FOCWM: FACTORIZED OBJECT-CENTRIC WORLD MODEL FOR LATENT-ACTION DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing object-centric world models typically decompose scenes into object slots, yet each slot remains a homogeneous representation in which geometry, shape, and appearance are entangled. Such entanglement requires inverse dynamics models to implicitly extract action-relevant cues from object representations, complicating latent-action inference. To address this challenge, we propose a Factorized Object-Centric World Model (FOCWM), which introduces a structured object representation together with adaptive latent-action inference for object-centric latent dynamics modeling. Specifically, we decompose each object slot into three complementary factors: spatial geometry, occupancy, and appearance. We further propose an action-oriented adaptive fusion mechanism that integrates these factors into object-wise representations for latent-action inference, allowing each factor to contribute adaptively according to the underlying dynamics. The inferred latent actions, together with the factorized object states, are then used to predict future object states. Experiments on synthetic and real-world datasets show that FOCWM consistently improves future prediction performance and downstream behavior learning compared with existing object-centric world models. The source code will be made publicly available.

1 INTRODUCTION

Action-conditioned world models aim to learn compact representations of an environment and predict how its state evolves in response to actions, providing a foundation for reasoning, planning, and decision-making in dynamic environments (Hafner et al., 2025; Gao et al., 2025). However, learning such dynamics remains challenging, as visual scenes contain multiple entities whose states, interactions, and visual properties require effective representation. Object-centric learning provides a natural abstraction for this problem by decomposing a scene into individual entities and representing them separately (Locatello et al., 2020; Kipf et al., 2021). By treating objects as the basic units of state, object-centric world models provide a structured state space for modeling object-wise dynamics and interactions, and have shown promising results in video prediction and action-conditioned dynamics modeling (Wu et al., 2022; Villar-Corrales & Behnke, 2025; Mosbach et al., 2025; Daniel et al., 2026).

Existing object-centric world models mainly build on two types of object representations: slot-based and particle-based representations. Slot-based approaches encode each object as a compact latent vector, facilitating object-level tracking and dynamics modeling (Locatello et al., 2020; Kipf et al., 2021; Elsayed et al., 2022). However, heterogeneous information within each object, such as spatial configuration, shape, and visual appearance, remains jointly encoded in a single latent representation. Particle-based approaches instead represent scenes using multiple localized latent particles, providing finer-grained spatial structure and local visual information (Daniel & Tamar, 2022; 2023; Daniel et al., 2026). Such representations often involve a larger number of latent entities, and multiple particles may jointly describe a single object, potentially increasing representation redundancy and the complexity of dynamics modeling. These limitations motivate a representation that preserves a compact object-level abstraction while introducing explicit structure within each object.

Learning action-conditioned dynamics further requires access to the actions that drive state transitions (Menapace et al., 2021; Bruce et al., 2024). In many real-world settings, however, action

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.

annotations are unavailable or costly to obtain, motivating latent-action world models that infer action-like variables directly from unlabeled videos by exploiting the visual transitions between consecutive observations (Villar-Corrales & Behnke, 2025; Gao et al., 2025). Existing approaches typically infer latent actions from holistic object representations (Villar-Corrales & Behnke, 2025; Daniel et al., 2026). While such representations are desirable for reconstructing and predicting complete visual states, they jointly encode heterogeneous information such as geometry, shape, and appearance, which plays different roles in describing object transitions. Consequently, the inverse dynamics model must implicitly disentangle and integrate these cues when inferring the underlying action. This motivates explicitly organizing within-object information before it is used for latent-action inference.

To address these challenges, we propose the **Factorized Object-Centric World Model (FOCWM)**, which integrates a structured object representation with adaptive latent-action inference. Instead of representing each object with a single homogeneous slot vector, FOCWM factorizes each slot into three feature subspaces: (1) a spatial feature encoding object geometry (position, scale, and depth), (2) an alpha feature modeling occupancy (object shape and visibility), and (3) an RGB feature capturing appearance (color, texture, and material). These subspaces serve as structured object-level state factors for action-conditioned dynamics. This design preserves a compact object-level abstraction while introducing explicit structure within each object.

Building on this structured representation, we introduce an adaptive feature fusion mechanism for latent-action inference. Rather than directly feeding the complete object representation into the inverse dynamics model, FOCWM dynamically integrates the spatial, occupancy, and appearance components to construct an action-oriented object representation. The fusion module learns to weight each subspace according to its utility for predicting the underlying latent action, thereby emphasizing action-relevant cues and down-weighting appearance factors that are less informative for the transition. The resulting representation is used to infer object-wise latent actions, while the complete factorized object state is retained for forward dynamics prediction. In this way, FOCWM can exploit complementary cues from different feature components for action inference.

We evaluate FOCWM on both synthetic and real-world video datasets. Our results demonstrate that the factorization of object representations, together with the adaptive integration of their constituent features, provides a more effective basis for latent dynamics modeling than directly operating on holistic object representations. More broadly, our findings suggest that object-centric world models can benefit not only from structuring scenes across objects, but also from introducing explicit subspaces within each object.

Our contributions are summarized as follows:

- We propose a novel factorized object representation that decomposes each object slot into functionally specialized geometric, occupancy, and appearance feature subspaces, providing a compact yet structured state representation for object-centric world modeling;
- We introduce an adaptive feature fusion mechanism that constructs an action-oriented object representation from complementary feature components for latent-action inference;
- We demonstrate the effectiveness of the proposed framework on synthetic and real-world datasets, showing improved future prediction performance and downstream behavior learning over existing object-centric world models.

2 Related Work

Object-centric learning represents visual scenes as sets of entities rather than holistic embeddings. Slot-based methods (Locatello et al., 2020; Kipf et al., 2021; Elsayed et al., 2022) provide compact object representations but typically encode heterogeneous information within a single slot. Particle-based approaches (Daniel & Tamar, 2022; 2023) capture finer-grained spatial structure using localized particles, often with more latent entities. FOCWM instead retains compact slot-level abstraction while factorizing each object into specialized feature subspaces. Object-centric world models learn visual dynamics in structured latent spaces (Wu et al., 2022; Song et al., 2025). Recent work has further explored controllable and task-relevant dynamics, including Dyn-O (Wang et al., 2026), OC-STORM (Zhang et al., 2026), PlaySlot (Villar-Corrales & Behnke, 2025), SOLD (Mosbach et al., 2025), FOCUS (Ferraro et al., 2025), and LPWM (Daniel et al., 2026). While existing latent-action

approaches typically infer actions from holistic object representations, FOCWM explicitly factorizes each object into spatial, occupancy, and appearance components and adaptively integrates them for latent-action inference. Additional discussion of related work is provided in Appendix A.1.

3 METHOD

We propose FOCWM, an object-centric world model for learning latent actions and visual dynamics from videos without action annotations. Given a video sequence $I_{1:T}$, we first extract temporally consistent object slots using Slot Attention (Locatello et al., 2020). To explicitly structure the information within each slot, we decompose each foreground slot into three complementary factors: geometric, occupancy, and appearance. These factors are instantiated as spatial, alpha, and RGB feature subspaces, respectively, yielding a structured object state. Furthermore, we propose an action-oriented adaptive fusion mechanism that integrates these feature subspaces into object-wise representations for latent-action inference, allowing each factor to contribute adaptively according to the underlying dynamics. The resulting representations are then fed into an inverse dynamics model to infer continuous latent actions for each state transition. Conditioned on the inferred actions and object-state history, an action-conditioned dynamics model predicts future factorized object states. The predicted spatial attributes, RGB patches, and alpha masks are then decoded and composited with the background to reconstruct future video frames. The model architecture is shown in Figure 1.

At inference time, FOCWM autoregressively predicts future object states and video frames from initial observations and latent actions, which can be inferred from observed transitions or provided externally by a policy operating in the latent-action space.

3.1 OBJECT-CENTRIC REPRESENTATION LEARNING

Object Discovery. To obtain compact object-level states from raw frames, we follow the slot encoding scheme of SAVi (Kipf et al., 2021) and extract object slots at each timestep. Specifically, we first encode each frame I_t with a convolutional network, flatten the spatial dimensions into a sequence of tokens, and apply a shared MLP to each token:

$$F_t = \mathrm{MLP}\left(\mathrm{Flatten}\left(\mathrm{CNN}(I_t)\right)\right) \in \mathbb{R}^{N \times d_f}, \tag{1}$$

where CNN denotes the convolutional encoder, N denotes the number of spatial tokens, and d_f is the feature dimension of each token. Slot Attention (Locatello et al., 2020) aggregates these tokens into a set of slots through iterative attention and recurrent updates:

$$\begin{aligned} S_t &= \mathrm{SlotAttention}(F_t, S_t^{\mathrm{init}}), \\ S_t^{\mathrm{init}} &= \mathrm{Trans}(S_{t-1}), \qquad t > 1. \end{aligned} \tag{2}$$

The first-frame initialization is sampled from a learned Gaussian distribution. For subsequent frames, a Transformer $Trans$ propagates the preceding slots to initialize the current updates, encouraging temporal consistency. Within Slot Attention, attention is normalized across slots to induce competition for image features. The aggregated features update each slot through a gated recurrent unit and a residual MLP.

