SLOT-MPC: GOAL-CONDITIONED MODEL PREDICTIVE CONTROL WITH OBJECT-CENTRIC REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Predictive world models enable agents to model scene dynamics and reason about the consequences of their actions. Inspired by human perception, object-centric world models capture scene dynamics using object-level representations, which can be used for downstream applications such as action planning. However, most object-centric world models and reinforcement learning (RL) approaches learn reactive policies that are fixed at inference time, limiting generalization to novel situations. We propose Slot-MPC, an object-centric world modeling framework that enables planning through Model Predictive Control (MPC). Slot-MPC leverages vision encoders to learn slot-based representations, which encode individual objects in the scene, and uses these structured representations to learn an action-conditioned object-centric dynamics model. At inference time, the learned dynamics model enables action planning via MPC, allowing agents to adapt to previously unseen situations. Since the learned world model is differentiable, we can use gradient-based MPC to directly optimize actions, which is computationally more efficient than relying on gradient-free, sampling-based MPC methods. Experiments on simulated robotic manipulation tasks show that Slot-MPC improves both task performance and planning efficiency compared to non-object-centric world model baselines. In the considered offline setting with limited state-action coverage, we find that gradient-based MPC performs better than gradient-free, sampling-based MPC. Our results demonstrate that explicitly structured, object-centric representations provide a strong inductive bias for controllable and generalizable decision-making. Code and additional material will be released at `https://slot-mpc.github.io`.

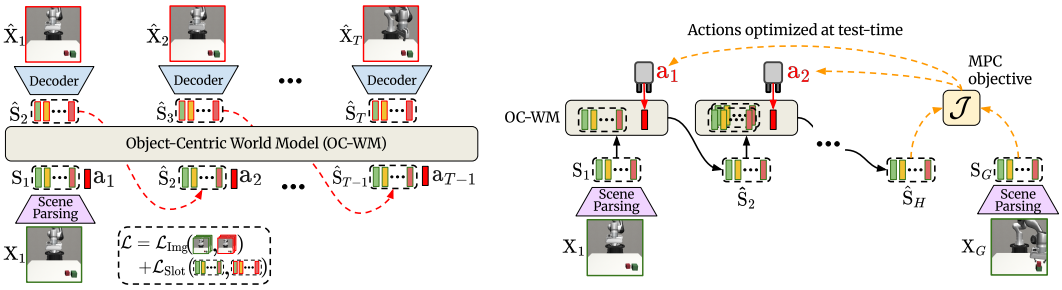
1 INTRODUCTION

The ability to predict how the world evolves in response to actions is fundamental to intelligent behavior. World models (Ha & Schmidhuber, 2018; LeCun, 2022) aim to equip artificial agents with this capability by learning predictive models that enable forecasting future environment states, supporting anticipation and planning. Humans, however, do not perceive the world as an unstructured stream of pixels. Instead, we parse scenes into persistent objects, which move independently, can interact with each other, or compose into more complex entities (Kahneman et al., 1992). Inspired by human perception, several works have investigated object-centric and compositional inductive biases for predictive modeling, demonstrating desirable properties such as generalization to novel compositions (Haramati et al., 2024), transferability to novel tasks (Zhang et al., 2022), and improvements in sample efficiency (Mosbach et al., 2025), among others.

Based on these insights, the field of object-centric learning has rapidly progressed in recent years, moving from learning object representations on synthetic images (Burgess et al., 2019; Locatello et al., 2020) and videos (Kipf et al., 2022), towards more complex real-world scenarios and datasets (Seitzer et al., 2023; Zadaianchuk et al., 2024). Additionally, the learned object representations have been used for world modeling and robotic tasks, including model-based RL (Ferraro et al., 2023; Mosbach et al., 2025) or imitation learning (Villar-Corrales & Behnke, 2025).

While effective, most object-centric RL policies are fixed at inference time, leading to purely reactive behaviors that limit generalization to novel situations (Byravan et al., 2022). Furthermore, online RL approaches are typically not sample-efficient and require large numbers of environment interactions, making them costly. Planning-based control offers a complementary alternative. Model Predictive

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) OC-WM training (b) Slot-MPC inference

Figure 1: Overview of Slot-MPC. **(a)** The object-centric world model (OC-WM) is trained given a single video frame $\mathbf{X}_1$ and actions $\mathbf{a}$, and autoregressively predicts future video frames and slot representations $\mathbf{S}$. **(b)** Slot-MPC parses an image $\mathbf{X}_1$ into its object representations $\mathbf{S}_1$ and then predicts the future object states over the horizon H using OC-WM given actions $\mathbf{a}$, which are optimized by MPC. The goal image $\mathbf{X}_G$ is also parsed into its object representations $\mathbf{S}_G$ and the slots at time step $\hat{\mathbf{S}}_T$ and $\mathbf{S}_G$ are used to optimize the actions using the MPC objective $\mathcal{J}$ defined in Eq. (9).

