls the cost of changing mass. We solve for the plan with Sinkhorn iterations (Chizat et al., 2018a) (see Appendix B.3). Combining UOT, mask ℓ_1, and the original representation reconstruction gives

$$\mathcal{L}_{\text{pred}} = \lambda_{\text{uot}} \sum_k \mathcal{T}_{\varepsilon,\rho}(\hat{m}_{t+1,k}, m_{t+1,k}) + \lambda_{\text{mask}} \|\hat{\boldsymbol{M}}_{t+1} - \boldsymbol{M}_{t+1}\|_1 + \lambda_{\text{feat}} \|\hat{\boldsymbol{Z}}_{t+1} - \boldsymbol{Z}_{t+1}\|_1. \tag{13}$$

At test time, the first two terms form the planning objective after replacing the target masks with $\boldsymbol{M}^g$ extracted from the visual goal. Both the current observation and goal are decoded against the same $\boldsymbol{S}$: The goal therefore does not receive an independent slot set, and slot index k remains aligned across current, predicted, and goal masks. We roll out candidate action sequences, optimize them with the cross-entropy method (CEM) (Chua et al., 2018), execute an action chunk, and replan. Appendix B.3 details generalized Sinkhorn scaling, change-aware weighting, and static-slot identification; Appendix C.3 gives the environment-specific obstacle-aware ground cost and MPC settings.

4 Experiments

4.1 Experimental Setup

We conduct experiments around four questions: **1)** Does the slot decomposition preserve the semantic information (Sec. 4.2)? **2)** How well does Slot-JEPA perform on visual planning in controlled planar environments (Sec. 4.3)? **3)** Does the framework generalize to Robotic Arm Planning (Sec. 4.4)? **4)** How does each component contribute to performance (Sec. 4.5)?

Representation pretraining. Starting from a V-JEPA 2.1 encoder (Mur-Labadia et al., 2026), we pretrain the slot decomposition module and TMP on Kinetics-400 videos (Kay et al., 2017) using

Table 1: Visual-understanding results on all three probes.

Method	Segmentation (mIoU↑)			Depth (δ_1 ↑)			VOS ($\mathcal{J}\&\mathcal{F}$ ↑)
	COCO	ADE	VOC	NYUv2	KITTI	SUNRGBD	DAVIS
MoCo v3 (Chen et al., 2021)	43.4	24.4	61.7	69.8	77.2	50.6	61.7
iBOT (Zhou et al., 2022a)	52.2	34.3	73.3	77.7	88.3	69.1	64.3
Mugs (Zhou et al., 2022b)	51.4	34.6	72.7	74.9	87.7	66.6	62.1
MAE (He et al., 2022)	37.9	20.6	52.3	65.3	84.3	59.1	48.9
DINOv2† (Oquab et al., 2024)	53.0	35.5	74.9	78.1	88.0	68.0	64.2
V-JEPA 2.1 (Mur-Labadia et al., 2026)	52.8	35.5	72.1	80.5	**89.2**	70.0	65.9
Slot-JEPA	**56.1**	**38.9**	**75.1**	**83.1**	87.8	**72.3**	**66.6**

inverse transition encoders. We freeze the pretrained visual encoder and slot decomposition module for downstream tasks. The full pretraining setup is given in Appendix C.1.

4.2 Preservation of Semantic Information

For each image or video frame, the frozen visual encoder and the slot decomposition module jointly provide a patch representation for downstream visual understanding tasks. We evaluate linear semantic segmentation on COCOStuff-27 (Caesar et al., 2018), ADE20K (Zhou et al., 2017), and PASCAL VOC (Everingham et al., 2010) to measure category-level structure; monocular depth estimation on NYUv2 (Silberman et al., 2012) and KITTI (Geiger et al., 2012), plus NYUv2-to-SUNRGBD transfer (Song et al., 2015) without adaptation; and VOS on DAVIS-2017 (Pont-Tuset et al., 2017) using zero-shot label propagation. The evaluation settings are detailed in Appendix C.2.

Results in Tab. 1 support two conclusions: **1) Slot decomposition preserves the main task-relevant semantics under strong compression.** With $K = 16$, the factorized slot representation contains only about 2.5% as many scalars as the dense patch field, yet the slot-reconstructed representation retains the semantic information needed across segmentation, depth, and VOS. **2) Slot reconstruction makes the patch representation more object-aware.** During representation pretraining, the slot-reconstruction objective fine-tunes the visual encoder and propagates object-level grouping back into its patch features. This supplies explicit object-aware supervision that is absent from the original V-JEPA 2.1 training objective, helping explain the consistent downstream gains.

4.3 Planar Visual Planning

Evaluation protocol. We evaluate three in-distribution (ID) tasks and their out-of-distribution (OOD) variants. The ID tasks use the same layouts and object shapes during training and testing, whereas the OOD variants use previously unseen layouts or object shapes. In *TwoRoom* (also called *Wall* in prior work) and *Maze* (Zhou et al., 2025), a point agent passes through a doorway and a ball navigates a U-shaped corridor, respectively. Both use position and velocity as r_t, while a_t specifies velocity in TwoRoom and force in Maze. In *PushT* (Florence et al., 2022; Chi et al., 2025), a pusher moves a T-shaped object and itself to their target poses; r_t is the pusher state and a_t is a relative target tracked by a PD controller. The OOD variants *TwoRoomR* and *MazeR* randomize obstacle layouts, while *PushObj* changes the object shape. Each task provides an initial image v_0 and visual goal v_g. For comparison, we report DINO-WM baselines that additionally use the low-dimensional goal state for planning in TwoRoom and PushT. We evaluate 50 cases per environment using MPC (Rawlings et al., 2026) with CEM (de Boer et al., 2005). See Appendix C.3 for more details.

Main results. Tab. 2 reports success rates and indicates whether goal-state input is used. We draw two conclusions: **1) Slot-JEPA improves visual planning in both ID and OOD environments.** Using only a visual goal, it succeeds in 100%, 100%, and 92% of ID cases and 90%, 96%, and 58% of OOD cases. It also outperforms every baseline that receives the low-dimensional goal state in all four environments where goal-state results are available. **2) Slot masks and UOT cost provide a planning signal aligned with progress.** As shown in Fig. 2, the obstacle-aware UOT loss changes gradually even far from the goal, which provides a smoother landscape compared to ℓ_1 losses. Stable slots and object-level mask decomposition provide the foundation for applying UOT for planning. Additional loss landscapes and predicted-mask rollouts are collected in Appendix C.5.

Table 2: Planar-planning success rates (%). Goal State indicates whether the planner also receives the low-dimensional goal state.

Method	Goal State	ID Environments			OOD Environments		
		TwoRoom	Maze	PushT	TwoRoomR	MazeR	PushObj
DINO-CLS (Caron et al., 2021)	✗	6	88	20	34	70	2
	✓	84	–	28	48	–	8
ResNet (He et al., 2016)	✗	6	98	34	12	68	4
	✓	84	–	40	60	–	16
R3M (Nair et al., 2023)	✗	38	**100**	16	8	32	2
	✓	90	–	42	46	–	36
DINO-WM (Zhou et al., 2025)	✗	50	**100**	74	82	86	28
	✓	84	–	88	82	–	40
V-JEPA 2.1 (Mur-Labadia et al., 2026)	✗	50	94	58	50	86	46
	✓	82	–	66	72	–	54
LeWM (Maes et al., 2026)	✗	80	98	88	52	88	30
Slot-JEPA (Ours)	✗	**100**	**100**	**92**	**90**	**96**	**58**

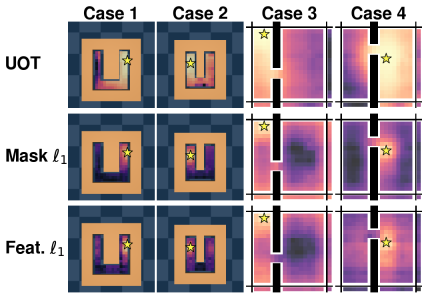

Figure 2: Goal-loss landscapes in MazeR and TwoRoomR. Stars mark goals; lighter colors indicate lower loss.

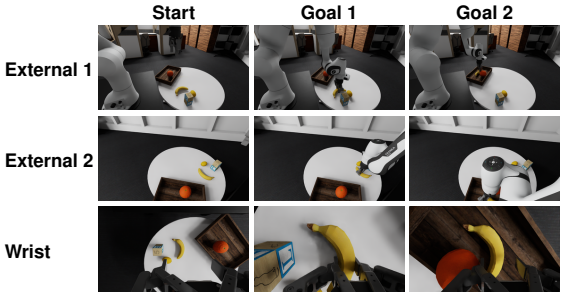

Figure 3: Three-view start and stage-wise goal images for a REALM-JEPA Place task.

Table 3: Robotic arm planning success rates on REALM-JEPA (count and %).

Method	Reach	Pick	Rotate	Place	Overall
V-JEPA 2.1	13/18 (72.2)	19/30 (63.3)	6/30 (20.0)	0/30 (0.0)	38/108 (35.2)
V-JEPA 2.1 + TMP	16/18 (88.9)	22/30 (73.3)	13/30 (43.3)	12/30 (40.0)	63/108 (58.3)
Slot-JEPA	**18/18 (100.0)**	**27/30 (90.0)**	**25/30 (83.3)**	**23/30 (76.7)**	**93/108 (86.1)**

Table 4: Spearman correlations among observed/predicted loss and EE–goal distance.

Method	Obs. loss vs. EE–goal dist.			Pred. loss vs. EE–goal dist.			Pred. loss vs. Obs. loss		
	Ext. 1	Ext. 2	Wrist	Ext. 1	Ext. 2	Wrist	Ext. 1	Ext. 2	Wrist
V-JEPA 2.1	0.350	0.407	0.533	0.503	0.453	0.297	0.511	0.609	0.549
V-JEPA 2.1 + TMP	0.154	0.103	0.652	0.327	0.319	0.454	0.409	0.421	0.720
Slot-JEPA	**0.792**	**0.891**	**0.807**	**0.677**	**0.788**	**0.814**	**0.668**	**0.825**	**0.865**

4.4 Robotic Arm Planning

Dataset & evaluation protocol. Based on the REALM manipulation environment and task configurations (Sedlacek et al., 2026), we construct the **REALM-JEPA** dataset for JEPA-based visual-goal planning. It provides start observations and stage-wise visual goals for 108 cases in four task families: 18 *Reach* cases and 30 each for *Pick*, *Rotate*, and *Place*. We fine-tune the robot dynamics predictor on DROID trajectories (Khazatsky et al., 2024). The training settings are detailed in Appendix C.4.

In each task, we control a 7-DoF robot arm: six dimensions describe the end-effector (EE) pose, and one specifies the gripper opening angle. At step t, the visual observation v_t contains images from two fixed external cameras and one wrist camera. Following V-JEPA 2 (Assran et al., 2025), we provide up to two stage-wise visual goals, each comprising images from the same three views, as illustrated in Fig. 3. The low-dimensional state r_t contains the current EE pose, gripper angle, and camera poses.

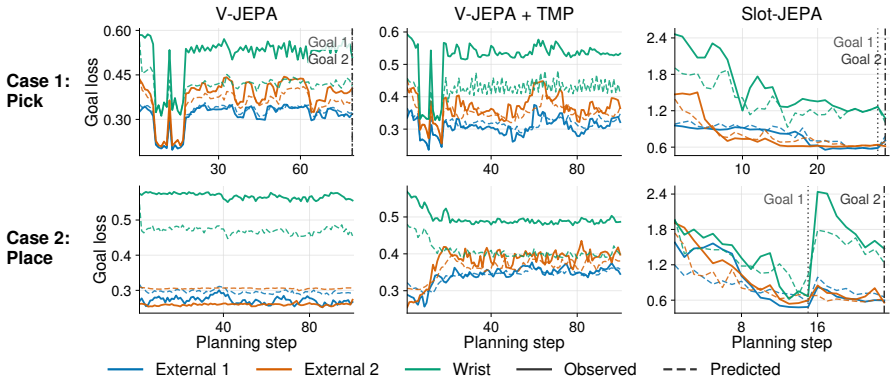

Figure 4: Goal-loss trajectories for Pick and Place. Panels use independent scales.