The resulting scene representation is

$$S_t = (s_t^1, \ldots, s_t^K, s_t^{\mathrm{bg}}) \in \mathbb{R}^{(K+1) \times d_s}, \tag{3}$$

where K slots are assigned to foreground objects and the last slot s_t^{bg} is assigned to background rendering. Together, these slots represent the complete scene.

Factorized Object States. To make the internal structure of each object slot explicit and facilitate action-relevant reasoning, we factorize the object state into complementary feature groups. We map each foreground slot into spatial, alpha, and RGB feature groups using separate projection heads:

$$\begin{aligned} x_t^{i,g} &= f^g(s_t^i), \qquad g \in \mathcal{G}, \\ x_t^i &= [x_t^{i,\mathrm{spa}}; x_t^{i,\alpha}; x_t^{i,\mathrm{rgb}}], \end{aligned} \tag{4}$$

where $\mathcal{G} = \{\mathrm{spa}, \alpha, \mathrm{rgb}\}$ and $[\cdot;\cdot]$ denotes concatenation. Each projection head consists of a layer normalization and a two-layer MLP, and is shared across foreground slots and time. Each factorized

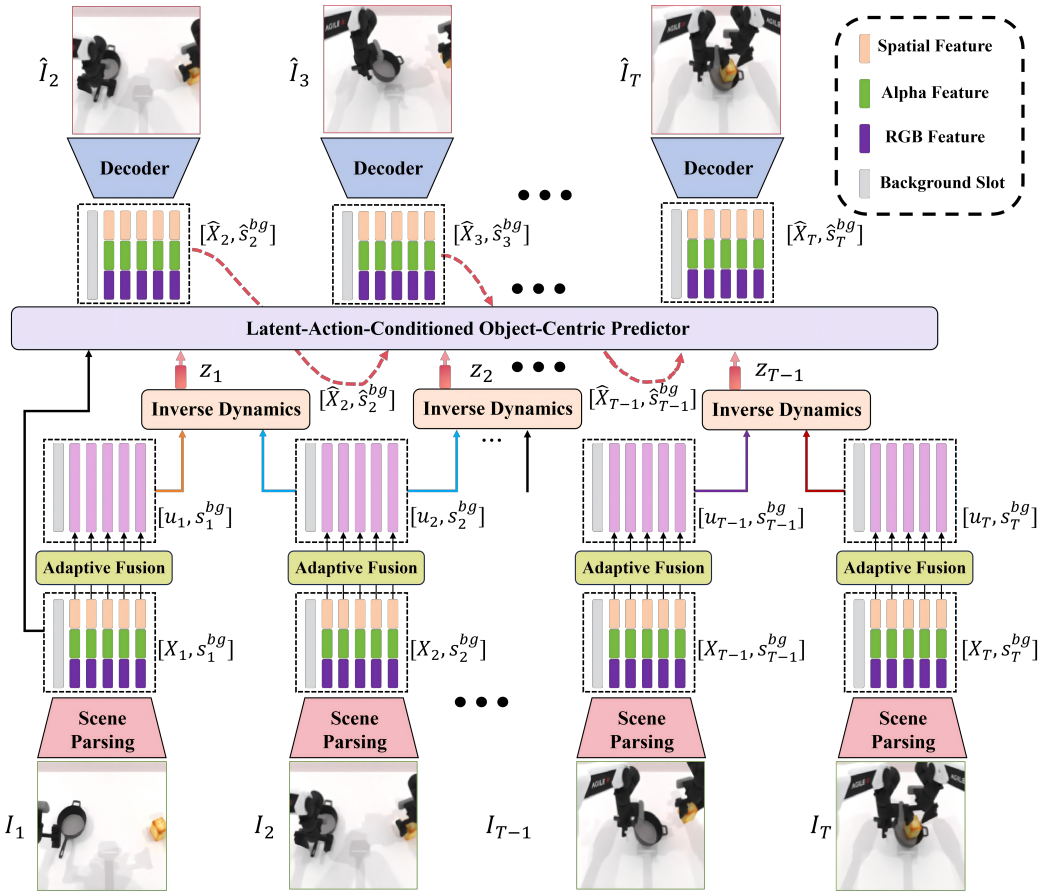


Figure 1: Model architecture. FOCWM is trained on unlabeled video sequences to learn factorized object-centric representations and infer latent actions, which together support autoregressive prediction of future object states and video frames.

object state has dimension $d_x = d_{\text{spa}} + d_\alpha + d_{\text{rgb}}$. We denote the collection of foreground object states by $X_t = (x_t^1, \ldots, x_t^K)$.

The three feature groups acquire distinct functional roles through their respective decoding paths. Spatial features parameterize object placement, alpha features generate local opacity masks, and RGB features generate local appearance patches. This factorization organizes the representation according to rendering functions without imposing statistical independence between feature groups.

Decoding. To ensure that the three feature subspaces acquire distinct and complementary functional roles, we decode each subspace through a dedicated path into an interpretable object factor. A spatial head predicts object position, scale, and depth from the spatial features:

$$(p_t^i, \sigma_t^i, d_t^i) = H^{\text{spa}}(x_t^{i,\text{spa}}), \tag{5}$$

where $p_t^i \in [-1, 1]^2$ denotes the normalized object position, $\sigma_t^i \in \mathbb{R}^2$ denotes the object scale, and $d_t^i \in \mathbb{R}$ denotes a learned depth-ordering variable used for object composition. The predicted depth does not necessarily correspond to physical 3D depth, as our model is trained solely on monocular RGB observations.

Meanwhile, we use separate decoders to map alpha and RGB features to an opacity patch and an appearance patch:

$$\begin{aligned} A_t^i &= \text{sigmoid}\big(H^\alpha(x_t^{i,\alpha})\big), \\ R_t^i &= H^{\text{rgb}}(x_t^{i,\text{rgb}}). \end{aligned} \tag{6}$$

For each object, the separately decoded RGB patch and alpha mask form a local RGBA patch $P_t^i = [R_t^i; A_t^i] \in \mathbb{R}^{L\times L\times 4}$, where L is the patch resolution. A differentiable spatial transformation (Jaderberg et al., 2015) scales and places each patch onto the full-resolution canvas according to its predicted position p_t^i and scale σ_t^i. The transformed RGB layers are blended using weights determined by their alpha masks and predicted depths, producing a foreground image $\hat{I}_t^{\mathrm{fg}}$. The background slot s_t^{bg} is decoded by an upsampling network into a background image $\hat{I}_t^{\mathrm{bg}}$. The complete frame is reconstructed as

$$\hat{I}_t = M_t \odot \hat{I}_t^{\mathrm{fg}} + (1 - M_t) \odot \hat{I}_t^{\mathrm{bg}}, \tag{7}$$

where M_t is the aggregate foreground opacity computed from the transformed alpha masks, and $\odot$ denotes elementwise multiplication.

3.2 Latent-Action Inference

We infer latent actions from observed state changes. Our inference model adaptively fuses the three foreground feature groups and separately embeds the background state. These representations are jointly contextualized across time and scene entities, after which latent actions are inferred from consecutive contextualized states and their differences.

Action-Oriented Adaptive Feature Fusion. For each foreground object and time step, we map the factorized features into a common embedding space using group-specific encoders:

$$e_t^{i,g} = E^g(x_t^{i,g}) \in \mathbb{R}^{d_e}, \qquad i = 1, \ldots, K, \tag{8}$$

where $x_t^{i,g}$ denotes the feature of group g for object i at time t, and E^g is the group-specific encoder. To construct an action-oriented object representation for latent-action inference, we introduce a gating network G that assigns a scalar weight to each group:

$$w_t^i = \mathrm{softmax}\left(G([e_t^{i,\mathrm{spa}}; e_t^{i,\alpha}; e_t^{i,\mathrm{rgb}}])\right), \tag{9}$$

where $\sum_{g\in\mathcal{G}} w_t^{i,g} = 1$. The embeddings are then combined through a weighted sum and passed through the fusion network M to obtain the object feature:

$$u_t^i = M\left(\sum_{g\in\mathcal{G}} w_t^{i,g} e_t^{i,g}\right). \tag{10}$$

The foreground fusion network is shared across objects and time. This adaptive weighting mechanism allows the model to dynamically emphasize informative subspaces while down-weighting less relevant ones for latent-action learning.

The background slot s_t^{bg} is mapped directly into the same embedding space through a separate encoder E^{bg}:

$$u_t^{\mathrm{bg}} = E^{\mathrm{bg}}(s_t^{\mathrm{bg}}) \in \mathbb{R}^{d_e}. \tag{11}$$

Since the background is represented by a single feature vector, it does not pass through the foreground feature-fusion module.

The gating weights are learned end-to-end through the inverse dynamics objective, so that the fusion emphasizes the factorized cues most useful for explaining the observed transition.