Control (Cutler & Ramaker, 1979; Richalet et al., 1978) (MPC) enables online planning by optimizing actions using a learned dynamics model at inference time. Recent works such as TD-MPC2 (Hansen et al., 2024) combine policy networks with MPC to leverage the benefits of both paradigms.

In this work, we propose Slot-MPC, a novel object-centric framework for goal-conditioned planning with MPC. Rather than relying only on a reactive policy, Slot-MPC learns a structured latent world model that enables online planning directly in an object-centric representation space. At inference time, as illustrated in Fig. 1b, Slot-MPC parses environment observations into a set of object slots, which represent individual entities present in the scene. This formulation enables planning directly in a structured object-centric latent representation, rather than relying on holistic scene features. Using an action-conditioned object-centric dynamics model, Slot-MPC predicts future object states from past slots and candidate action sequences, and optimizes actions via MPC by minimizing the distance between predicted and goal object configurations in slot space. Leveraging the differentiability of the learned world model, candidate action sequences are directly optimized by gradient descent, enabling efficient planning.

In our experiments, we demonstrate that Slot-MPC learns meaningful object representations, which enable more efficient planning compared to holistic, non-object-centric representations. Slot-based object-centric models reduce the latent space dimensionality by 99% compared to patch-based approaches such as DINO-WM (Zhou et al., 2025), which leads to more efficient planning. Furthermore, the explicit disentanglement of object-level entities enables more direct and controllable reasoning over scene dynamics, which is particularly beneficial in low-data and low-compute regimes.

In summary, our contributions are as follows:

- We introduce Slot-MPC, a gradient-based MPC method that uses a latent object-centric dynamics model and slot-based representations for goal-conditioned planning from purely visual inputs.
- We show that the learned slot-based representations enable more efficient gradient-based planning compared to holistic and patch-based latent representations.
- We demonstrate that Slot-MPC successfully solves complex, long-horizon planning tasks that remain challenging for non-object-centric approaches.

2 Related Work

Slot-Based Object-Centric Learning: Slot-based object-centric models aim to decompose visual scenes into a set of $N_\mathbf{S}$ latent embeddings, referred to as *slots*, where each slot represents a distinct object or entity in the scene (Locatello et al., 2020; Greff et al., 2020). Early approaches learned such object-centric representations end-to-end from synthetic images (Locatello et al., 2020; Burgess et al., 2019; Singh et al., 2022a) and videos (Kipf et al., 2022; Singh et al., 2022b; Zoran et al., 2021), demonstrating unsupervised scene decomposition under controlled settings. More recent

methods leverage pretrained vision encoders (Seitzer et al., 2023; Zadaianchuk et al., 2024; Aydemir et al., 2023) or additional supervision (Elsayed et al., 2022; Bao et al., 2023) to enable object-centric learning on complex real-world scenes. Beyond representation learning, slot-based representations have been shown to benefit a range of downstream task, including model-based RL (Mosbach et al., 2025), visual-question-answering (Mamaghan et al., 2025) or imitation learning (Villar-Corrales & Behnke, 2025).

Model-based RL and MPC: Model-based RL aims to learn a predictive model of environment dynamics, which can then be used to improve decision-making, either by learning policies from imagined experience (Hafner et al., 2020; 2025), or by planning actions at inference time (Sutton, 1991). Unlike model-free approaches, which learn reactive policies directly from reward signals, model-based methods explicitly reason about future state evolution, enabling improved sample efficiency and adaptability to new situations. However, purely policy-based solutions present limited generalization capabilities as their learned policies are solely reactive (Byravan et al., 2022). Recent works, such as TD-MPC2 (Hansen et al., 2024), combine policy networks with MPC for online trajectory optimization at inference time, demonstrating strong performance for a broad variety of different environments. In practice, planning is typically performed using gradient-free, sampling-based optimization methods such as the Cross-Entropy Method (CEM) (Rubinstein, 1997) or Model Predictive Path Integral control (MPPI) (Williams et al., 2015). Recent work (Sobal et al., 2025) shows the potential of gradient-free MPC (MPPI) with a latent dynamics model learned from reward-free offline data on a set of navigation tasks compared to goal-conditioned RL.