The action a_t specifies changes to the EE pose and gripper angle. The three views share one predictor, but maintain separate slots, transition memories, and predictions. At each MPC step, the two fixed views first update the CEM distribution; the wrist view then starts from that distribution, continues the search, and selects the action for execution. An inverse-kinematics controller converts the selected absolute EE pose into joint targets, which the arm tracks using PD feedback. Each planning attempt is limited to 100 MPC steps. Since ℓ_1 losses provide weak guidance at large displacements, we use ground-truth target-position guidance for the patch-based baselines until the EE is within 30 cm of the target position. Slot-JEPA uses visual goal only throughout. Full multi-view and guidance details are provided in Appendix C.4.

Quantitative results. Tab. 3 reports success rates for the four task families. To assess whether the visual goal loss reflects physical progress, Tab. 4 reports its Spearman correlation with 7-DoF EE–goal distance, using both losses predicted by the predictor and losses computed from the observations after execution. We also examine how the observed goal loss changes over planning steps in Fig. 4. Appendix C.4 gives the complete correlation definitions and statistics. These results support three conclusions: **1) Slot-JEPA improves success across all four task families, especially Rotate and Place.** Its overall success rate is 86.1%, compared with 35.2% for V-JEPA 2.1 and 58.3% for V-JEPA 2.1 + TMP. On Rotate and Place, Slot-JEPA reaches 83.3% and 76.7%, respectively, compared with 20.0% and 0.0% for V-JEPA 2.1. **2) Pointwise ℓ_1 losses give weak guidance far from the goal.** Despite this ground-truth guidance, both patch-based baselines still fail often. Their losses computed from actual observations also correlate less strongly with EE–goal distance than Slot-JEPA's loss, indicating that predictor error alone cannot explain these failures. **3) Replacing V-JEPA 2.1's predictor with TMP substantially improves planning.** The success rate rises from 35.2% to 58.3% while retaining the patch representation and goal loss. This gain is consistent with TMP retrieving useful information from the full trajectory history, whereas the original predictor relies on temporally contiguous frames and is restricted to nearby observations.

Visualizations. Fig. 5 shows examples from Kinetics-400 pretraining, DROID downstream training, and REALM-JEPA planning inference, respectively. The selected slots remain temporally consistent despite object motion, camera viewpoint changes, and objects leaving the view. Although the slot decomposition is frozen after pretraining, it continues to produce coherent masks in the unseen downstream scenes. Predicted future representations also closely match those extracted from subsequent observations (Fig. 6), supporting TMP's ability to model action-conditioned dynamics. Full-scale slot decompositions, additional UOT landscapes, and predicted-mask rollouts are provided in Appendix C.5.

4.5 Ablation Studies

Tab. 5 compares five configurations. **Line 1**: replace TMP with a V-JEPA 2.1-style sequential predictor (Mur-Labadia et al., 2026) using recent transitions while retaining the mask interface and two-branch design. **Line 2**: remove UOT from planning. **Line 3**: give TMP only recent transitions.



Figure 5: Slot-mask overlays from Kinetics-400, DROID, and REALM-JEPA (top to bottom). Colors track slots across each sequence.

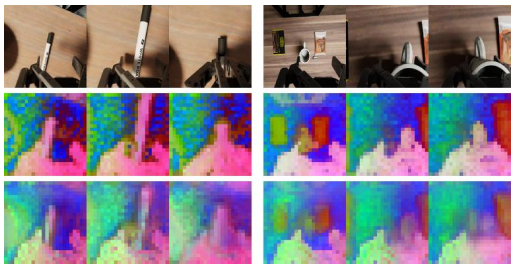

Figure 6: Wrist-view RGB, observed features, and reconstructed features from predicted masks without seeing the frame (top to bottom).

Table 5: Ablation success rates (%) across six planar environments and robot Reach. Bold marks the highest value in each column.

ID	Predictor	Planning loss	Context	TwoRoom	Maze	PushT	TwoRoomR	MazeR	PushObj	Reach
1	Sequential	UOT + mask ℓ_1	Recent transitions	**100**	**100**	76	**94**	**98**	40	–
2	TMP	Mask ℓ_1	Sampled via Eq. (6)	96	92	82	78	90	50	88.9
3	TMP	UOT + mask ℓ_1	Recent transitions	94	86	76	84	80	**58**	55.6
4	TMP	UOT + mask ℓ_1	Random transitions	96	88	80	40	72	48	94.4
5	TMP	UOT + mask ℓ_1	Sampled via Eq. (6)	**100**	**100**	**92**	90	96	**58**	**100.0**

Line 4: use random transitions from the memory bank. **Line 5**: use the full Slot-JEPA model with transitions sampled according to Eq. (6).

These comparisons support three conclusions: **1) UOT is important for planning.** Comparing lines 2 and 5, UOT improves success in every planar environment and robot Reach, even with goal-pose guidance for line 2. **2) Complex planar tasks benefit from TMP's diverse transition context.** Lines 1 and 5 perform comparably on TwoRoom, TwoRoomR, Maze, and MazeR, consistent with recent information being most useful for these simpler tasks. On PushT and PushObj, line 5 performs better, highlighting the value of TMP's access to diverse historical transitions. **3) Context must balance relevance and diversity.** Lines 3 and 5 show the benefit of drawing from a broader history; lines 4 and 5 show that query-conditioned relevance matters, especially under layout shift.

5 Limitations

Slot decomposition is trained without instance or part annotations, using feature reconstruction rather than physical identity as its objective. Its masks support planning, but one slot may merge multiple objects or split one object, so consistent instance- or part-level decomposition is not guaranteed. This can limit reasoning about specific components and contact points. TMP builds memory from observed transitions and retrieves context using fixed state and mask cues. These cues may miss contact or appearance changes and select redundant examples. Better memory construction and retrieval criteria that balance relevance with diversity remain open problems.

6 Conclusion

We introduced Slot-JEPA, which factorizes dense visual representations into stable object-structured slots and time-varying spatial masks for action-conditioned prediction. This structure enables a UOT planning loss that is better aligned with spatial progress when pointwise losses provide weak guidance. TMP further draws on relevant transitions beyond a short contiguous history. Across seven visual-understanding benchmarks, mixtures of encoder and slot-reconstructed features remain transferable, while six planar environments and four Robotic Arm Planning tasks in the REALM-JEPA dataset show improved planning success, especially on the more complex manipulation tasks. These results show the value of combining structured mask prediction, spatially informed goal costs, and transition memory for visual planning.

AI Use Statement

We used generative AI (GPT-5.6 Sol) to polish the manuscript, check grammar, assist with code implementation, and write code for visualizing experimental results. The authors reviewed and edited the AI-assisted text, checked the code and figures against the intended methods and underlying results, and take full responsibility for the final manuscript and artifacts.

Reproducibility Statement

The architecture, notation, and training objectives are defined in Sec. 3. The implementation shared by all tasks is documented in Appendix B. Appendix B.1 specifies slot initialization and decoding. Appendix B.2 describes the TMP architecture, pretraining contexts, inverse encoders, and memory maintenance. Appendix B.3 provides the UOT formulation, Sinkhorn algorithm, and static-slot criterion.

Pretraining can be reproduced from Appendix C.1, which identifies the initial checkpoint, Kinetics-400 data, frame sampling, optimization schedule, hardware, and loss weights. The semantic-segmentation, depth-estimation, and video-object-segmentation evaluations are specified in Appendix C.2, including their datasets, downstream protocols, backbones, metrics, and complete results. Appendix C.3 gives the environment splits, slot and memory initialization, downstream training, obstacle-aware UOT, and MPC settings for each planar task. Appendix C.4 defines the REALM-JEPA cases and DROID fine-tuning setup, then details multi-view planning, goal guidance, correlation metrics, and action execution for each robotic-arm task. The supplementary materials include the implementation code, and Appendix C.5 provides qualitative outputs for checking the reproduced representations and predictions.

References

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. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629, 2023.

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.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *Transactions on Machine Learning Research*, 2024.

Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. COCO-Stuff: Thing and stuff classes in context. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1209–1218, 2018.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9912–9924, 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *IEEE/CVF International Conference on Computer Vision*, pp. 9630–9640, 2021.

Xiaokang Chen, Mingyu Ding, Xiaodi Wang, Ying Xin, Shentong Mo, Yunhao Wang, Shumin Han, Ping Luo, Gang Zeng, and Jingdong Wang. Context autoencoder for self-supervised representation learning. *International Journal of Computer Vision*, 132(1):208–223, 2024. doi: 10.1007/s11263-023-01852-4.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *IEEE/CVF International Conference on Computer Vision*, pp. 9620–9629, 2021.

Ho Kei Cheng and Alexander G. Schwing. XMem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *European Conference on Computer Vision*, pp. 640–658, 2022.

Ho Kei Cheng, Seoung Wug Oh, Brian Price, Joon-Young Lee, and Alexander Schwing. Putting the object back into video object segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3151–3161, 2024.

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. *The International Journal of Robotics Research*, 44(10–11):1684–1704, 2025. doi: 10.1177/02783649241273668.

Lénaïc Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. Scaling algorithms for unbalanced transport problems. *Mathematics of Computation*, 87(314):2563–2609, 2018a.

Lénaïc Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. Unbalanced optimal transport: Dynamic and kantorovich formulations. *Journal of Functional Analysis*, 274(11):3090–3123, 2018b.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, volume 31, pp. 4754–4765, 2018.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26, pp. 2292–2300, 2013.

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

David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems*, pp. 2366–2374, 2014.

Gamaleldin F. Elsayed, Aravindh Mahendran, Sjoerd van Steenkiste, Klaus Greff, Michael C. Mozer, and Thomas Kipf. SAVi++: Towards end-to-end object-centric learning from real-world videos. In *Advances in Neural Information Processing Systems*, pp. 28940–28954, 2022.

Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88:303–338, 2010. doi: 10.1007/s11263-009-0275-4.

Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *IEEE International Conference on Robotics and Automation*, pp. 2786–2793, 2017.

Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in Neural Information Processing Systems*, volume 29, pp. 64–72, 2016.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A. Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 158–168, 2022.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361, 2012.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, volume 31, pp. 2451–2463, 2018.

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*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565, 2019.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.

Yihang Huang, Peisong Wen, Chenhao Zhou, Difu Feng, and Qianqian Xu. When 1+1 backfires: Rethinking relevance-diversity balancing for sample-aware visual token reduction. In *ACM International Conference on Multimedia*, 2026. Accepted.

Rishabh Kabra, Daniel Zoran, Goker Erdogan, Loic Matthey, Antonia Creswell, Matt Botvinick, Alexander Lerchner, and Chris Burgess. SIMONe: View-invariant, temporally-abstracted object representations via unsupervised video decomposition. In *Advances in Neural Information Processing Systems*, volume 34, pp. 20146–20159, 2021.

Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

Alexander Khazatsky et al. DROID: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems*, 2024.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Thomas Kipf, Gamaleldin F. Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff. Conditional object-centric learning from video. In *International Conference on Learning Representations*, 2022.

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*, volume 33, pp. 11525–11538, 2020.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

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

Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mido Assran, Koustuv Sinha, Mike Rabbat, Yann LeCun, Nicolas Ballas, and Adrien Bardes. V-JEPA 2.1: Unlocking dense features in video self-supervised learning. In *European Conference on Computer Vision*, pp. 671–689, 2026. doi: 10.1007/978-3-032-37369-4_36.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3M: A universal visual representation for robot manipulation. In *Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 892–909, 2023.

Heejeong Nam, Quentin Le Lidec, Lucas Maes, Yann LeCun, and Randall Balestriero. Causal-JEPA: Learning world models through object-level latent masking. In *2nd Workshop on Compositional Learning at ICML*, 2026.

Shukrullo Nazirjonov, Sai Prasanna, Anna Manasyan, and Georg Martius. Better slots, better worlds: Representation quality and robustness in object-centric world models. In *Model-Based Reinforcement Learning in the Era of Generative World Models Workshop at RLC*, 2026.

Seoung Wug Oh, Joon-Young Lee, Kalyan Sunkavalli, and Seon Joo Kim. Fast video object segmentation by reference-guided mask propagation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7376–7385, 2018.

Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video object segmentation using space-time memory networks. In *IEEE/CVF International Conference on Computer Vision*, pp. 9226–9235, 2019.

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.

Federico Perazzi, Anna Khoreva, Rodrigo Benenson, Bernt Schiele, and Alexander Sorkine-Hornung. Learning video object segmentation from static images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2663–2672, 2017.

Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 DAVIS challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017.

Ali Pourganjalikhan and Charalambos Poullis. Adaptive memory management for video object segmentation. In *Conference on Robots and Vision*, pp. 75–82, 2022.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloé Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *International Conference on Learning Representations*, 2025.