Inverse Dynamics. The fused object representations capture the factorized cues within each object, but inferring latent actions also requires reasoning about how these representations evolve over time and interact with one another. To model such temporal and inter-object dependencies, we add temporal positional encodings to the foreground tokens u_t^i and background token u_t^{bg}, and jointly process them with a causal spatio-temporal Transformer (Zhu et al., 2024). Each token can attend to all foreground and background tokens in the current and preceding frames, while attention to future frames is masked. The resulting representations h_t^i and h_t^{bg} incorporate temporal history and interactions between foreground objects and the background.

Let $\mathcal{J} = \{1, \ldots, K, \mathrm{bg}\}$ index all scene entities. For transition $t \to t+1$, we concatenate the contextualized representations of both endpoints and their difference:

$$v_t^j = P\left([h_t^j; h_{t+1}^j; h_{t+1}^j - h_t^j]\right), \qquad j \in \mathcal{J}, \tag{12}$$

where P is a shared projection MLP. This construction retains information about both states while explicitly exposing their change.

We denote the complete scene state by $\mathcal{X}_t = (X_t, s_t^{\mathrm{bg}})$. A shared distribution head maps each transition feature v_t^j to the mean μ_t^j and log variance ℓ_t^j of a Gaussian latent-action distribution:

$$\begin{aligned} (\mu_t^j, \ell_t^j) &= H(v_t^j), \\ q(z_t^j \mid \mathcal{X}_{1:t+1}) &= \mathcal{N}\left(z_t^j; \mu_t^j, \mathrm{diag}(\exp(\ell_t^j))\right), \qquad j \in \mathcal{J}. \end{aligned} \tag{13}$$

The latent actions are obtained via reparameterization:

$$z_t^j = \mu_t^j + \exp(\ell_t^j/2) \odot \epsilon_t^j, \qquad \epsilon_t^j \sim \mathcal{N}(0, \mathbf{I}), \quad j \in \mathcal{J}, \tag{14}$$

3.3 Action-Conditioned Object Dynamics

Our dynamics model predicts future scene states directly in the factorized object space, rather than in holistic slot or pixel space. Each foreground object and the background are conditioned on their own inferred latent actions, allowing the model to capture object-specific effects of actions while preserving the structured object representation throughout rollouts.

Let $\widetilde{x}_t^i$ and $\widetilde{s}_t^{\mathrm{bg}}$ denote the observed or previously predicted foreground and background states supplied at time t. We project each state and its corresponding latent action into a common embedding space, combine them by addition, and add a temporal positional encoding:

$$y_t^i = A(\widetilde{x}_t^i) + B(z_t^i) + \mathrm{PE}(t), \qquad y_t^{\mathrm{bg}} = A^{\mathrm{bg}}(\widetilde{s}_t^{\mathrm{bg}}) + B^{\mathrm{bg}}(z_t^{\mathrm{bg}}) + \mathrm{PE}(t). \tag{15}$$

The foreground and background use separate learned linear projections, producing the scene tokens $Y_t = (y_t^1, \ldots, y_t^K, y_t^{\mathrm{bg}})$. A causal spatio-temporal Transformer processes the action-conditioned tokens across time:

$$(g_t^1, \ldots, g_t^K, g_t^{\mathrm{bg}}) = \left[\mathrm{Tr}_\theta(Y_{1:t})\right]_t. \tag{16}$$

Separate regression heads predict residual updates for foreground features and the background slot:

$$\widehat{x}_{t+1}^i = \widetilde{x}_t^i + R(g_t^i), \qquad \widehat{s}_{t+1}^{\mathrm{bg}} = \widetilde{s}_t^{\mathrm{bg}} + R^{\mathrm{bg}}(g_t^{\mathrm{bg}}). \tag{17}$$

Here, R and R^{bg} are foreground and background regression heads that map the Transformer outputs to residual updates in their respective state spaces.

Finally, we decode the predicted foreground states and background slot using the pretrained decoder to obtain the next video frame:

$$\widehat{I}_{t+1} = D_\phi\left(\widehat{X}_{t+1}, \widehat{s}_{t+1}^{\mathrm{bg}}\right). \tag{18}$$

The decoder converts the predicted spatial, alpha, and RGB features into object layers and combines them with the decoded background.

3.4 Training

Representation Learning. We first train the object-centric representation learning model. Let $\hat{I}_t$ denote the reconstruction of frame I_t. The representation model is optimized using a combination of pixel-level reconstruction and perceptual supervision:

$$\mathcal{L}_{\mathrm{repr}} = \mathcal{L}_{\mathrm{img}}^{\mathrm{repr}} + \lambda_{\mathrm{perc}}^{\mathrm{repr}} \mathcal{L}_{\mathrm{perc}}^{\mathrm{repr}}, \tag{19}$$

where $\mathcal{L}_{\mathrm{img}}^{\mathrm{repr}}$ encourages accurate pixel-level reconstruction, while $\mathcal{L}_{\mathrm{perc}}^{\mathrm{repr}}$ promotes perceptual consistency between reconstructed and observed frames.

Latent-Action and Dynamics Learning. We then freeze the pretrained representation model and jointly train the latent-action inference model and the dynamics model. Given latent actions inferred from the observed sequence, the dynamics model predicts the next foreground states and background representation, which are subsequently decoded into future frames. The training objective is

$$\mathcal{L}_{\mathrm{dyn}} = \lambda_{\mathrm{img}}\mathcal{L}_{\mathrm{img}} + \lambda_{\mathrm{feat}}\mathcal{L}_{\mathrm{feat}} + \lambda_{\mathrm{bg}}\mathcal{L}_{\mathrm{bg}} + \lambda^{\mathrm{dyn}}_{\mathrm{perc}}\mathcal{L}_{\mathrm{perc}} + \lambda_{\mathrm{KL}}\mathcal{L}_{\mathrm{KL}}, \tag{20}$$

where $\mathcal{L}_{\mathrm{img}}$ supervises future-frame prediction, $\mathcal{L}_{\mathrm{feat}}$ and $\mathcal{L}_{\mathrm{bg}}$ constrain the predicted foreground and background representations, respectively, $\mathcal{L}_{\mathrm{perc}}$ encourages perceptual consistency, and $\mathcal{L}_{\mathrm{KL}}$ regularizes the inferred latent-action space. Foreground and background supervision targets are obtained from the frozen representation model.

Full loss details are provided in Appendix A.3.3.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate our method on four robotic manipulation datasets spanning simulated and real-world settings. *ButtonPress* (Mosbach et al., 2025; Yu et al., 2020) is a MetaWorld-based Sawyer manipulation task (Yu et al., 2020), while *BlockPush* (Li et al., 2020) involves multi-object pushing and object interactions. We further consider *PlaceBreadSkillet* from RoboTwin (Chen et al., 2025), a dual-arm manipulation setting requiring bread placement into a skillet, and the real-world *Sketchy* dataset (Cabi et al., 2019), which contains interactions with objects of diverse shapes and appearances. Input resolution is 64×64 for ButtonPress and BlockPush, and 128×128 for PlaceBreadSkillet and Sketchy. Additional dataset details are provided in the Appendix A.2.

Implementation Details. FOCWM is implemented in PyTorch and trained on a single NVIDIA A100 GPU using Adam optimizer (Kingma & Ba, 2014). We first train the object-centric representation model and then freeze it during latent-action and dynamics learning. Additional architectural and optimization details are provided in Appendix A.3.

Metrics. For latent-action-conditioned video prediction, we report Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM) (Wang et al., 2004), and Learned Perceptual Image Patch Similarity (LPIPS) (Zhang et al., 2018) to compare predicted frames with ground-truth observations. Higher PSNR and SSIM and lower LPIPS indicate better prediction quality.

4.2 Latent-Action-Conditioned Video Prediction

Following the evaluation protocol of PlaySlot (Villar-Corrales & Behnke, 2025) and LPWM (Daniel et al., 2026), we evaluate FOCWM under latent-action-conditioned prediction, where latent actions inferred from ground-truth transitions are used to condition future rollouts. As shown in Table 1, FOCWM consistently outperforms PlaySlot and achieves the best overall performance on most metrics compared with LPWM. These results demonstrate that the proposed factorized object representation provides an effective and robust state space for long-horizon visual dynamics modeling.

Table 1: Quantitative comparison across different environments. c denotes the number of conditioning frames, and p denotes the number of predicted frames.