Object-Centric World Models and RL: Object-centric world models aim to explicitly model object dynamics and interactions in video sequences in order to forecast future object and scene states (Villar-Corrales et al., 2023; Wu et al., 2023). By representing scenes as compositions of persistent entities, these approaches provide a structured abstraction that is naturally suited for reasoning and decision-making. Object-centric representations have been explored for downstream control, both in model-free RL (Zadaianchuk et al., 2021; Mambelli et al., 2022; Haramati et al., 2024) and model-based settings (Ferraro et al., 2023; Mosbach et al., 2025). However, despite progress in reinforcement learning, the integration of slot-based representations with downstream planning remains relatively underexplored. Early approaches such as O2P2 (Janner et al., 2019) and OP3 (Veerapaneni et al., 2020) model object interactions but do not employ modern slot-based representations and rely on predefined high-level actions rather than low-level control signals. Moreover, planning in these works is typically performed using sampling-based, gradient-free optimization methods. Concurrently to our work, Nam et al. (2026) propose a slot-based world model evaluated in an MPC setting on the Push-T task. While conceptually similar, their approach is tightly coupled to the JEPA (LeCun, 2022) framework, does not match the performance of recent holistic world models such as DINO-WM (Zhou et al., 2025), and also relies solely on sampling-based MPC.

Gradient-based MPC: Sampling-based MPC methods typically evaluate hundreds or thousands of candidate trajectories at each decision step, resulting in substantial computational cost. When dynamics models are represented by differentiable neural networks, it is therefore natural to instead optimize action sequences directly using gradient-based optimization. The idea of gradient-based planning dates back to the 1960's (Kelley, 1960), yet its successful application with learned world models has remained limited. Prior approaches often rely on large amounts of expert demonstrations (Srinivas et al., 2018), are only evaluated on low-dimensional domains (Bharadhwaj et al., 2020; S V et al., 2023), or struggle to scale to realistic robotics settings (Henaff et al., 2018). Despite their potential benefits, most recent gradient-based MPC methods using learned world models empirically underperform their sampling-based counterparts, and incur high computational cost due to a large number of optimization iterations (Zhou et al., 2025; Parthasarathy et al., 2025; Terver et al., 2026). Recent work such as Dream-MPC (Spieler & Behnke, 2026) revisits this direction by combining gradient-based planning with learned reward and value functions within a model-based reinforcement learning framework. In contrast, we consider an object-centric dynamics model learned from offline, reward-free data and formulate planning as minimizing the distance between structured slot representations. This formulation removes the need for environment interaction during training and allows learning task-agnostic world models that can be reused across tasks within the same environment, enabling trajectory optimization at inference time without retraining.

3 SLOT-MPC

We propose Slot-MPC, a novel method that combines gradient-based MPC with a slot-based object-centric latent dynamics model. Fig. 1 illustrates the main components of our approach, as well as its training (Fig. 1a) and inference (Fig. 1b). Slot-MPC uses a *Scene Parsing* module (Section 3.1) to decompose an image $\mathbf{X}_t$ into object representations, called slots $\mathbf{S}_t = (\mathbf{s}_t^1, ..., \mathbf{s}_t^{N_\mathbf{S}}) \in \mathbb{R}^{N_\mathbf{S} \times D_\mathbf{S}}$, where $N_\mathbf{S}$ denotes the number of slots and $D_\mathbf{S}$ their dimensionality. Subsequently, a *Action-Conditional Object-Centric World Model* (OC-WM) autoregressively forecasts future object states over the prediction horizon H, conditioned on the initial parsed object slots and an action sequence, which can be randomly initialized or produced by a learned policy (Section 3.2). Given a goal image, the slots predicted at the final rollout step are compared with the goal slots – obtained by parsing the goal image – in order to optimize the action sequence via the MPC objective defined in Eq. (9) (Section 3.3).