James B. Rawlings, David Q. Mayne, and Moritz M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2 edition, 2026.

Martin Sedlacek, Pavlo Yefanov, Georgy Ponimatkin, Jai Bardhan, Simon Pilc, Mederic Fourmy, Evangelos Kazakos, Cees G. M. Snoek, Josef Sivic, and Vladimir Petrik. REALM: A real-to-sim validated benchmark for generalization in robotic manipulation. *IEEE Robotics and Automation Letters*, 11(7):8315–8322, 2026. doi: 10.1109/LRA.2026.3692088.

Maximilian Seitzer, Max Horn, Andrii Zadaianchuk, Dominik Zietlow, Tianjun Xiao, Carl-Johann Simon-Gabriel, Tong He, Zheng Zhang, Bernhard Schölkopf, Thomas Brox, and Francesco Locatello. Bridging the gap to real-world object-centric learning. In *International Conference on Learning Representations*, 2023.

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *European Conference on Computer Vision*, pp. 746–760, 2012.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, et al. DINOv3. *arXiv preprint arXiv:2508.10104*, 2025.

Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 567–576, 2015.

Basile Terver, Tsung-Yen Yang, Jean Ponce, Adrien Bardes, and Yann LeCun. What drives success in physical planning with joint-embedding predictive world models? *Transactions on Machine Learning Research*, 2026.

Ying Wang, Oumayma Bounou, Yann LeCun, and Mengye Ren. AdaJEPA: An adaptive latent world model. *arXiv preprint arXiv:2606.32026*, 2026.

Peisong Wen, Qianqian Xu, Siran Dai, Runmin Cong, and Qingming Huang. Semantic concentration for self-supervised dense representations learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(2):1185–1201, 2026. doi: 10.1109/TPAMI.2025.3609758.

Ziyi Wu, Nikita Dvornik, Klaus Greff, Thomas Kipf, and Animesh Garg. SlotFormer: Unsupervised visual dynamics simulation with object-centric models. In *International Conference on Learning Representations*, 2023.

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 633–641, 2017.

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 Learning Representations*, 2025.

Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. iBOT: Image BERT pre-training with online tokenizer. In *International Conference on Learning Representations*, 2022a.

Pan Zhou, Yichen Zhou, Chenyang Si, Weihao Yu, Teck Khim Ng, and Shuicheng Yan. Mugs: A multi-granular self-supervised learning framework. *arXiv preprint arXiv:2203.14415*, 2022b.

Adrian Ziegler and Yuki M. Asano. Self-supervised learning of object parts for semantic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14482–14491, 2022.

Appendix Contents

A Extended Related Work **16**
A.1 Joint-Embedding Predictive World Models . . . 16
A.2 Object-Centric Representation and Dynamics . . . 16
A.3 Memory-Based VOS and Transition Memory . . . 16
A.4 Unbalanced Optimal Transport for Visual Correspondence and Control . . . 17
B Additional Method Details **17**
B.1 Complete Slot Decomposition . . . 17
B.2 Predictor Architecture, Pretraining, and Memory . . . 18
B.3 UOT and Static-Slot Identification . . . 19
C Experimental Details and Complete Results **21**
C.1 Pretraining . . . 21
C.2 Visual Understanding . . . 22
C.3 Planar Planning . . . 25
C.4 Robotic Arm Planning . . . 25
C.5 Additional Visualizations . . . 28

A Extended Related Work

Slot-JEPA connects four lines of work that are often studied separately: joint-embedding representation prediction, object-centric decomposition and dynamics, memory-based video segmentation, and optimal transport. Together, they motivate its slot-and-mask state, transition memory, and spatial planning objective.

A.1 Joint-Embedding Predictive World Models

Joint-embedding methods predict target representations rather than reconstructing raw observations, allowing the learned state to retain predictable semantic structure while discarding some pixel-level detail. I-JEPA introduces this principle for masked image regions, and V-JEPA extends it to spatiotemporal video representations (Assran et al., 2023; Bardes et al., 2024). Dense self-supervised features from DINO and DINOv2 provide another strong latent target for spatial prediction (Caron et al., 2021; Oquab et al., 2024); recent work further studies how the semantic information in such features can be spatially concentrated for dense transfer (Wen et al., 2026). These methods primarily concern how a useful visual representation is learned.

A second line of work turns representation prediction into an action-conditioned world model. DINO-WM predicts future DINOv2 patch features from offline interaction trajectories and plans by comparing imagined features with goal features (Zhou et al., 2025). V-JEPA 2-AC post-trains a frame-causal, action-conditioned predictor for robot planning, and V-JEPA 2.1 improves the spatial density and temporal consistency of the underlying video features (Assran et al., 2025; Mur-Labadia et al., 2026). Subsequent studies isolate the effects of representation, predictive objective, and planner design (Terver et al., 2026); AdaJEPA adapts the encoder and predictor online from newly observed transitions (Wang et al., 2026); and LeWorldModel investigates stable end-to-end training of JEPA world models from pixels (Maes et al., 2026).

Slot-JEPA is complementary to these advances: rather than replacing joint-embedding pretraining, it introduces a structured slot-and-mask interface between a pretrained dense encoder and action dynamics, together with a spatial objective for prediction and planning.

A.2 Object-Centric Representation and Dynamics

Slot Attention maps a set of visual features to a small collection of exchangeable components through competition for input tokens (Locatello et al., 2020). Subsequent methods extend this idea from individual images to video. SIMONe separates temporally abstracted object latents from per-pixel mixture assignments (Kabra et al., 2021); SAVi and SAVi++ use temporally propagated slots to support object tracking in synthetic and real videos (Kipf et al., 2022; Elsayed et al., 2022); and SlotFormer learns dynamics over slot sequences for video prediction and planning (Wu et al., 2023). DINOSAUR shows that reconstructing pretrained self-supervised features, rather than RGB pixels, can induce object-centric components in real images (Seitzer et al., 2023). This last observation is especially relevant to our goal of reorganizing a pretrained feature field without discarding its semantic information.

Object-centric prediction commonly updates the full latent content of every slot at each time step. Causal-JEPA, the closest joint-embedding counterpart, applies masked and future prediction to object-centric latents (Nam et al., 2026); Better Slots, Better Worlds studies how the quality of these latents affects prediction, planning, and robustness (Nazirjonov et al., 2026). In contrast, Slot-JEPA holds a reference-derived semantic slot bank fixed within a prediction context and predicts its spatial masks. As in unsupervised slot learning generally, these slots are object-centric components rather than guaranteed physical-instance segmentations.

A.3 Memory-Based VOS and Transition Memory

Semi-supervised Video Object Segmentation (VOS) must preserve the identity of a target while its appearance and location change. Early propagation methods predict the next mask from an initial annotation and recent frames or masks (Perazzi et al., 2017; Oh et al., 2018). Space-Time Memory Networks instead store past image–mask pairs and let the current query read from them (Oh et al., 2019). XMem separates working, long-term, and sensory memories for long videos, while adaptive

memory methods retain frames according to their utility rather than keeping every observation (Cheng & Schwing, 2022; Pourganjalikhan & Poullis, 2022). Cutie introduces object-level memory reading, and SAM 2 develops a streaming memory architecture for promptable segmentation (Cheng et al., 2024; Ravi et al., 2025). Collectively, these methods show that a query can benefit from selected non-contiguous history instead of only the immediately preceding frame.

TMP transfers this memory principle from perceptual propagation to controlled prediction. Unlike VOS memories that store evidence for locating a target, TMP stores observed source–action–target transitions and retrieves potentially non-contiguous examples to predict a new action outcome.

A.4 Unbalanced Optimal Transport for Visual Correspondence and Control

Pointwise distances compare features or masks only at matching grid locations. They can therefore provide little information about *how far* mass should move when predicted and target supports do not overlap. Optimal transport instead compares nonnegative spatial measures under a ground cost, explicitly charging for displacement between source and target locations. Entropic regularization makes the problem amenable to efficient Sinkhorn scaling (Cuturi, 2013). Unbalanced optimal transport (UOT) further relaxes exact marginal matching by assigning a finite penalty to the creation or removal of mass (Chizat et al., 2018b;a). This relaxation is appropriate for visual masks, whose total mass can change with occlusion, viewpoint, scale, or imperfect decomposition.

Slot-JEPA applies UOT between masks with the same slot index, using their shared correspondence to preserve semantic grouping. UOT complements pointwise and reconstruction losses during training and supplies a displacement-aware goal cost during planning; its ground cost can additionally incorporate obstacle geometry in planar environments.

B Additional Method Details

This appendix gives the architecture and procedures shared by pretraining and all downstream tasks. Environment-specific slot and memory initialization, UOT settings, and planning procedures are reported in Appendix C.

B.1 Complete Slot Decomposition

Reference features and initialization. During pretraining, the reference set $\mathcal{V}^S$ consists only of the observed/context frames in the sampled clip; none of the target frames to be predicted are used to construct the slots. We encode these frames once and partition their patch features into non-overlapping windows spanning 4×4 spatial patches. Within each window, four input patch tokens at evenly spaced valid positions initialize four local slots. This feature-dependent initialization avoids introducing a separate learned slot template while retaining spatial diversity.

Local-to-global Slot Attention. Each local window is processed by three Slot Attention iterations. In every iteration, layer-normalized slots form queries and layer-normalized patch tokens form keys and values. Attention is normalized first across the competing slots and then across the input tokens assigned to each slot. The resulting weighted value is passed through a GRU update followed by a residual two-layer MLP, as formalized in Sec. 3.2. We pool the local slots from all windows and select $K = 16$ of them by farthest-point sampling (FPS) in feature space. FPS starts from a valid local slot and repeatedly chooses the slot with the largest distance to its nearest selected slot. The selected features initialize a second, global Slot Attention module, which applies the same three-step iterative update to the pooled local-slot candidates and produces the shared slot bank $\boldsymbol{S}$. The complete local-to-global extraction and mask-decoding network is shown in Fig. 7.

Mask decoding and optimization. The mask decoder first projects patch features and slots to width 128. Two decoder blocks alternately update the patch queries by cross-attending to the fixed slot bank and by applying self-attention among patches. For every patch–slot pair, an MLP receives the concatenated decoded patch feature and slot feature and produces an assignment logit. A softmax across the K slots yields $\boldsymbol{M}$, and the mask-weighted slot sum reconstructs the original dense representation as in Eq. (1). We jointly optimize the local and global Slot Attention modules and the mask

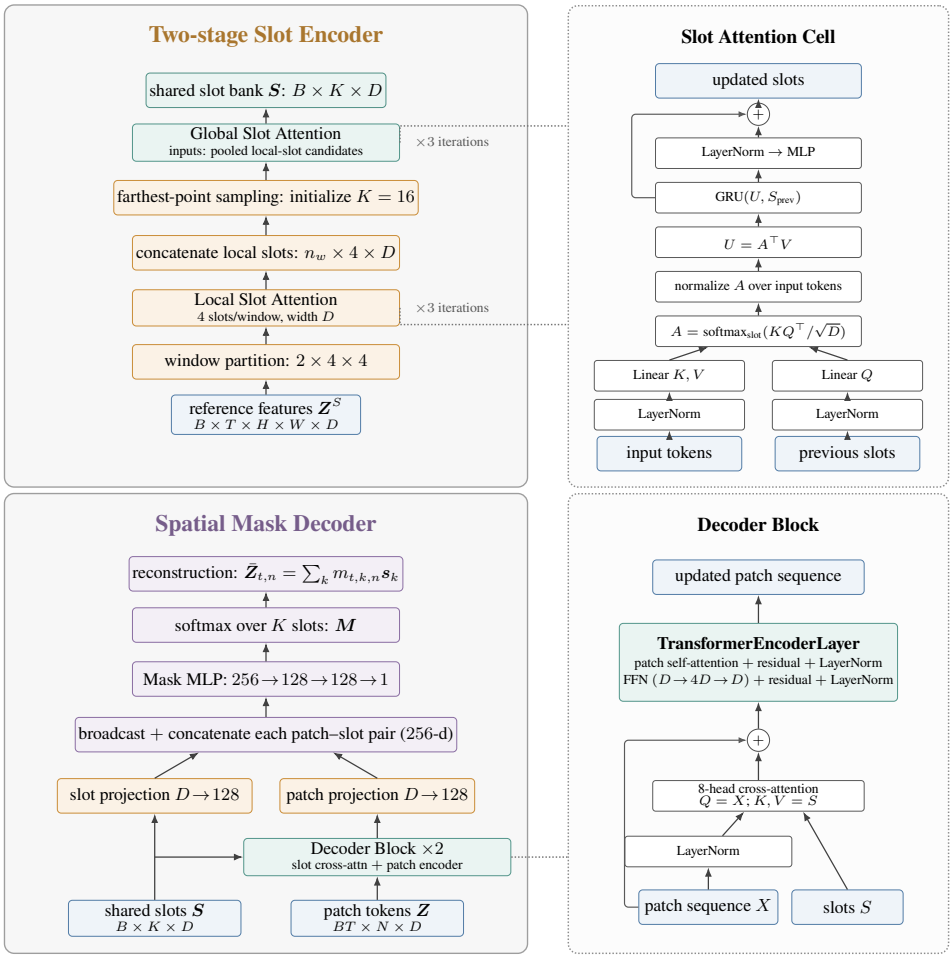

Figure 7: Network architecture of the slot decomposition module.

decoder with the feature-reconstruction and slot-orthogonality terms in Sec. 3.2. The same extracted slot bank is held fixed when decoding the observed, predicted, and goal frames of one environment. The environment-specific choice of inference reference frames and the slot-refresh policy are given in Appendix C.