	BlockPush (64×64) c:6, p:15			ButtonPress (64×64) c:6, p:15			PlaceBreadSkillet (128×128) c:6, p:30			Sketchy (128×128) c:6, p:44		
Model	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
PlaySlot	26.69	0.94	0.037	32.03	0.96	0.024	16.82	0.65	0.372	22.63	0.80	0.275
LPWM	29.03	0.97	0.021	32.52	0.96	0.011	22.87	0.87	0.115	28.41	0.91	0.070
Ours	**30.34**	**0.98**	**0.016**	**34.22**	**0.98**	**0.007**	**23.93**	**0.90**	**0.112**	**29.45**	**0.92**	**0.061**

Figure 2 presents qualitative comparisons of video prediction across different methods on a real-world dataset. PlaySlot exhibits lower reconstruction quality, with predicted frames becoming increasingly blurry and losing fine-grained object details. In contrast, both LPWM and our method produce substantially sharper predictions and better preserve object structures. While LPWM and our method achieve comparable visual quality at short prediction horizons, their behaviors diverge as

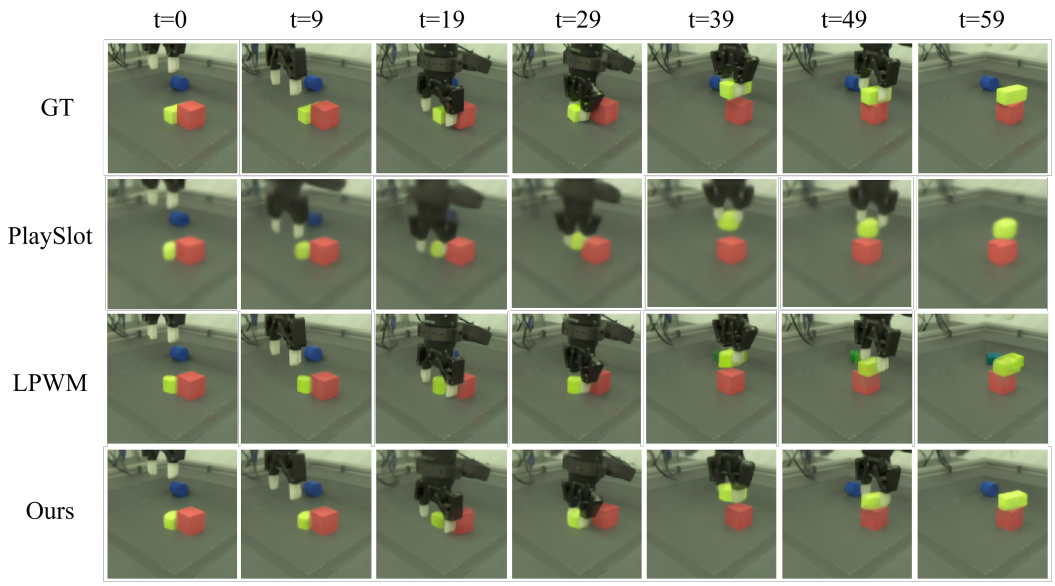


Figure 2: Qualitative results of different methods on the Sketchy dataset. FOCWM better preserves object appearance and visual consistency compared with PlaySlot and LPWM.

Table 2: Success rates on ButtonPress and BlockPush. Results are reported as mean ± standard deviation.

Model	ButtonPress (%) ↑	BlockPush (%) ↑
PlaySlot	63.4 ± 1.6	52.8 ± 2.8
LPWM	77.6 ± 1.9	63.2 ± 2.1
Ours	$\mathbf{80.3 \pm 2.1}$	$\mathbf{65.8 \pm 1.4}$

the rollout length increases. LPWM tends to suffer from progressive appearance drift, where object colors and shapes gradually deviate from those in the input observations. Our method, by comparison, maintains more consistent object appearance and geometry over long-horizon prediction, resulting in visually more stable rollouts. More results are provided in Appendix A.6.

4.3 Learning Behaviors from Expert Demonstrations

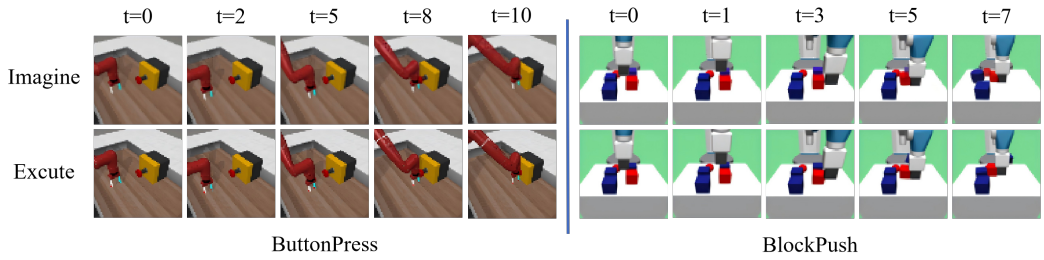


Figure 3: Imagined and executed trajectories on ButtonPress and BlockPush.

Following PlaySlot (Villar-Corrales & Behnke, 2025), we train a policy model (see Appendix A.4) on the ButtonPress and BlockPush tasks to evaluate the quality of the learned latent actions. Given a limited set of expert demonstrations, the policy operates on the frozen FOCWM representations to predict latent actions, which are subsequently mapped to executable robot actions by an action decoder. We compare the resulting policies with baseline methods under the same demonstration settings. Table 2 reports the success rates on the ButtonPress and BlockPush tasks, with each method evaluated over 500 trials per task across three random seeds. FOCWM consistently outperforms the

baseline methods on both tasks, demonstrating the effectiveness of the learned object-centric representations and latent-action space for downstream behavior learning and control. Figure 3 shows qualitative comparisons between imagined and executed trajectories on the two tasks. Imagined frames are predicted using latent actions generated by the learned latent policy, while executed frames are obtained by applying the corresponding decoded actions in the environment. The imagined trajectories closely align with the executed ones on both ButtonPress and BlockPush, indicating that the learned latent actions capture task-relevant dynamics and can be reliably decoded into executable behaviors.

4.4 Model Analysis

Table 3: Ablation study of different fusion strategies.

	Sketchy			ButtonPress
Model	PSNR ↑	SSIM ↑	LPIPS ↓	Success Rate (%) ↑
Holistic slot	28.98	0.90	0.065	75.5 ± 1.7
Spatial	27.89	0.83	0.078	70.4 ± 2.6
[Spatial, Alpha]	28.66	0.91	0.068	74.8 ± 2.0
[Spatial, Alpha, RGB]	29.23	**0.92**	0.063	79.2 ± 1.9
Adaptive fusion	**29.45**	**0.92**	**0.061**	$\mathbf{80.3 \pm 2.1}$

Ablation on Feature Fusion. Table 3 evaluates feature factorization and fusion strategies on Sketchy and ButtonPress. The Holistic-slot baseline applies inverse dynamics directly to the unfactorized object slot, with all other settings unchanged. Performance improves as alpha and RGB features are added to spatial features, indicating their complementary roles. Directly combining all factorized features already surpasses the holistic slot baseline, while adaptive fusion achieves the best overall performance, validating both within-object factorization and adaptive weighting. Additional results are provided in Appendix A.5.

Inference Time and Memory. Table 4 compares inference time and memory usage on BlockPush and Sketchy (implementation details in Appendix A.3.4). FOCWM is more efficient than both PlaySlot and LPWM. Compared with LPWM, it maintains a compact object-level representation instead of modeling many latent particles, reducing temporal and relational modeling costs. Compared with PlaySlot, it reconstructs objects from localized patches rather than full-image decoding, further lowering computation and memory overhead.

Table 4: Comparison of inference time and GPU memory usage.

	BlockPush		Sketchy	
Model	Time (s) ↓	GPU Usage (MiB) ↓	Time (min) ↓	GPU Usage (MiB) ↓
PlaySlot	87	39396	20.5	73064
LPWM	371	7638	31.1	36798
Ours	**28**	**2864**	**6.2**	**3134**

5 Conclusion

We presented FOCWM, a factorized object-centric world model for learning latent actions and visual dynamics from videos without action annotations. FOCWM explicitly organizes each object representation into spatial, alpha, and RGB feature subspaces, and adaptively integrates these components to construct representations for latent-action inference. This design preserves a compact object-level state while exposing meaningful within-object structure for dynamics modeling. Experiments across simulated and real-world robotic environments demonstrate that FOCWM consistently improves video prediction over existing object-centric world models. The learned representations and latent-action space also transfer effectively to downstream behavior learning, while maintaining substantially lower inference time and memory consumption. Overall, our results highlight the importance of modeling not only the structure between objects, but also the heterogeneous structure within each object when learning object-centric world models.

6 AI USE STATEMENT

Generative AI tools were used solely for language polishing and improving the clarity and readability of the manuscript. They were not used to generate research ideas, design experiments, or produce scientific conclusions. The authors reviewed and revised all AI-assisted text and take full responsibility for the content of the paper.

REFERENCES

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, et al. Genie: Generative interactive environments. In *Proceedings of the 41st International Conference on Machine Learning*, Proceedings of Machine Learning Research, pp. 4603–4623. PMLR, 2024.

Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad Zolna, Yusuf Aytar, David Budden, Mel Vecerik, et al. Scaling data-driven robotics with reward sketching and batch reinforcement learning. *arXiv preprint arXiv:1909.12200*, 2019.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

Tal Daniel and Aviv Tamar. Unsupervised image representation learning with deep latent particles. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 4644–4665. PMLR, 2022.

Tal Daniel and Aviv Tamar. Ddlp: Unsupervised object-centric video prediction with deep dynamic latent particles. *arXiv preprint arXiv:2306.05957*, 2023.

Tal Daniel, Carl Qi, Dan Haramati, Amir Zadeh, Chuan Li, Aviv Tamar, Deepak Pathak, and David Held. Latent particle world models: Self-supervised object-centric stochastic dynamics modeling. *arXiv preprint arXiv:2603.04553*, 2026.

Gamaleldin 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. *Advances in Neural Information Processing Systems*, 35:28940–28954, 2022.