3.1 OBJECT-CENTRIC LATENT DYNAMICS LEARNING

Slot-MPC builds upon SAVi (Kipf et al., 2022), a recursive encoder–decoder model that parses each frame of a video sequence into temporally aligned object representations. At time step t, SAVi encodes the input frame $\mathbf{X}_t$ into N permutation-equivariant object embeddings $\mathbf{S}_t \in \mathbb{R}^{N_\mathbf{S} \times D_\mathbf{S}}$, and uses *Slot Attention* (Locatello et al., 2020) to iteratively refine the previous slot representations conditioned on image features $\mathbf{h}_t \in \mathbb{R}^{L \times D_\mathbf{h}}$, where L denotes the number of spatial feature locations and $D_\mathbf{h}$ their dimensionality. Specifically, Slot Attention performs cross-attention between slots and image features, with attention coefficients normalized over the slot dimension in order to encourage slots to compete for representing feature locations:

$$\mathbf{A} = \mathrm{softmax}_{N_\mathbf{S}} \left(\frac{q(\mathbf{S}_{t-1}) \cdot k(\mathbf{h}_t)^T}{\sqrt{D_\mathbf{S}}} \right) \in \mathbb{R}^{N_\mathbf{S} \times L}, \tag{1}$$

where k and q are learned linear projections. The slots are then independently updated via a shared Gated Recurrent Unit (GRU) (Cho et al., 2014) followed by a residual Multi-Layer Perceptron (MLP):

$$\mathbf{S}_t = \mathrm{GRU}(\mathbf{A} \cdot v(\mathbf{h}_t), \mathbf{S}_{t-1}) \ , \quad \mathbf{A}_{n,l} = \frac{\mathbf{A}_{n,l}}{\sum_{i=0}^{L-1} \mathbf{A}_{n,i}}, \tag{2}$$

where v is a learned linear projection. The computations described in Eqs. (1) and (2) can be repeated multiple times with shared weights to iteratively refine the slot representations, producing an accurate object-centric representation of the scene.

To reconstruct images from slots, SAVi employs a slot decoder module. Namely, each slot $\mathbf{s}_t^n$ is independently processed by a Spatial Broadcast Decoder (Watters et al., 2019) ($\mathcal{D}_{\text{SAVi}}$) to produce an object image $\mathbf{o}_t^n$ and mask $\mathbf{m}_t^n$, which can be normalized and combined via a weighted sum to synthesize a video frame:

$$\mathbf{o}_t^n, \mathbf{m}_t^n = \mathcal{D}_{\text{SAVi}}(\mathbf{s}_t^n), \quad \forall\, \mathbf{s}_t^n \ \in \ \mathbf{S}_t, \tag{3}$$

$$\hat{\mathbf{X}}_t = \sum_{n=1}^{N_\mathbf{S}} \mathbf{o}_t^n \cdot \tilde{\mathbf{m}}_t^n \quad \text{with} \ \ \tilde{\mathbf{m}}_t^n = \mathrm{softmax}_{N_\mathbf{S}}(\mathbf{m}_t^n). \tag{4}$$

SAVi is trained self-supervised using an image reconstruction loss:

$$\mathcal{L}_{\text{SAVi}} = \sum_{t=1}^{T} ||\mathcal{D}_{\text{SAVi}}(\mathcal{E}_{\text{SAVi}}(\mathbf{X}_t)) - \mathbf{X}_t||_2^2, \tag{5}$$

where $\mathcal{E}_{\text{SAVi}}$ and $\mathcal{D}_{\text{SAVi}}$ correspond to the scene parsing and object rendering modules, respectively.

Inspired by OCVP (Villar-Corrales et al., 2023), we adopt a transformer-based (Vaswani et al., 2017) latent dynamics model that autoregressively predicts future objects slots conditioned on past object states. The model leverages self-attention to capture object dynamics and agent-object interactions while preserving the permutation equivariance of the slot representations.

We extend the OCVP predictor to an action-conditional setting (OC-WM), enabling the model to explicitly account for control inputs when forecasting future object dynamics. At each time step,

action vectors $\mathbf{a}_t \in \mathbb{R}^{N_a}$ are mapped into the predictor embedding space through a learnable linear projection $f_{\mathbf{a}}$ and combined additively with the slot representations to form the predictor input. Given past object slots and the corresponding action sequence, the model autoregressively estimates the next slot state:

$$\hat{\mathbf{S}}_{t+1} = \text{OC-WM}\big(\mathbf{S}_1, f_{\mathbf{a}}(\mathbf{a}_1), \ldots, \mathbf{S}_t, f_{\mathbf{a}}(\mathbf{a}_t)\big). \tag{6}$$

Starting from a single seed frame, this process is applied autoregressively by feeding predicted slots back as inputs, allowing future states to be generated over a prediction horizon H.