B.2 Predictor Architecture, Pretraining, and Memory

Predictor architecture. Fig. 8 shows the layer-level architecture of the two TMP branches. TMP uses branch-specific context–query stacks. The reconstructed-feature branch has 12 blocks of width 192 with three attention heads. The mask branch applies separate object-wise predictors at three spatial scales, each with four blocks. Their (spatial size, width, heads) configurations are $(16, 48, 3)$, $(12, 96, 3)$, and $(6, 196, 4)$. The current and future fields are concatenated channel-wise at each scale; the unavailable future query field is replaced by learned tokens. After attention, we retain only the query positions associated with those future tokens, upsample the three outputs, and concatenate them to form $\boldsymbol{H}_{t+1}^{\text{mask}}$. The shared feature branch uses the same context–query organization on the reconstructed feature field and masks to model interactions across slots. Its output is fused with every object-wise mask feature before the slot-wise softmax. These dimensions are fixed in comparisons with the sequential-predictor ablation unless stated otherwise.

Bidirectional pretraining contexts. Each pretraining sample contains seven frames: four observed/context frames and three held-out target frames. We construct both directions from every clip. In the forward case, the first four frames predict the last three; in the backward case, the last four

predict the first three. Thus the predictor learns a relation between observed transitions and a query transition without relying on a fixed chronological direction. The observed transitions from the same sampled clip directly form the training context, so pretraining neither retrieves from nor maintains a persistent memory bank.

Inverse condition encoders. Natural pretraining videos have no action labels. We therefore reconstruct each pair of source and target frames from their masks and shared slots, and feed the paired reconstructions and masks to an inverse transition encoder. A spatial projection, learned condition tokens, and an eight-layer, three-head Transformer of width 192 compress their change into the low-dimensional pseudo-action $\tilde{a}_t$ in Sec. 3.3. A companion Transformer with the same dimensions maps each reconstructed frame and its masks to a pseudo-state. The target reaches TMP through these low-dimensional condition tokens rather than through the full target mask field. After pretraining, both inverse encoders are discarded and replaced by task-specific state and action encoders; the TMP body is transferred.

Memory construction and maintenance at inference. All entries in an inference memory bank are observed transitions decoded with one common slot bank. An entry stores $(\boldsymbol{M}_i, r_i, a_i, \boldsymbol{M}_{i+1}, r_{i+1})$, and each newly executed transition is appended after its next observation becomes available. For a query, TMP computes relevance from normalized low-dimensional state, per-slot area, and per-slot centroid, then samples five transitions without replacement using Eq. (6). One sampled context set is shared by all candidate actions in the same prediction request. When the slot bank is refreshed, retained frames are decoded again with the new slots and the associated memory masks are rebuilt, preserving slot-index consistency. The source of the initial nonempty bank and the refresh schedule depend on the environment and are specified in Appendix C.

B.3 UOT and Static-Slot Identification

UOT computation. The UOT term in Sec. 3.4 is evaluated independently for corresponding slot masks. We retain each mask's original mass: the masks are neither divided by their areas nor constrained to have equal total mass. Instead, the relaxed marginal constraints allow mass to disappear or emerge at finite cost. For nonnegative vectors x and y, we use the generalized KL divergence

$$\mathrm{KL}(x \,\|\, y) = \sum_i \left[x_i \log \frac{x_i}{y_i} - x_i + y_i \right]. \tag{14}$$

The default ground cost is $c_{ij} = \|\boldsymbol{\xi}_i - \boldsymbol{\xi}_j\|_2^2$ on normalized patch coordinates. Given $p, q \in \mathbb{R}_+^N$, we form the Gibbs kernel and unbalanced exponent

$$K_{ij} = \exp(-c_{ij}/\varepsilon), \qquad \theta = \frac{\rho}{\rho + \varepsilon}. \tag{15}$$

Algorithm 1 gives the implemented scaling procedure. All divisions and logarithms clamp their arguments below by $\delta = 10^{-8}$ for numerical stability. The entropy parameter ε enters through K and the scaling exponent; after the final iteration, we evaluate the spatial transport and relaxed-marginal terms of Eq. (12) at the resulting plan.

Algorithm 1 Generalized Sinkhorn scaling for one aligned slot-mask pair

Input: masks $p, q \in \mathbb{R}_+^N$, ground cost $C \in \mathbb{R}_+^{N \times N}$, entropy ε, mass penalty ρ, iteration count L, and numerical floor δ.

1. Set $a = \max(p, \delta)$, $b = \max(q, \delta)$, $K = \max(\exp(-C/\varepsilon), \delta)$, and $\theta = \rho/(\rho + \varepsilon)$ element-wise.
2. Initialize $u = \mathbf{1}_N$ and $v = \mathbf{1}_N$.
3. For $\ell = 1, \ldots, L$, update

$$u \leftarrow \left(\frac{a}{\max(Kv, \delta)} \right)^{\theta}, \qquad v \leftarrow \left(\frac{b}{\max(K^\top u, \delta)} \right)^{\theta}.$$

4. Construct $\pi = \mathrm{diag}(u) K \,\mathrm{diag}(v)$.

Return: $\langle \pi, C \rangle + \rho\,\mathrm{KL}(\pi \mathbf{1} \,\|\, a) + \rho\,\mathrm{KL}(\pi^\top \mathbf{1} \,\|\, b)$.

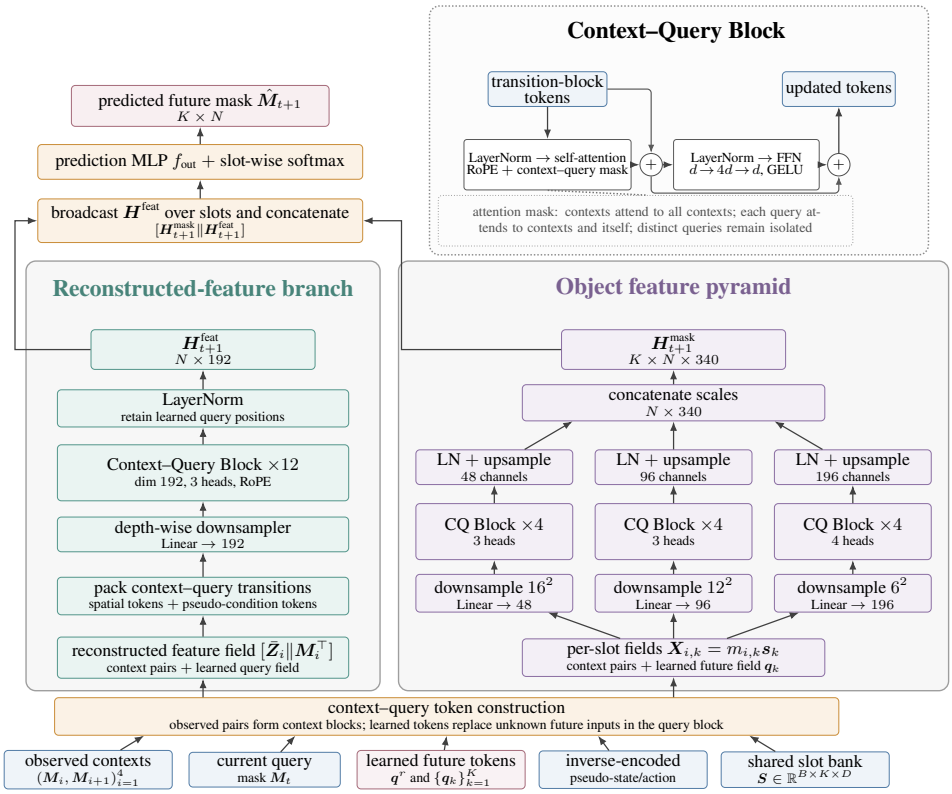

Figure 8: TMP architecture. Observed transitions form the context blocks, while inverse-encoded pseudo-action and learned future tokens specify the query transition. The reconstructed-feature tower and the three-scale object feature pyramid run in parallel and are fused to predict the future slot mask.

To emphasize slots that must change between the reference mask p^{ref} and target mask q, we compute weights only from observed or target masks, never from the prediction:

$$d_k = \frac{1}{N}\|q_k - p_k^{\text{ref}}\|_1, \quad \bar{d} = \frac{1}{K}\sum_j d_j,$$
$$w_k = \text{clip}\left(\frac{d_k}{\bar{d}}, w_{\min}, w_{\max}\right), \quad \mathcal{L}_{\text{uot}} = \frac{\sum_k w_k \mathcal{T}_{\varepsilon,\rho}(p_k, q_k)}{\sum_k w_k}. \tag{16}$$

If all d_k vanish, we set all weights to one. We use $w_{\min} = 0.05$ and $w_{\max} = 5$; the same weights are applied to the mask-ℓ_1 term. This reduction prevents persistent background slots from dominating the objective.

Static-slot identification. We determine static slots from predicted mask changes rather than semantic labels. For each probe transition p and slot k, we compare an aligned source mask m_k with its model-predicted next mask $\hat{m}_k^{(p)}$. We retain the L largest absolute patch-wise changes and average them. The model-level change score is

$$\Delta_k = \frac{1}{P}\sum_{p=1}^{P}\frac{1}{L}\sum_{n \in \text{TopL}(|\hat{m}_k^{(p)} - m_k|)} |\hat{m}_{k,n}^{(p)} - m_{k,n}|. \tag{17}$$

A slot is static when $\Delta_k \le \tau_{\text{static}}$ and moving when $\Delta_k > \tau_{\text{moving}}$; slots in the gap are left unclassified. This two-threshold rule prevents borderline changes from contaminating either group. As illustrated in Fig. 9, moving slots shift with controlled entities, whereas static slots remain aligned with persistent scene geometry.

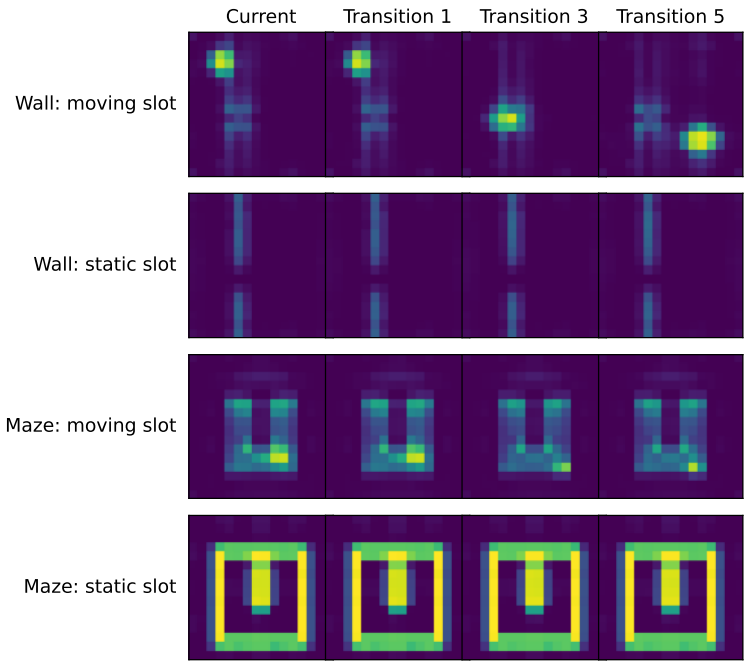

Figure 9: Examples of slots identified from their predicted changes. Moving-slot masks follow the controlled entity across probe transitions, while static-slot masks remain aligned with the wall or maze geometry. The labels are produced by the change-score rule rather than ground-truth segmentation.