Stefano Ferraro, Pietro Mazzaglia, Tim Verbelen, and Bart Dhoedt. Focus: object-centric world models for robotic manipulation. *Frontiers in Neurorobotics*, 19:1585386, 2025.

Shenyuan Gao, Siyuan Zhou, Yilun Du, Jun Zhang, and Chuang Gan. Adaworld: Learning adaptable world models with latent actions. In *Proceedings of the 42nd International Conference on Machine Learning*, Proceedings of Machine Learning Research, pp. 18744–18771. PMLR, 2025.

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

Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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. *arXiv preprint arXiv:2111.12594*, 2021.

Richard Li, Allan Jabri, Trevor Darrell, and Pulkit Agrawal. Towards practical multi-object manipulation using relational reinforcement learning. In *2020 ieee international conference on robotics and automation (icra)*, pp. 4051–4058. IEEE, 2020.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems*, volume 33, pp. 11525–11538, 2020.

Willi Menapace, Stephane Lathuiliere, Sergey Tulyakov, Aliaksandr Siarohin, and Elisa Ricci. Playable video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10061–10070, June 2021.

Malte Mosbach, Jan Niklas Ewertz, Angel Villar-Corrales, and Sven Behnke. Sold: Slot object-centric latent dynamics models for relational manipulation learning from pixels. In *Proceedings of the 42nd International Conference on Machine Learning*, Proceedings of Machine Learning Research, pp. 44911–44935. PMLR, 2025.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Yeon-Ji Song, Jaein Kim, Suhyung Choi, Jin-Hwa Kim, and Byoung-Tak Zhang. Ock: Unsupervised dynamic video prediction with object-centric kinematics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11359–11368, October 2025.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Angel Villar-Corrales and Sven Behnke. Playslot: Learning inverse latent dynamics for controllable object-centric video prediction and planning. In *Proceedings of the 42nd International Conference on Machine Learning*, Proceedings of Machine Learning Research, pp. 61534–61563. PMLR, 2025.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Zizhao Wang, Kaixin Wang, Li Zhao, Peter Stone, and Jiang Bian. Dyn-o: Building structured world models with object-centric representations. *Advances in Neural Information Processing Systems*, 38:121788–121812, 2026.

Ziyi Wu, Nikita Dvornik, Klaus Greff, Thomas Kipf, and Animesh Garg. Slotformer: Unsupervised visual dynamics simulation with object-centric models. *arXiv preprint arXiv:2210.05861*, 2022.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 586–595. IEEE, 2018.

Weipu Zhang, Adam Jelley, Trevor McInroe, Amos Storkey, and Gang Wang. Object-centric world models from few-shot annotations for sample-efficient reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 91366–91394, 2026.

Fangqi Zhu, Hongtao Wu, Song Guo, Yuxiao Liu, Chilam Cheang, and Tao Kong. Irasim: Learning interactive real-robot action simulators. *arXiv preprint arXiv:2406.14540*, 2(5):6, 2024.

A Appendix

A.1 Related Work

A.1.1 Object-Centric Representation Learning

Object-centric learning aims to decompose complex visual scenes into a set of object-level representations, providing a structured alternative to holistic scene embeddings. A representative line of work is based on slot representations. Slot Attention (Locatello et al., 2020) introduces an iterative attention mechanism that binds visual features to a fixed set of latent slots, with each slot typically corresponding to an individual object. Its temporal extensions, such as SAVi (Kipf et al., 2021) and SAVi++ (Elsayed et al., 2022), propagate object slots across video frames to maintain temporal consistency and enable object discovery and tracking. Slot-based approaches offer a compact object abstraction, yet heterogeneous information remains jointly encoded within each slot, leaving its internal structure implicit.

An alternative line of research represents scenes using latent particles. DLP (Daniel & Tamar, 2022) models an image as a collection of particles equipped with explicit spatial locations and local visual features, while DDLP (Daniel & Tamar, 2023) extends this representation to videos by modeling the temporal evolution of dynamic particles. More recently, LPWM (Daniel et al., 2026) builds a world model on top of latent particles to capture stochastic dynamics and interactions. Compared with slot-based representations, particle-based models provide finer-grained spatial structure and localized visual information. However, such representations operate at a finer granularity and may require multiple particles to describe different parts of an object, increasing the number of latent entities and potentially the complexity of relational and temporal dynamics modeling.

Our method preserves the compact object abstraction of slots while factorizing each object into specialized subspaces, providing a structured representation for dynamics and latent-action modeling.

A.1.2 Object-Centric World Models

Object-centric representations provide a natural state space for modeling visual dynamics. SlotFormer (Wu et al., 2022) models temporal dependencies and object interactions directly in the slot space using a Transformer, enabling future object-state prediction. OCK (Song et al., 2025) further incorporates object-centric kinematic structure for dynamic video prediction. In parallel, particle-based approaches such as DDLP (Daniel & Tamar, 2023) model temporal evolution over localized latent particles. These methods demonstrate the advantage of representing scene dynamics at the object level rather than directly in pixel or global latent spaces.

Another line of work focuses on learning controllable dynamics from videos without explicit action annotations. Dyn-O (Wang et al., 2026) builds a structured world model over object-centric representations and further decouples object features into dynamics-aware and dynamics-agnostic components. OC-STORM (Zhang et al., 2026) integrates object-centric representations into model-based reinforcement learning using few-shot annotations, improving the modeling of task-relevant object dynamics and interactions in visually complex environments. PlaySlot (Villar-Corrales & Behnke, 2025) learns inverse latent dynamics over object-centric representations for controllable video prediction and planning, while SOLD (Mosbach et al., 2025) and FOCUS (Ferraro et al., 2025) further explore object-centric dynamics and world models for robotic manipulation. LPWM (Daniel et al., 2026) builds a stochastic object-centric world model over localized latent particles, modeling their temporal evolution and interactions for future-state prediction.

Despite these advances, existing latent-action world models often infer actions from holistic object representations, leaving the roles of geometry, shape, and appearance implicit, whereas our method makes these functional components explicit and adaptively integrates them for latent-action inference.

A.2 Dataset Details

BlockPush. The BlockPush environment, simulated in MuJoCo (Todorov et al., 2012), consists of a robot arm interacting with multiple uniformly colored blocks on a tabletop. We use two dataset variants with different interaction patterns and purposes. The first variant is generated by a random

exploration policy, where the robot moves freely in the environment with limited task-specific object interactions. This dataset contains 20,000 training sequences and 2,000 validation sequences. We use it to train FOCWM. The second variant consists of expert demonstrations for a goal-directed pushing task, where the robot is required to move a distinctly colored block toward a target location indicated by a red sphere. This setting requires the model to capture object relations and collision dynamics. We use 5,000 expert demonstrations using a pretrained policy (Mosbach et al., 2025), which achieves an approximately 80% success rate on the task. The dataset is split into 4,500 training sequences for training the policy model and action decoder, and 500 evaluation sequences for benchmarking prediction performance.

ButtonPress. The ButtonPress environment is based on MetaWorld (Yu et al., 2020) and features a Sawyer robot arm tasked with pressing a red button. Compared with BlockPush, this environment involves more complex object geometries and visual textures, providing a complementary setting for evaluating object-centric dynamics modeling. We similarly use two dataset variants with different interaction patterns and purposes. The first variant is generated using a random exploration policy and contains 10,000 sequences, including 9,000 training sequences and 1,000 validation sequences. We use this dataset to train FOCWM. The second variant consists of expert demonstrations in which the robot successfully presses the target button. We use 1,000 demonstrations obtained from an expert policy, of which 900 are used for training the policy model and action decoder, while the remaining 100 are reserved for benchmarking prediction performance.

PlaceBreadSkillet. The PlaceBreadSkillet environment is based on RoboTwin 2.0 (Chen et al., 2025) and features an Aloha-AgileX robot tasked with picking up a piece of bread from a table and placing it into a skillet using one arm. Compared with the previous environments, this task involves more complex object geometries and multi-stage interactions, including grasping, lifting, and placement. We collect 500 demonstration sequences, consisting of 400 training sequences, 50 validation sequences, and 50 test sequences. Visual observations are captured from the head-mounted camera and resized to 128×128 pixels, with every third frame retained to reduce temporal redundancy. We use the training split to train FOCWM, while the validation and test splits are used for model validation and final evaluation, respectively.

Sketchy. Sketchy is a real-world robotic dataset introduced by (Cabi et al., 2019), featuring a robotic gripper interacting with diverse objects under stochastic dynamics and relatively sparse interactions. We focus on the `stack_green_on_red` task, which contains 198 expert trajectories and 3,241 rollout trajectories. We split the dataset into 80% training, 10% validation, and 10% test sets. Each trajectory is truncated to the first 70 frames and resized to 128×128.

A.3 Implementation Details

A.3.1 Model Architecture

Our FOCWM model consists of an object-centric learning model, a latent action inference module, and an action-conditioned dynamics module.

Object-centric learning model. Given an input video, an encoder combines convolutional features with recurrent Slot Attention to extract seven foreground slots and one background slot, each of dimension 256. Three independent MLPs map each foreground slot into spatial, alpha, and appearance features of dimensions 64, 64, and 128, respectively. Their concatenation forms the object state $x_t^i \in \mathbb{R}^{256}$ for object i at time t.