Given the pretrained SAVi model, we train OC-WM by minimizing a combined objective:

$$\mathcal{L}_{\text{OC-WM}} = \sum_{t=2}^{T+1} \underbrace{\lambda_{\text{Img}} \cdot ||\hat{\mathbf{X}}_t - \mathbf{X}_t||_2^2}_{\text{future frame prediction}} + \underbrace{\lambda_{\text{Slot}} \cdot ||\hat{\mathbf{S}}_t - \mathcal{E}_{\text{SAVi}}(\mathbf{X}_t)||_2^2}_{\text{joint-embedding alignment}}, \tag{7}$$

where λ_{Img} and λ_{Slot} are scalar coefficients that balance the contribution of each loss term. The first loss penalizes frame prediction errors, encouraging accurate visual forecasting, whereas the second term aligns predicted slots with the object-centric representations inferred from the corresponding ground-truth frames, stabilizing latent dynamics learning and improving temporal consistency.

3.2 Policy Learning

Initializing MPC from randomly sampled action sequences frequently results in suboptimal solutions, since the optimizer lacks a meaningful starting point in high-dimensional action spaces. Prior work has shown that warm-starting MPC with an informed initial action proposal can substantially improve convergence and optimization stability (Parmas et al., 2018; Hansen et al., 2022). Motivated by this observation, we learn a policy network that provides an informed initialization for the MPC procedure.

To this end, we train a policy model via behavior cloning from a small set of expert demonstrations. Given a pretrained SAVi object-centric decomposition model, the policy network π_θ is trained to predict the expert actions $\mathbf{a}_t$ from the corresponding structured slot-based latent representations $\mathbf{S}_t$:

$$\mathcal{L}_{\pi_\theta} = \sum_{t=1}^{T} ||\pi_\theta(\mathbf{S}_t) - \mathbf{a}_t||_2^2. \tag{8}$$

While such a policy may have limited generalization capabilities, we hypothesize that it can provide a strong initial action proposal, enabling more efficient MPC optimization and faster convergence.

3.3 Model Predictive Control

At inference time, we perform model predictive control (MPC) in the learned latent object-centric space in order to optimize actions sequences for reaching a goal state. Given a goal image, we first encode it using the object-centric encoder to obtain a set of latent goal slots $\mathbf{S}_{\text{Goal}}$ representing the target object configuration. Starting from the current observation at time step t, Slot-MPC encodes the image into latent object slots $\mathbf{S}_t$, which serve as the initial state for planning. The learned dynamics model autoregressively forecasts future latent slot states conditioned on the initial slot representations and a candidate action sequence $\mathbf{a}_{t:t+H-1}$, producing the predicted slot configuration $\hat{\mathbf{S}}_{t+H}$ at planning horizon H.

MPC optimizes the action sequence by minimizing the distance between the last predicted latent slot configuration $\hat{\mathbf{S}}_{t+H}$ and the latent goal slots $\mathbf{S}_{\text{Goal}}$. Formally, the MPC objective is defined as

$$\mathcal{J}_{\text{MPC}} = ||\hat{\mathbf{S}}_{t+H} - \mathbf{S}_{\text{Goal}}||_2^2. \tag{9}$$

At each optimization iteration, MPC evaluates candidate action sequences by rolling them out through the learned dynamics model, updates them based on their predicted costs, and executes only the first action of the optimized sequence before replanning at the next time step.

Since the ordering of object slots is not guaranteed to be consistent across time steps or between predicted and goal states for object-centric models, we apply Hungarian matching when computing

MPC costs to ensure a meaningful comparison between object-centric latent states. At each time step, predicted object slots are aligned to goal slots by minimizing the pairwise Euclidean distance in latent space. The MPC costs are then computed after alignment, using the matched object representations.

We evaluate two different MPC variants, namely *MPPI* (Williams et al., 2015) and *gradient-based MPC*. Following Hansen et al. (2024), we additionally leverage a learned policy network to warm-start the optimization by providing good initial action trajectories. The policy is rolled out over the planning horizon using the dynamics model. We will focus on gradient-based MPC in the following, but details on MPPI are included in Appendix C.4.