For obstacle-aware planning, we peak-normalize and optionally sharpen the selected static masks, then take their maximum to obtain an obstacle map. For a moving slot, this map modifies the ground cost as

$$c_{ij}^{\text{obs}} = c_{ij} + \lambda_{\text{obs}} r_{ij}, \tag{18}$$

where r_{ij} measures obstacle occupancy along the path from patch i to patch j. The ground cost therefore encodes obstacle geometry directly. The probe transitions, thresholds, and ground-cost variant are environment-specific and are reported with the corresponding experiments.

C Experimental Details and Complete Results

This appendix follows the four stages of the empirical study: representation pretraining, visual-understanding evaluation, planar planning, and robotic arm planning. We collect qualitative analyses in a final visualization subsection so that the representation, planning objective, and predicted rollouts can be compared in one place.

C.1 Pretraining

Initialization and data. We initialize the visual encoder from the released distilled V-JEPA 2.1 ViT-B/16 checkpoint (Mur-Labadia et al., 2026). This 80M-parameter encoder is distilled from the 2B-parameter V-JEPA 2.1 ViT-G teacher on VisionMix-163M, which combines LVD-142M images with videos from SSv2, Kinetics, HowTo100M, and YT-1B. We then pretrain Slot-JEPA on the training split of Kinetics-400 (Kay et al., 2017), which contains roughly 240,000 clips. The complete dataset contains 306,245 clips from 400 human-action classes. Each clip lasts approximately 10 seconds and comes from a distinct YouTube video. Its content spans individual actions such as drawing and drinking, person–person interactions such as hugging and shaking hands, and person–object interactions such as opening objects and playing instruments. We decode the videos at 384×384 resolution.

Optimization. We use $K = 16$ global slots and 4×4 spatial windows with four local slots per window. Each seven-frame clip supplies four observed frames and three target frames; the observed transitions provide context for predicting the targets. We jointly optimize the slot decomposition module, TMP, and the inverse transition encoders while fine-tuning the final six visual-encoder blocks; the remaining blocks are frozen. Pretraining runs for 10 epochs on eight RTX 4090 GPUs and takes about 30 hours. We use AdamW (Loshchilov & Hutter, 2019) with learning rate 10^{-4} and weight decay 10^{-2}. The learning rate follows cosine decay (Loshchilov & Hutter, 2017) to 10^{-5}. We set $\lambda_{\text{orth}} = 0.001$ in $\mathcal{L}_{\text{slot}}$ and $(\lambda_{\text{uot}}, \lambda_{\text{mask}}, \lambda_{\text{feat}}) = (3.0, 3.0, 0.1)$ in $\mathcal{L}_{\text{pred}}$. UOT uses five Sinkhorn iterations and $\varepsilon = 0.05$. For all downstream tasks, the pretrained encoder and slot decomposition module are frozen.

C.2 Visual Understanding

Downstream protocols. The encoder and decomposition module remain frozen for all three probes. Segmentation trains a linear head for 2,000 steps using Adam (Kingma & Ba, 2015) with batch size 256, selecting the learning rate from $\{10^{-4}, 3\times10^{-4}, 8\times10^{-4}, 10^{-3}, 2\times10^{-3}, 3\times10^{-3}, 10^{-2}\}$. Depth trains a linear scale-invariant logarithmic loss (SigLoss) head (Eigen et al., 2014) for 38,400 steps (AdamW, learning rate 3×10^{-4}, batch size 16). VOS requires no training: nearest-neighbor propagation uses the latest 20 frames, top-5 neighbors, and 16 frames to initialize slots from first-frame mask centers.

Competitors. We compare against MoCo v2 (Chen et al., 2020), MoCo v3 (Chen et al., 2021), SwAV (Caron et al., 2020), DINO (Caron et al., 2021), iBOT (Zhou et al., 2022a), Mugs (Zhou et al., 2022b), Leopart (Ziegler & Asano, 2022), Context Autoencoder (CAE) (Chen et al., 2024), and Masked Autoencoders (MAE) (He et al., 2022). The locally evaluated V-JEPA 2.1 (Mur-Labadia et al., 2026) is our initialization baseline. In Tables C1–C2, only the two DINOv2 (Oquab et al., 2024) models pretrained on LVD-142M are treated as gray scale references; the reproduced DINOv2 models trained on ImageNet-1K remain in the ranked comparison. In Table C3, only the 7B-parameter DINOv3 (Siméoni et al., 2025) model is excluded as a scale reference.

Results. As shown in Tab. C1, Slot-JEPA consistently improves over its V-JEPA 2.1 initialization on all three segmentation datasets. The mIoU gains are 3.3% on COCOStuff-27, 3.4% on ADE20K, and 3.0% on PASCAL VOC, with corresponding improvements in pixel accuracy; it also achieves the strongest results across all six metrics among the comparison models, excluding the gray reference rows. Tab. C2 shows improvements on NYUv2 and on transfer from NYUv2 to SUNRGBD. On NYUv2, δ_1 increases from 80.5% to 83.1% while AbsRel and RMSE decrease from 0.139 and 0.486 to 0.130 and 0.456. All three transfer metrics also improve, although performance on KITTI decreases slightly. Finally, Tab. C3 reports an increase from 65.9% to 66.6% in DAVIS $\mathcal{J}\&\mathcal{F}$, with gains in both region similarity and contour accuracy. This is the strongest result among the comparison entries, supporting that slot reconstruction preserves temporally consistent object information.

Table C1: Complete linear semantic-segmentation results (%). "Data" and "Ep." denote representation-pretraining data and epochs; IN1K, COCO, K400, and VM163M denote ImageNet-1K, COCO, Kinetics-400, and VisionMix-163M. † marks reproduced DINOv2.

				COCOStuff-27		ADE20K		PASCAL VOC	
Method	Backbone	Data	Ep.	mIoU↑	Acc.↑	mIoU↑	Acc.↑	mIoU↑	Acc.↑
DINOv2	ViT-S/14	LVD-142M	–	57.4	77.7	43.5	78.9	79.4	95.3
DINOv2	ViT-B/14	LVD-142M	–	59.1	78.4	46.3	79.8	81.3	95.7
MoCo v2	RN-50	IN1K	800	37.6	64.0	20.0	61.2	53.3	88.4
SwAV	RN-50	IN1K	800	42.2	68.0	23.3	65.8	55.4	89.0
DINO	RN-50	IN1K	800	40.7	66.6	22.4	64.5	53.1	88.4
MoCo v3	ViT-S/16	IN1K	300	43.4	70.6	24.4	70.5	61.7	91.4
DINO	ViT-S/16	IN1K	800	39.5	67.1	23.1	67.4	50.0	88.4
iBOT	ViT-S/16	IN1K	800	47.7	72.7	28.5	72.3	66.7	92.4
Mugs	ViT-S/16	IN1K	800	48.7	72.6	30.9	72.8	71.8	93.3
Leopart	ViT-S/16	IN1K+COCO	800+50	48.8	73.2	29.7	72.1	68.2	92.6
DINOv2†	ViT-S/14	IN1K	800	48.1	72.7	29.6	72.6	68.3	92.7
CAE	ViT-B/16	IN1K	800	45.5	71.0	27.0	71.9	64.9	92.2
CAE	ViT-B/16	IN1K	1600	46.2	71.6	27.5	72.2	65.6	92.3
MAE	ViT-B/16	IN1K	800	37.9	65.5	20.6	66.2	52.3	88.8
MoCo v3	ViT-B/16	IN1K	300	48.2	73.0	29.6	72.9	69.6	92.9
DINO	ViT-B/16	IN1K	800	45.7	71.5	28.8	72.0	63.6	91.6
DINO	ViT-B/8	IN1K	800	47.0	72.3	30.5	72.7	65.3	92.3
iBOT	ViT-B/16	IN1K	400	52.2	75.3	34.3	75.2	73.3	93.9
Mugs	ViT-B/16	IN1K	400	51.4	74.1	34.6	74.5	72.7	93.5
DINOv2†	ViT-B/14	IN1K	400	53.0	75.6	35.5	75.8	74.9	93.9
V-JEPA 2.1	ViT-B/16	VM163M	–	52.8	75.8	35.5	76.0	72.1	93.8
Slot-JEPA	ViT-B/16	K400	10	**56.1**	**77.8**	**38.9**	**77.8**	**75.1**	**94.5**

Table C2: Complete monocular-depth and NYUv2-to-SUNRGBD transfer results. † marks reproduced DINOv2.

		NYUv2			KITTI			NYU→SUNRGBD		
Method	Backbone	δ_1 ↑	AbsRel↓	RMSE↓	δ_1 ↑	AbsRel↓	RMSE↓	δ_1 ↑	AbsRel↓	RMSE↓
DINOv2 (LVD-142M)	ViT-S/14	84.8	0.134	0.437	91.0	0.096	3.119	77.4	0.269	0.473
DINOv2 (LVD-142M)	ViT-B/14	87.8	0.120	0.408	92.0	0.092	2.931	80.7	0.255	0.443
MoCo v2	RN-50	62.9	0.230	0.719	82.4	0.130	4.319	55.5	0.366	0.749
SwAV	RN-50	61.9	0.225	0.721	84.3	0.123	3.985	55.0	0.367	0.753
DINO	RN-50	61.6	0.228	0.726	83.7	0.127	4.068	54.7	0.370	0.755
MoCo v3	ViT-S/16	69.8	0.192	0.615	77.2	0.170	5.007	50.6	0.411	0.815
DINO	ViT-S/16	67.7	0.202	0.651	83.9	0.132	4.061	59.6	0.343	0.678
iBOT	ViT-S/16	72.1	0.183	0.584	85.3	0.125	3.907	63.5	0.330	0.634
Mugs	ViT-S/16	72.3	0.179	0.586	84.1	0.132	3.963	64.4	0.306	0.613
DINOv2†	ViT-S/14	72.7	0.184	0.583	86.6	0.112	3.757	63.2	0.332	0.631
MAE	ViT-B/16	65.3	0.224	0.673	84.3	0.123	4.087	59.1	0.350	0.688
MoCo v3	ViT-B/16	72.5	0.181	0.585	85.3	0.122	3.954	65.9	0.308	0.606
DINO	ViT-B/16	71.7	0.182	0.593	86.5	0.117	3.771	62.8	0.336	0.644
DINO	ViT-B/8	72.8	0.178	0.579	86.9	0.115	3.697	65.1	0.314	0.613
iBOT	ViT-B/16	77.7	0.160	0.523	88.3	0.110	3.514	69.1	0.299	0.565
Mugs	ViT-B/16	74.9	0.164	0.550	87.7	0.111	3.620	66.6	0.294	0.585
DINOv2†	ViT-B/14	78.1	0.160	0.519	88.0	0.112	3.502	68.0	0.310	0.570
V-JEPA 2.1	ViT-B/16	80.5	0.139	0.486	**89.2**	**0.102**	**3.481**	70.0	0.292	0.557
Slot-JEPA	ViT-B/16	**83.1**	**0.130**	**0.456**	87.8	0.107	3.705	**72.3**	**0.278**	**0.524**

Table C3: Complete DAVIS video object segmentation results (%). MoCo v3, SwAV, and MAE report best-checkpoint $\mathcal{J}\&\mathcal{F}$ only.