The alpha and RGB decoders map each predicted object's alpha and appearance features to a local alpha mask and RGB patch, respectively. The local patch size is set to $L = 32$ for ButtonPress and BlockPush, and $L = 64$ for PlaceBreadSkillet and Sketchy. A separate head maps its spatial features to position, two-dimensional scale, and depth. Differentiable spatial transformations (Jaderberg et al., 2015) place the local patches onto the image canvas, where alpha- and depth-dependent blending combines overlapping objects. The resulting foreground is composited with an independently decoded background at the full image resolution to reconstruct the predicted frame.

Latent action inference. We project the three feature groups into a shared 128-dimensional space and fuse them using adaptive, per-object gating. A four-layer temporal causal Transformer with four

attention heads processes the fused tokens and background tokens to capture temporal dependencies and interactions between objects and background. The dimensionality of the latent action is set to 7.

Action-conditioned dynamics. The dynamics model is a four-layer Transformer with four attention heads and a hidden dimension of 256. Foreground tokens are constructed by summing linear projections of the object states and their latent actions. The background slot is embedded separately, and learned type embeddings distinguish foreground and background tokens. Temporal causal self-attention jointly models their evolution, while separate output heads predict residual updates. Predicted states are fed back into the model for autoregressive rollout, conditioned on a sequence of latent actions.

A.3.2 Training Details

Representation learning. This stage is trained for object-centric decomposition using the Adam optimizer (Kingma & Ba, 2014), with a batch size of 64, an input sequence length of eight frames, and a base learning rate of 1×10^{-4}. The learning rate is linearly warmed up over the first 4,000 training steps and then decayed using a cosine annealing schedule for the remainder of training. We additionally apply gradient clipping with a maximum norm of 0.05.

Latent-action and dynamics learning. We jointly train the latent-action and dynamics modules using a pretrained object-centric decomposition model. Both modules are optimized with the Adam optimizer (Kingma & Ba, 2014), using a batch size of 16 and a base learning rate of 1×10^{-4}. The learning rate is decayed throughout training using a cosine annealing schedule. To improve training stability, we apply gradient clipping with a maximum norm of 0.05.

A.3.3 Loss Function

Representation learning. For completeness, we provide the detailed definitions of the loss terms used for representation learning. Let $\hat{I}_t = D_\phi(X_t, s_t^{\mathrm{bg}})$ denote the reconstruction of frame I_t. The image reconstruction loss is defined as

$$\mathcal{L}_{\mathrm{img}}^{\mathrm{repr}} = \frac{1}{T}\sum_{t=1}^{T} \mathrm{MSE}(\hat{I}_t, I_t). \tag{21}$$

The perceptual reconstruction loss is

$$\mathcal{L}_{\mathrm{perc}}^{\mathrm{repr}} = \frac{1}{T}\sum_{t=1}^{T}\sum_{l\in\mathcal{S}} \mathrm{mean}\left|\Phi_l(\hat{I}_t) - \Phi_l(I_t)\right|, \tag{22}$$

where $\Phi_l(\cdot)$ denotes the feature map extracted from layer l of a frozen ImageNet-pretrained VGG16 network (Simonyan & Zisserman, 2014), and $\mathcal{S}$ denotes the set of selected layers.

The complete representation objective is therefore

$$\mathcal{L}_{\mathrm{repr}} = \mathcal{L}_{\mathrm{img}}^{\mathrm{repr}} + \lambda_{\mathrm{perc}}^{\mathrm{repr}}\mathcal{L}_{\mathrm{perc}}^{\mathrm{repr}}, \tag{23}$$

where $\lambda_{\mathrm{perc}}^{\mathrm{repr}} = 0.01$.

Latent-action and dynamics learning. We next provide the detailed definitions of the individual terms in the dynamics objective. The image prediction loss is

$$\mathcal{L}_{\mathrm{img}} = \frac{1}{T-1}\sum_{t=1}^{T-1} \mathrm{MSE}(\widehat{I}_{t+1}, I_{t+1}). \tag{24}$$

The foreground feature prediction loss is defined as

$$\mathcal{L}_{\mathrm{feat}} = \frac{1}{T-1}\sum_{t=1}^{T-1}\sum_{g\in\mathcal{G}} \mathrm{MSE}(\widehat{X}_{t+1}^{g}, X_{t+1}^{g}), \tag{25}$$

where $X_t^g = (x_t^{1,g}, \ldots, x_t^{K,g})$ collects feature group g across the K foreground objects, and $\mathcal{G}$ denotes the set of feature groups.

The background prediction loss is

$$\mathcal{L}_{\text{bg}} = \frac{1}{T-1} \sum_{t=1}^{T-1} \text{MSE}(\widehat{s}_{t+1}^{\text{bg}}, s_{t+1}^{\text{bg}}). \tag{26}$$

The perceptual prediction loss is given by

$$\mathcal{L}_{\text{perc}} = \frac{1}{T-1} \sum_{t=1}^{T-1} \sum_{l \in \mathcal{S}} \text{mean} \left| \Phi_l(\widehat{I}_{t+1}) - \Phi_l(I_{t+1}) \right|. \tag{27}$$

Finally, the latent-action regularization term is

$$\mathcal{L}_{\text{KL}} = \frac{1}{T-1} \sum_{t=1}^{T-1} \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} D_{\text{KL}} \left(q_\psi(z_t^j \mid \mathcal{X}_{1:t+1}) \,\|\, \mathcal{N}(0, I) \right), \tag{28}$$

where $\mathcal{J}$ indexes the foreground and background latent-action variables.

The complete dynamics objective is

$$\mathcal{L}_{\text{dyn}} = \lambda_{\text{img}} \mathcal{L}_{\text{img}} + \lambda_{\text{feat}} \mathcal{L}_{\text{feat}} + \lambda_{\text{bg}} \mathcal{L}_{\text{bg}} + \lambda_{\text{perc}}^{\text{dyn}} \mathcal{L}_{\text{perc}} + \lambda_{\text{KL}} \mathcal{L}_{\text{KL}}. \tag{29}$$

We set the loss weights to $\lambda_{\text{img}} = 1$, $\lambda_{\text{feat}} = 1$, $\lambda_{\text{bg}} = 1$, $\lambda_{\text{perc}}^{\text{dyn}} = 0.1$, and $\lambda_{\text{KL}} = 0.001$. All foreground and background targets are extracted from the frozen representation model.

A.3.4 Analysis of time and memory

All methods are evaluated on a single NVIDIA A100 GPU. For BlockPush, we use the same evaluation sequences across methods with a batch size of 16, 6 conditioning frames, and 15 prediction frames. For Sketchy, we use a batch size of 4, 6 conditioning frames, and 44 prediction frames, again using the same evaluation sequences for all methods.

A.4 Behavior Learning

Policy learning. We learn a behavior policy that maps observed scene states to executable actions through the learned latent-action space. This stage uses demonstration sequences containing images and corresponding environment actions, denoted by $\mathcal{D}_{\text{demo}} = \{(I_{1:T}, a_{1:T-1})\}$. The pretrained representation model and latent-action inference model remain frozen. We train a latent-action policy π_ω and an action decoder D_η^{act}.

For each demonstration, we extract the scene states $\mathcal{X}_t = (X_t, s_t^{\text{bg}})$ and infer latent actions for the observed transitions. We use the inferred latent actions as supervision targets. The policy predicts latent actions from the current scene state:

$$\widehat{Z}_t = \pi_\omega(\mathcal{X}_t). \tag{30}$$

Foreground features and the background slot are separately projected into a common embedding space and jointly processed by a Transformer over the scene entities. Output heads map the resulting tokens to their corresponding latent actions. The policy receives no future observations and learns to predict the transition labels provided by the frozen inference model.

Decoding executable actions. An action decoder aggregates the predicted latent actions into a single environment action:

$$\widehat{a}_t = D_\eta^{\text{act}}(\widehat{Z}_t). \tag{31}$$

The decoder embeds the latent actions, appends a learned action token, and processes the tokens with a Transformer. An MLP maps the output action token to the environment's control space. This aggregation allows the executable action to depend on the predicted changes of multiple scene entities.

Loss. We freeze the representation and latent-action inference models and jointly train the policy π_ω and action decoder D_η^{act} using demonstrations with environment actions a_t. The policy predicts $\widehat{Z}_t = \pi_\omega(\mathcal{X}_t)$ from the current scene state, while the frozen inference model provides latent-action targets $Z_t^\star$. We minimize

$$\mathcal{L}_{\text{beh}} = \frac{1}{T-1}\sum_{t=1}^{T-1}\Big[\,\text{MSE}(\widehat{Z}_t, Z_t^\star) + \text{MSE}\big(D_\eta^{\text{act}}(\widehat{Z}_t), a_t\big)\Big]. \tag{32}$$

The first term supervises the policy with inferred latent actions. The second jointly trains the policy and decoder to reproduce demonstrated environment actions. Only the policy and action decoder are updated in this stage.