Method	Backbone	$\mathcal{J}\uparrow$	$\mathcal{F}\uparrow$	$\mathcal{J}\&\mathcal{F}\uparrow$
DINOv3	ViT-7B/16	–	–	71.1
MoCo v3	ViT-S/16	–	–	61.7
SwAV	RN-50	–	–	41.3
DINO	ViT-B/16	61.5	64.9	63.2
iBOT	ViT-B/16	63.2	65.5	64.3
MAE	ViT-S/16	–	–	48.9
CAE	ViT-L/16	62.8	67.5	65.1
iBOT	ViT-L/16	62.1	65.5	63.8
Mugs	ViT-L/16	60.3	63.0	62.1
DINOv2	ViT-L/14	62.2	66.3	64.2
V-JEPA 2.1	ViT-B/16	63.4	68.3	65.9
Slot-JEPA	ViT-B/16	**63.5**	**69.7**	**66.6**

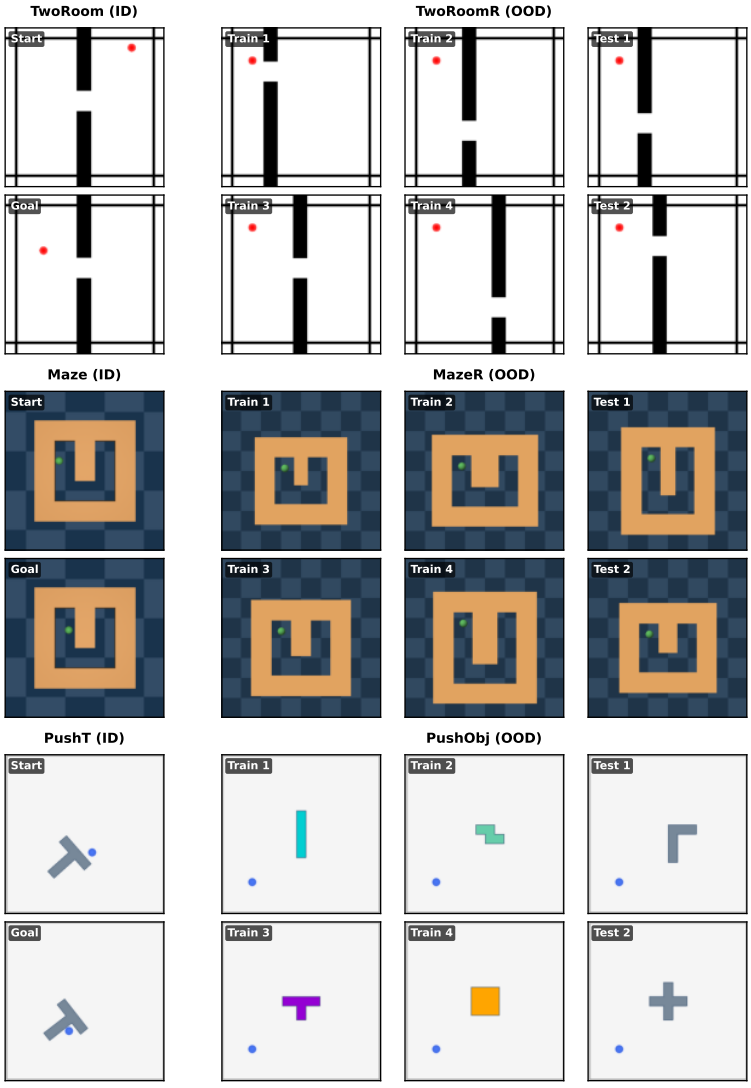

Figure 10: The three planar ID/OOD environment pairs.

Table C4: Training settings for the planar environments.

Training environment	Resolution	Epochs	Batch size	Weight decay
TwoRoom / Maze	384	100	128	10^{-5}
PushT	512	100	128	10^{-5}
TwoRoomR / MazeR	384	50	128	10^{-2}
PushObj	512	500	64	10^{-4}

C.3 Planar Planning

Environment examples. Fig. 10 organizes the six environments into three ID/OOD pairs. Each pair forms a 2×4 grid of equally sized images: the first column contains the ID start and goal, the middle two columns contain four OOD training instances, and the final column contains two held-out OOD test instances. For TwoRoomR and MazeR, these are representative subsets of the training and test layouts; PushObj shows its complete shape split.

PushObj is trained with turquoise I, aquamarine Z, violet small-T, and orange square objects, and evaluated on the held-out L and plus shapes. Object poses vary within both splits; shape identity defines the ID/OOD distinction.

Competitors. The planar baselines use LeWorldModel (LeWM) (Maes et al., 2026), DINO-WM (Zhou et al., 2025), V-JEPA 2.1 (Mur-Labadia et al., 2026), DINO class-token representations (Caron et al., 2021), ResNet (He et al., 2016), and R3M (Nair et al., 2023). We report each method with and without the low-dimensional goal state when those results are available.

Slot and memory initialization. All planar cases use one fixed slot bank extracted from a small subset of training trajectories. In-distribution cases initialize the transition memory with randomly sampled training transitions. For an OOD case, we first execute random actions in the new simulated layout or with the new object shape and use the resulting transitions to initialize its memory. These observations are collected before the evaluated goal-directed rollout and decoded with the same fixed slots. Each transition subsequently observed during planning is appended to the bank.

Downstream fine-tuning. We freeze the visual encoder and slot decomposition module and optimize TMP with task-specific state and action encoders using AdamW. The learning rate is 5×10^{-4} for both transferred and new predictor parameters, using the AdamW defaults $(\beta_1, \beta_2) = (0.9, 0.999)$ and $\epsilon = 10^{-8}$; no learning-rate schedule is applied. Training uses bfloat16, seed 0, and at most 1,000 iterations per epoch. Each training query predicts $\boldsymbol{M}_{t+1}$ from $(\boldsymbol{M}_t, r_t, a_t)$ using five observed context transitions from the same sampled training segment. During planning, five context transitions ($|\mathcal{C}_t| = 5$) are sampled from the 50 nearest memory candidates. We set $(\lambda_{\text{uot}}, \lambda_{\text{mask}}, \lambda_{\text{feat}}) = (1, 10, 1)$ in $\mathcal{L}_{\text{pred}}$, except for Push-Object, where we use $(0.1, 10, 1)$ together with a Sobel mask-boundary term of weight 10. UOT uses five Sinkhorn iterations, $\varepsilon = 0.01$, and $\rho = 0.2$.

Obstacle-aware UOT and MPC. For TwoRoom, TwoRoomR, Maze, and MazeR, the static and moving slot sets are estimated once from 128 deterministic training-transition pairs using the change score in Appendix B.3; the resulting classification is then shared across test cases. We use the ten largest patch changes per transition, with $\tau_{\text{static}} = 0.05$ and $\tau_{\text{moving}} = 0.15$. Their UOT ground cost penalizes paths through the static-mask obstacle map. PushT and PushObj use the standard Euclidean ground cost. For each MPC step, the sampled memory contexts are fixed across CEM candidates. TMP rolls out every candidate action sequence, CEM ranks it with the UOT and mask-ℓ_1 goal loss, and the agent executes the first action before observing the next state and replanning.

C.4 Robotic Arm Planning

Environment and benchmark. We construct **REALM-JEPA** from the REALM simulator and task configurations (Sedlacek et al., 2026) to evaluate visual-goal planning in tabletop manipulation. The environment uses a mounted Franka Panda arm with a Robotiq gripper. At every step, it synchronously renders 1280×720 RGB observations from two fixed external cameras and one wrist-

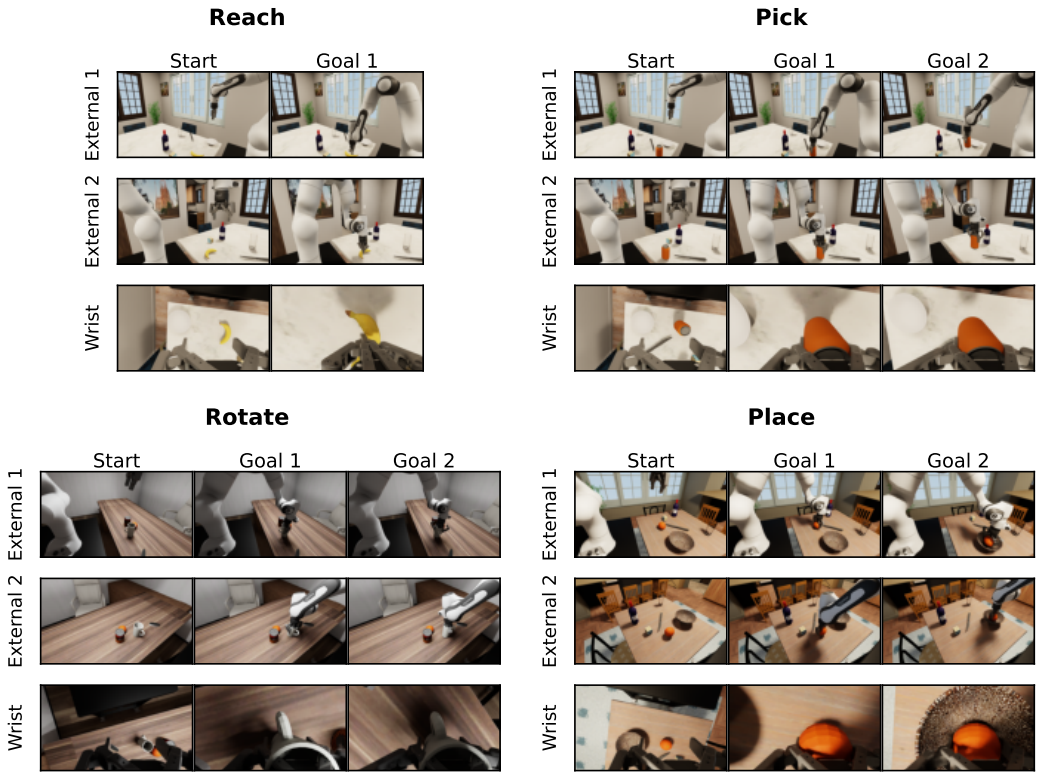

Figure 11: Representative REALM-JEPA start and goal images for all four Robotic Arm Planning categories. Each task panel shows the same scene and goal stages from the two external cameras and the wrist camera. Goal 1 and Goal 2 denote successive visual goals when a task has multiple stages.

mounted camera. The robot state comprises a 6-D end-effector (EE) pose and the gripper opening, and the 7-D action specifies an absolute EE target pose and gripper command. REALM-JEPA contains 108 cases across four task families. *Reach* includes 18 targets corresponding to the six Pick objects, six Rotate objects, and six objects used in Place. Each of the other families contains six task definitions under five scene configurations—the nominal scene, two shifted target-object poses, a block-distractor scene, and a shifted-distractor scene—giving 30 Pick, 30 Rotate, and 30 Place cases. Pick uses a banana, bowl, soda can, lemon, marker, or spoon; Rotate uses a banana, soda can, marker, mug, toy die, or water bottle; and Place uses the six object–receptacle pairs banana–box, soda can–bowl, green block–bowl, lemon–tray, marker–bowl, and orange–bowl. In total, the benchmark covers 11 manipulated-object types and three receptacle types, with $18 + 3 \times (6 \times 5) = 108$ cases.

Each benchmark case fixes the task configuration and initial simulator state, and provides synchronized visual goals from all three cameras. These goals are sampled from a scripted successful reference trajectory; the reference actions and intermediate states are used only to construct the benchmark and are unavailable to the planner. Fig. 11 shows representative start and goal observations for every task family. Reach moves the EE to the target object and uses one visual goal. Pick reaches and grasps the object before lifting it, Rotate grasps and changes its orientation, and Place grasps, transports, and releases it into the designated receptacle; these longer tasks use two successive visual goals. Every goal contains the same three synchronized views as the current observation.

Competitors. We compare with the standard V-JEPA 2.1 predictor (Mur-Labadia et al., 2026) and V-JEPA 2.1 + TMP, which keeps patch representations and the original goal loss but uses our transition-memory architecture. This isolates the effect of the predictor design from the slot-based state interface. On place and rotate tasks, both patch-based baselines use the ground-truth target EE position to bias CEM proposals while the 3-D positional distance exceeds 30 cm. Specifically, the proposal mean is initialized with 20% of the target displacement and clipped per axis to 20% of

the action cap. At 30 cm or closer, the bias is disabled and planning uses the visual goal loss alone. Slot-JEPA does not use this guidance.

Slot and memory initialization. Each REALM-JEPA case begins with a short random-action warmup. The initial frame, provided goal images, and warmup observations form the reference set for each view's slot extraction; the observed warmup transitions initialize its corresponding memory bank. Goal images contribute reference features but no memory transitions. Goal-directed planning starts only after this initialization, and each newly observed transition is appended to the appropriate per-view bank. After an extended rollout, slots may be refreshed from accumulated observed frames. All retained transitions are then decoded again so that their memory masks remain aligned with the refreshed slots. No future frame from the evaluated rollout is used for initialization or refresh.

Downstream fine-tuning. The robot predictor is trained on seven-frame DROID (Khazatsky et al., 2024) clips sampled at 1 fps and resized to 384×384. Training uses eight GPUs, four clips per GPU, bfloat16 precision, seed 239, 100 epochs, and 300 iterations per epoch. The visual encoder and slot decomposition module remain frozen, while TMP and the task-specific state and action encoders are optimized with AdamW using $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, and weight decay 10^{-2}; bias and normalization parameters are excluded from weight decay. The learning rate increases linearly from 10^{-4} to 10^{-3} over the first 15 epochs, remains at 10^{-3} for 70 epochs, and decays linearly to zero over the final 15 epochs. We set $(\lambda_{\text{uot}}, \lambda_{\text{mask}}, \lambda_{\text{feat}}) = (3.0, 3.0, 0.1)$ in $\mathcal{L}_{\text{pred}}$, with five Sinkhorn iterations and $\varepsilon = 0.05$. Each training sample uses four observed frames to predict three future target frames and additionally applies two autoregressive prediction steps.

Multi-view prediction and planning. Let $v \in \mathcal{V} = \{\text{external-1}, \text{external-2}, \text{wrist}\}$ index the three views. They share the same predictor parameters θ, but each view independently maintains its slot set $\boldsymbol{S}^{(v)}$, mask history, and transition memory $\mathcal{B}_t^{(v)}$. Context is sampled within each memory, and prediction is performed separately as

$$\hat{\boldsymbol{M}}_{t+1}^{(v)} = F_\theta\left(\boldsymbol{S}^{(v)}, \boldsymbol{M}_t^{(v)}, r_t^{(v)}, a_t^{(v)}; \mathcal{C}_t^{(v)}\right). \tag{19}$$

The same candidate robot action is transformed into each camera coordinate system using its camera pose before prediction; there is no cross-view attention or joint memory bank inside TMP. CEM then proceeds in two stages. First, both fixed views evaluate every pose candidate, and the mean of their goal losses updates the CEM mean and variance. Second, the wrist-view search is initialized with this updated distribution and continues optimizing the pose using the wrist-view goal loss. The best candidate from this refinement is executed; the gripper search is likewise evaluated from the wrist view. Thus the fixed cameras provide a stable global search direction, while the moving wrist camera refines the action from its local view. Robot planning uses the standard UOT ground cost. Within each CEM stage, all candidates use the same sampled context transitions for the corresponding view.

Correlation distance. For the correlations in Tab. 4, EE–goal distance at step t is

$$d_{t+1}^{\text{EE}} = \|\boldsymbol{e}_{t+1} - \boldsymbol{e}_g\|_2, \tag{20}$$

where $\boldsymbol{e}_{t+1}$ is the realized next 7-DoF state and $\boldsymbol{e}_g$ is the active visual-goal stage's target state; each contains the 6-D EE pose and gripper opening. This evaluation metric is distinct from the 3-D positional threshold used only for baseline guidance.

Predicted loss is the selected candidate's goal loss before executing action a_t, while observed loss is recomputed from the observation immediately after executing that action. Thus, predicted loss, observed loss, and d_{t+1}^{EE} refer to the same transition. To prevent task scale and trajectory length from dominating the aggregate, we standardize each variable within every task, view, and method, divide the distance range into 12 equal-width bins, and evenly subsample overrepresented bins. We compute each statistic per task and report the macro average across tasks.

Tab. C5 reports all measured statistics for three comparisons: observed loss against physical distance, predicted loss against physical distance, and predicted loss against observed loss. Kendall's τ and Spearman's ρ measure monotonic agreement. The standardized least-squares slope β measures linear sensitivity, and R^2 measures linear goodness of fit. Slot-JEPA gives the strongest agreement for every comparison and camera view under all four statistics.

Table C5: Complete correlation analysis. Values are task-wise statistics macro-averaged after within-task standardization and distance-stratified sampling.

(a) Kendall's τ

Method	Obs. loss vs. EE–goal dist.			Pred. loss vs. EE–goal dist.			Pred. loss vs. Obs. loss		
	Ext. 1	Ext. 2	Wrist	Ext. 1	Ext. 2	Wrist	Ext. 1	Ext. 2	Wrist
V-JEPA 2.1	0.299	0.326	0.397	0.386	0.345	0.248	0.391	0.481	0.419
V-JEPA 2.1 + TMP	0.167	0.075	0.538	0.277	0.266	0.356	0.322	0.304	0.599
Slot-JEPA	**0.692**	**0.804**	**0.720**	**0.551**	**0.671**	**0.706**	**0.527**	**0.706**	**0.758**

(b) Spearman's ρ

Method	Obs. loss vs. EE–goal dist.			Pred. loss vs. EE–goal dist.			Pred. loss vs. Obs. loss		
	Ext. 1	Ext. 2	Wrist	Ext. 1	Ext. 2	Wrist	Ext. 1	Ext. 2	Wrist
V-JEPA 2.1	0.350	0.407	0.533	0.503	0.453	0.297	0.511	0.609	0.549
V-JEPA 2.1 + TMP	0.154	0.103	0.652	0.327	0.319	0.454	0.409	0.421	0.720
Slot-JEPA	**0.792**	**0.891**	**0.807**	**0.677**	**0.788**	**0.814**	**0.668**	**0.825**	**0.865**

(c) Standardized slope β

Method	Obs. loss vs. EE–goal dist.			Pred. loss vs. EE–goal dist.			Pred. loss vs. Obs. loss		
	Ext. 1	Ext. 2	Wrist	Ext. 1	Ext. 2	Wrist	Ext. 1	Ext. 2	Wrist
V-JEPA 2.1	0.423	0.391	0.594	0.549	0.519	0.391	0.601	0.610	0.504
V-JEPA 2.1 + TMP	0.144	0.106	0.699	0.388	0.360	0.544	0.411	0.475	0.745
Slot-JEPA	**0.865**	**0.940**	**0.874**	**0.706**	**0.848**	**0.882**	**0.730**	**0.887**	**0.922**

(d) Coefficient of determination R^2

Method	Obs. loss vs. EE–goal dist.			Pred. loss vs. EE–goal dist.			Pred. loss vs. Obs. loss		
	Ext. 1	Ext. 2	Wrist	Ext. 1	Ext. 2	Wrist	Ext. 1	Ext. 2	Wrist
V-JEPA 2.1	0.402	0.362	0.391	0.394	0.352	0.237	0.424	0.477	0.312
V-JEPA 2.1 + TMP	0.313	0.262	0.530	0.290	0.311	0.404	0.312	0.490	0.602
Slot-JEPA	**0.792**	**0.890**	**0.793**	**0.531**	**0.740**	**0.793**	**0.561**	**0.797**	**0.859**

Action execution. The 7-D planning command consists of a 6-D end-effector target pose and a scalar gripper-opening command. In absolute-pose mode, the arm controller computes Cartesian pose error, maps it through inverse kinematics to desired joint positions, and applies joint-space PD feedback with Jacobian-weighted Cartesian gains. A separate continuous gripper controller applies the opening command. We allow at most 100 MPC steps per planning attempt.

C.5 Additional Visualizations

UOT objective. Fig. 12 shows how the goal loss varies over sampled states. Low-loss regions follow feasible progress toward the goal while the obstacle-aware ground cost assigns less favorable energy to visually nearby but obstructed states, yielding a planning signal that better reflects scene geometry.

Predicted masks. Fig. 13 compares action-conditioned slot-mask predictions with the RGB observations obtained after execution. The predicted masks track the task-relevant regions across environments and time. Because the future is represented as separate slot masks, the prediction is also more interpretable: one can inspect which object or region is expected to move and where, rather than only comparing an opaque dense latent.

Slot decomposition. Figures 14–17 visualize individual slots as colored masks over the input RGB frames. Across Kinetics-400 pretraining videos and DROID downstream-training clips, different slots emphasize distinct objects or spatial regions and remain recognizable over time. This decomposition exposes which parts of a scene contribute to the latent state instead of collapsing them into one dense feature map.

Overall, these visualizations indicate that slot decomposition provides a spatially structured and inspectable state, while UOT and TMP turn that state into goal-directed costs and coherent future-mask predictions.

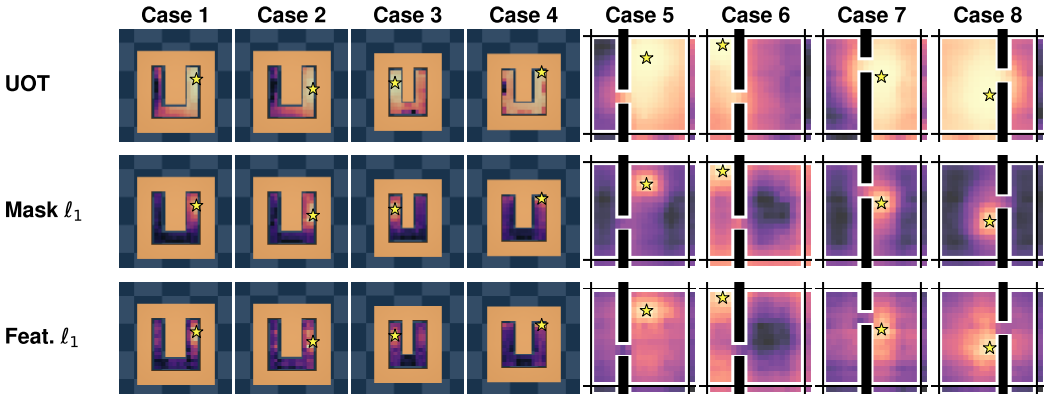

Figure 12: Additional goal-loss landscapes in MazeR and TwoRoomR. Each sampled state is rendered independently; the panels visualize energy geometry rather than executed rollouts. Stars mark goals, and lighter colors indicate lower loss.

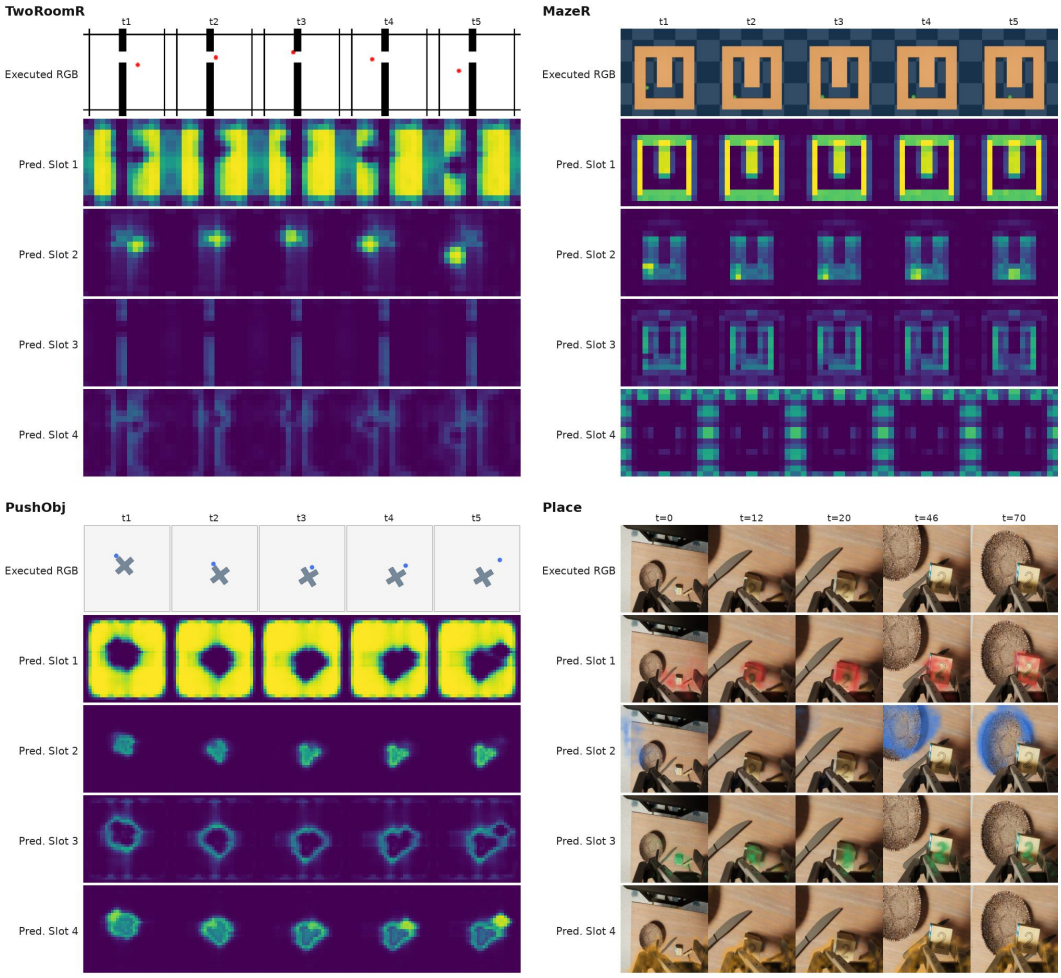

Figure 13: Predicted slot masks along successful TwoRoomR, MazeR, PushObj, and Place planning rollouts (top left, top right, bottom left, and bottom right, respectively). Each column pairs the RGB observation obtained after action execution with masks predicted for the same time step. The planar panels show mask heatmaps, whereas the Place panel overlays sharpened predictions on the executed wrist-view RGB.