A.5 Additional Ablation Results

Adaptive feature weighting. Table 5 shows a consistent non-uniform weighting pattern across environments: spatial features receive the highest weight, followed by alpha and RGB features. This suggests that spatial cues dominate latent-action inference, while occupancy and appearance provide complementary information. The variation across environments further indicates that the fusion module adaptively adjusts their relative contributions.

Table 5: Average adaptive fusion weights (%) across different datasets.

Dataset	Spatial	Alpha	RGB
BlockPush	49.24	31.78	18.99
ButtonPress	51.46	36.60	11.94
PlaceBreadSkillet	53.88	27.77	18.35
Sketchy	51.92	28.40	19.68

Table 6: Ablation study on the slot dimension on Sketchy.

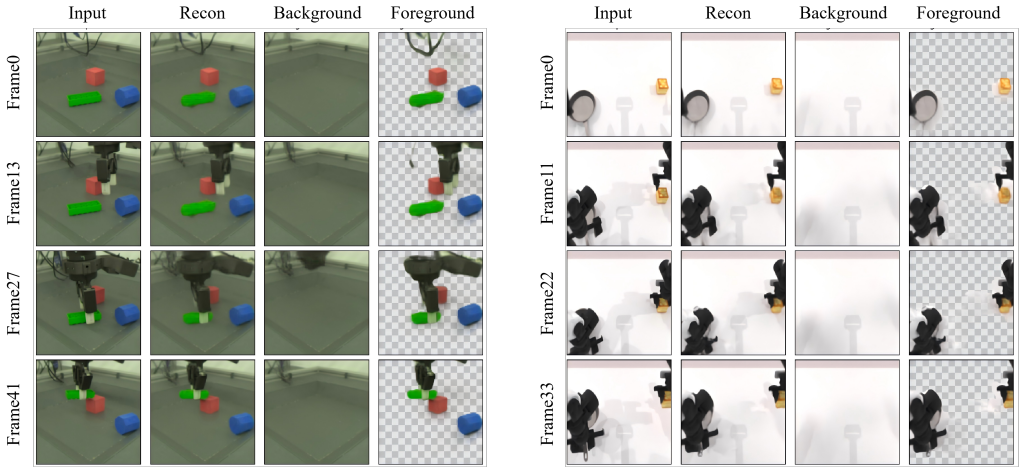
	Sketchy		
Slot dim	PSNR ↑	SSIM ↑	LPIPS ↓
64	23.23	0.76	0.235
128	27.98	0.86	0.102
256	<u>29.45</u>	<u>0.92</u>	<u>0.061</u>
512	**29.75**	**0.93**	**0.060**

Ablation on slot dimension. Table 6 studies the effect of the slot dimensionality on Sketchy. Increasing the slot dimension consistently improves prediction quality, with higher PSNR and SSIM and lower LPIPS. The improvement is particularly pronounced when increasing the dimension from 64 to 256, indicating that a sufficiently expressive latent representation is important for capturing complex object appearance and dynamics. Considering the marginal improvement beyond 256 dimensions, we use a slot dimension of 256 in our model.

Table 7: Ablation study on the latent action dimension on Sketchy.

	Sketchy		
Latent action dim	PSNR ↑	SSIM ↑	LPIPS ↓
3	28.14	0.88	0.077
7	<u>29.45</u>	**0.92**	**0.061**
16	**29.58**	**0.92**	**0.061**

Ablation on latent action dimension. Table 7 studies the effect of the latent action dimensionality on Sketchy. Increasing the latent action dimension from 3 to 7 leads to a clear improvement across all metrics, indicating that a very low-dimensional latent space is insufficient to capture the diversity



(a) Foreground-background seperation on Sketchy dataset

(b) Foreground-background seperation on PlaceBreadSkillet dataset

Figure 4: Foreground-background separation on Sketchy and PlaceBreadSkillet datasets. FOCWM consistently separates objects and background content while preserving high-quality reconstruction.

of object-level dynamics. Further increasing the dimension from 7 to 16 brings a marginal gain in PSNR. We therefore use a 7-dimensional latent action representation in our final model, which provides a compact representation without sacrificing prediction quality.

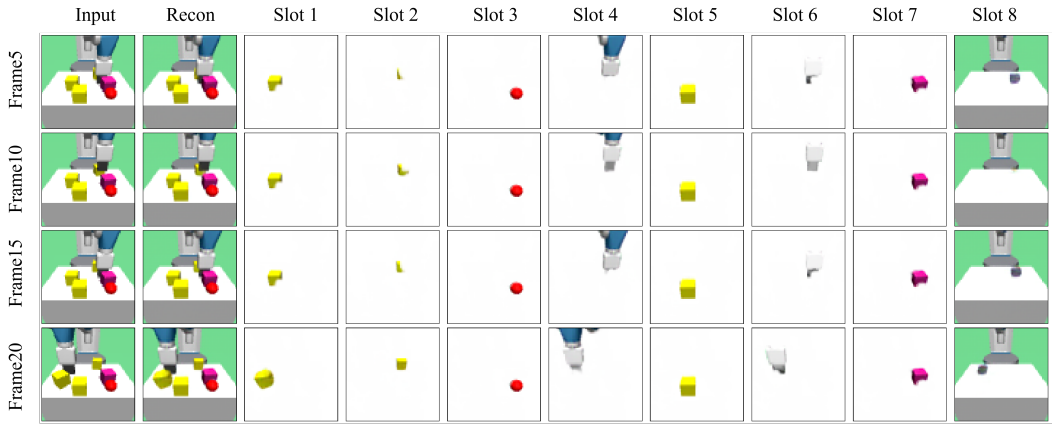
A.6 Visualization Results

Foreground-background decomposition. Figure 4 visualizes the learned foreground–background decomposition on Sketchy and PlaceBreadSkillet. Across different time steps, FOCWM consistently separates task-relevant objects and robot components into the foreground while capturing largely stationary scene content in the background. The reconstructed frames closely match the inputs, indicating that the explicit decomposition preserves visual information while providing a structured scene representation. Similar behavior across both datasets further demonstrates the robustness of the learned decomposition under different visual environments and robotic embodiments.

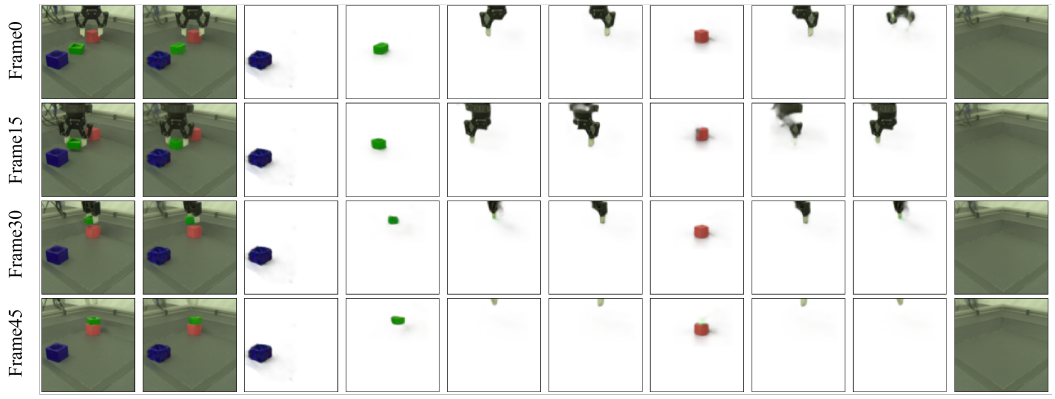
Object-wise slot decomposition. Figure 5 shows that FOCWM decomposes scenes into distinct object- and component-level slots while maintaining relatively consistent slot assignments over time. The reconstructed frames closely match the inputs, and similar behavior is observed on both Block-Push and Sketchy, indicating temporally stable object-centric representations across simulated and real-world settings. Nevertheless, some redundant slots remain in the learned decomposition. Despite these imperfections, the learned slots retain sufficient object-level structure for subsequent factorized dynamics modeling.

Analysis of factorized object representations. Figure 6 visualizes the learned factorized object representations. The spatial component localizes each object, the alpha feature captures its shape and occupancy, and the RGB feature preserves local appearance. Their combination reconstructs objects with consistent geometry and color, demonstrating that the three feature groups acquire distinct and complementary functional roles.

Manipulation of factorized object features. Figure 7 illustrates the controllability of the learned factorized representation. Starting from the reconstructed scene, modifying the spatial component independently changes the target object's scale, position, and depth ordering, while leaving the remaining scene unchanged. Manipulating the appearance component by replacing the appearance feature of the red object with that of the blue object changes the target object's visual appearance while preserving its other configuration. These results indicate that the learned feature groups capture functionally distinct and locally controllable object properties.



(a) Object-wise slot decomposition on BlockPush



(a) Object-wise slot decomposition on Sketchy

Figure 5: Object-wise slot decomposition across different video frames. FOCWM consistently captures distinct objects and backgrounds while maintaining relatively stable slot assignments over time.

Qualitative future prediction with a learned latent-action prior. In addition to the posterior latent-action distribution described in Sec. 3.2, we introduce a prior head that predicts latent actions without observing the subsequent state. Specifically, while the posterior distribution is inferred from the transition feature v_t^j, which contains information from both h_t^j and h_{t+1}^j, the prior only takes the current contextualized representation h_t^j as input.

For each scene entity $j \in \mathcal{J}$, the prior head predicts the mean and log variance of a Gaussian distribution:

$$\begin{aligned}(\mu_t^{j,p}, \ell_t^{j,p}) &= H_\psi^p(h_t^j),\\ p_\psi(z_t^j \mid \mathcal{X}_{1:t}) &= \mathcal{N}\Big(z_t^j; \mu_t^{j,p}, \operatorname{diag}\left(\exp(\ell_t^{j,p})\right)\Big), \qquad j \in \mathcal{J},\end{aligned} \tag{33}$$

where H_ψ^p denotes the prior distribution head.

During training, the posterior distribution $q_\psi(z_t^j \mid \mathcal{X}_{1:t+1})$ serves as the target distribution for the prior. We minimize

$$\mathcal{L}_{\text{prior}}^t = \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} D_{\text{KL}}\left(q_\psi(z_t^j \mid \mathcal{X}_{1:t+1}) \,\|\, p_\psi(z_t^j \mid \mathcal{X}_{1:t})\right). \tag{34}$$

This objective encourages the prior to approximate the latent-action distribution inferred from the complete transition using only the available state history.

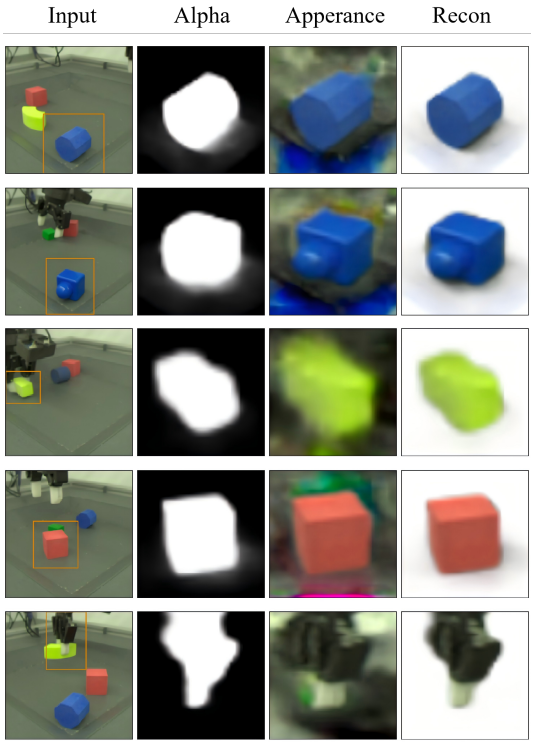


Figure 6: Visualization of the learned factorized object representations.

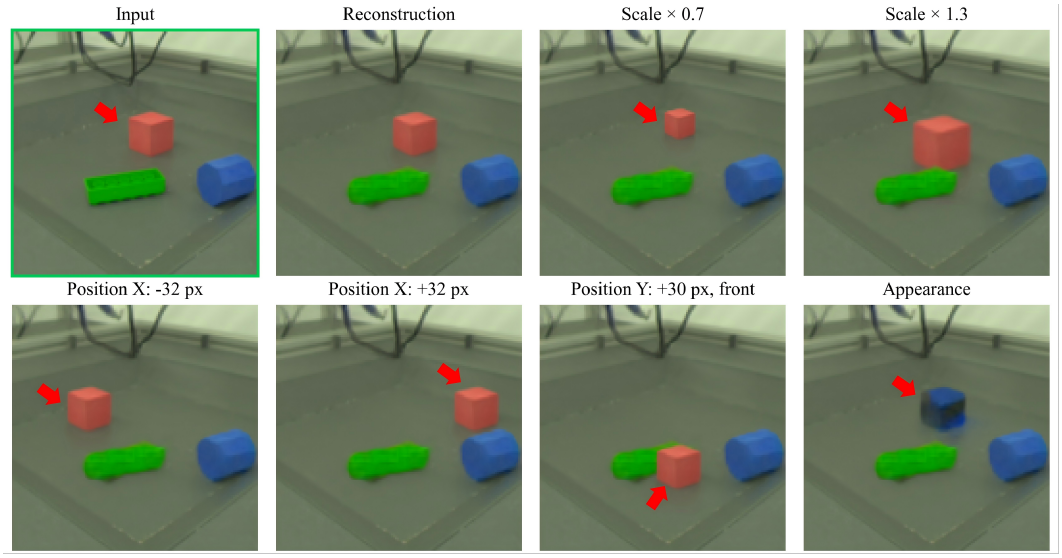


Figure 7: Manipulation of the learned factorized object representation. Modifying the spatial features controls the target object's scale, position, and depth, while changing the appearance feature alters its visual characteristics with minimal effects on the remaining scene.

Figure 8 provides a qualitative example of autoregressive prediction using latent actions sampled from the learned prior. Given six seed frames, the model preserves the scene structure and captures the subsequent robot motion over the rollout. We include this result to illustrate that the learned prior can be used when future transitions are unavailable.

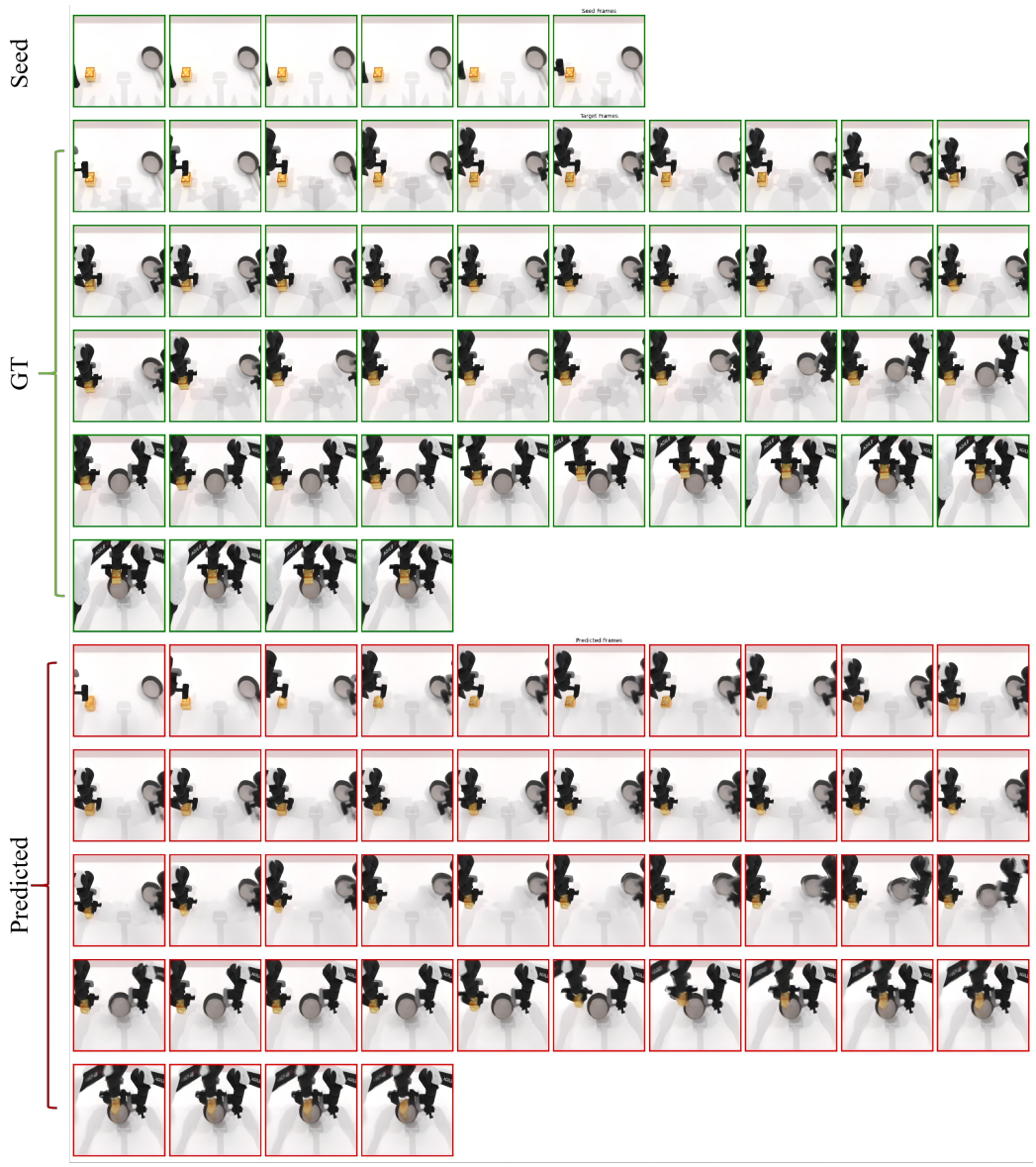


Figure 8: Future video prediction from six seed frames. FOCWM preserves the overall scene structure and captures the robot motion over extended rollouts, producing trajectories that remain largely consistent with the ground truth.