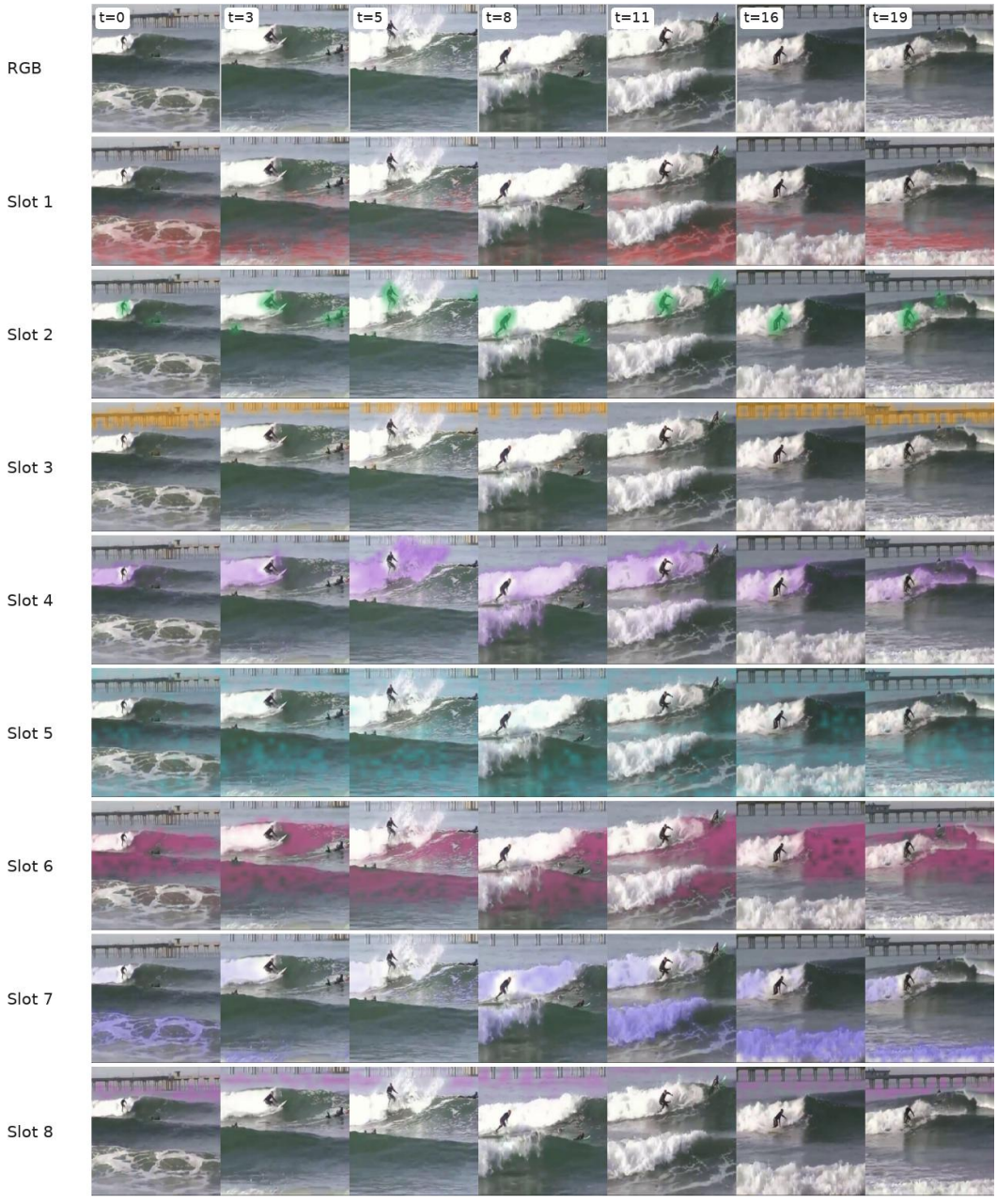

Figure 14: Slot decomposition on Kinetics-400 pretraining example. The first row contains the RGB frames, and each subsequent row overlays one selected soft slot mask.



Figure 15: Slot decomposition on Kinetics-400 pretraining example. The first row contains the RGB frames, and each subsequent row overlays one selected soft slot mask.

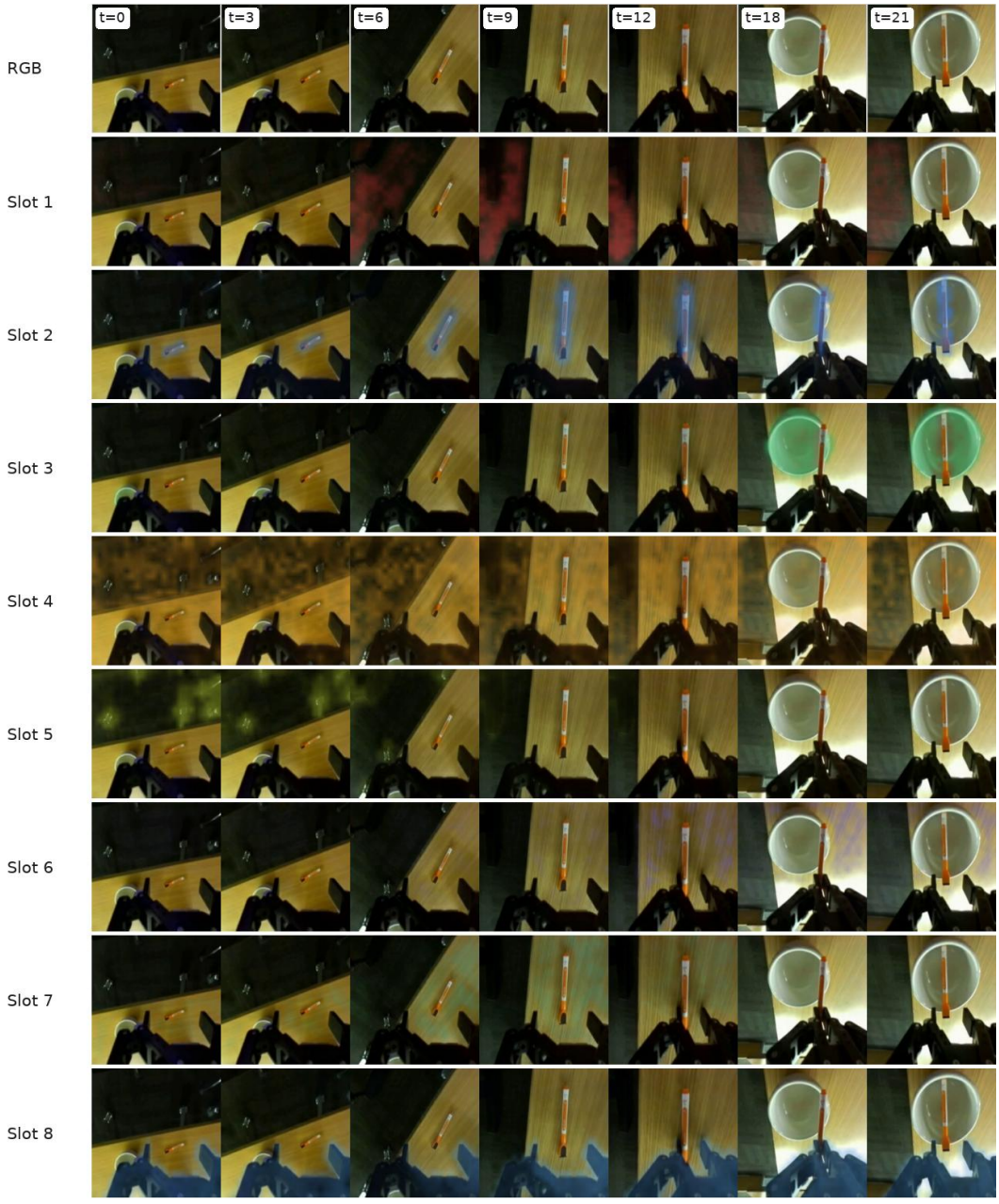

Figure 16: Slot decomposition on DROID downstream-training example. The first row contains the RGB frames, and each subsequent row overlays one selected soft slot mask.

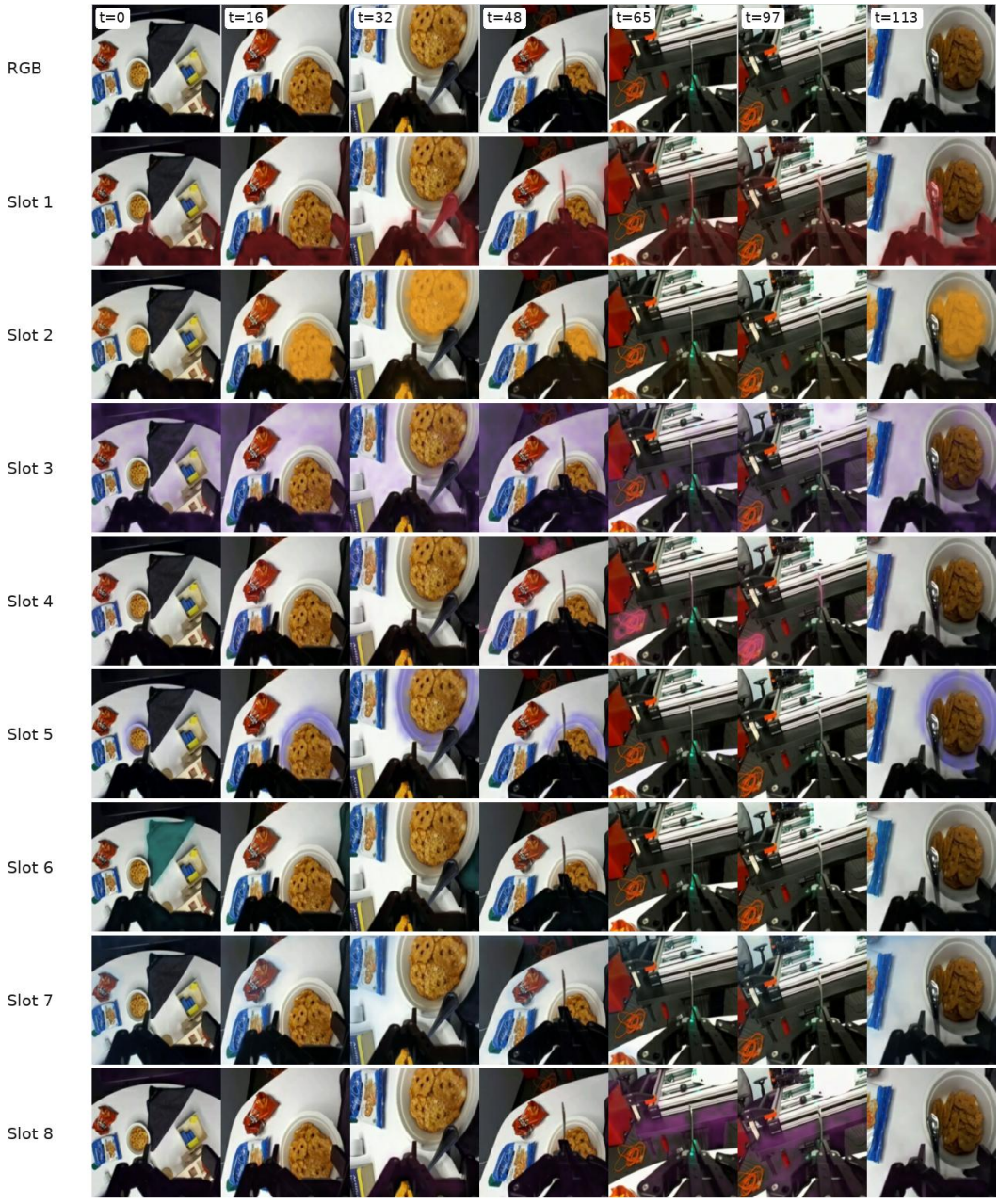

Figure 17: Slot decomposition on DROID downstream-training example. The first row contains the RGB frames, and each subsequent row overlays one selected soft slot mask.