Gradient-based MPC: A differentiable learned dynamics model enables computing gradients of the planning objective with respect to the action sequence by backpropagating through the rollout, enabling gradient-based MPC to directly optimize a single candidate action sequence instead of sampling hundreds of trajectories as in MPPI. Specifically, the actions are optimized by minimizing the planning objective (Eq. (9)) via gradient descent:

$$\mathbf{a} \leftarrow \mathbf{a} - \eta \nabla \mathcal{J}_{\text{MPC}}. \tag{10}$$

Inspired by Spieler & Behnke (2026), we use a policy network to efficiently guide the MPC procedure, which has shown to be beneficial for stabilizing gradient-based MPC. In contrast to model-based RL methods, we do not jointly train the policy and world model, but instead learn each component independently from offline datasets without reward signals. Rather than relying on learned reward or value functions, we directly optimize the object-wise distance between predicted and goal object slots using latent-space distances as the planning objective. This alternative formulation enables visual goal-directed planning without requiring rewards during training or planning. Additionally, instead of sampling multiple trajectories from the policy network, Slot-MPC directly optimizes a single trajectory initialized by the policy, thereby improving optimization efficiency.

4 Experiments

To evaluate the effectiveness of our object-centric planning framework, we investigate the following research questions: (i) Can an object-centric world model be learned purely from pre-collected offline trajectories and subsequently be used for goal-conditioned planning? (ii) Do object-centric representations enable more efficient planning compared to holistic scene representations? (iii) What components are required to efficiently solve long-horizon manipulation tasks? We evaluate our proposed method on five robotic manipulation environments and compare against state-of-the-art baselines to answer these questions.

4.1 Evaluation Setup

Datasets: We evaluate our proposed approach on five robotic manipulation environments adapted from two different benchmarks: *Button Press* and *Lever Pull* from Meta-World (Yu et al., 2020), and *Stack*, *Square* and *Stack Three* from robosuite (Zhu et al., 2020). For each environment, we generate two offline datasets: (i) trajectories collected by a random exploration agent, and (ii) a small set of expert demonstrations solving the corresponding task. A mixed dataset consisting of 25% expert data and 75% random exploration data is used to train both the object-centric scene parsing model and the structured dynamics model, whereas the expert demonstrations are used to train the behavior cloning policy that warm-starts MPC. We use only visual observations in all environments and do not rely on additional inputs such as proprioceptive states. Further details are provided in Appendix B.

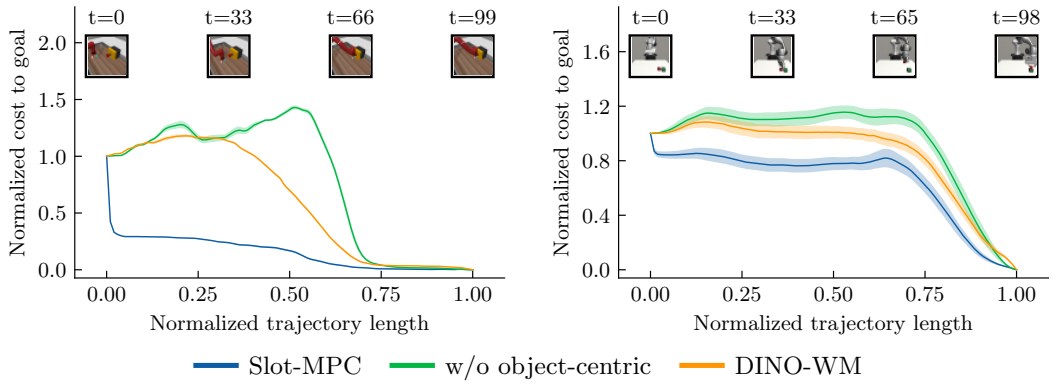
Baselines: We compare Slot-MPC against established baselines for both online and offline reinforcement learning, including goal-conditioned behavior cloning (GC-BC) (Lynch et al., 2020; Ghosh et al., 2021), Dreamer-v3 (Hafner et al., 2025), the world models DINO-WM (Zhou et al., 2025) and LeWM (Maes et al., 2026), and the object-centric methods EC-Diffuser (Qi et al., 2025) and LPWM (Daniel et al., 2026). DINO-WM learns a world model that operates in the latent space of a pretrained DINOv2 (Oquab et al., 2024) encoder using offline, reward-free data. LeWM learns a latent JEPA world model end-to-end from visual inputs. Both DINO-WM and LeWM then use the learned dynamics model for planning via CEM, a gradient-free, sampling-based MPC technique. Goal-conditioned behavior cloning (GC-BC) learns a goal-conditioned policy from reward-free,

offline data via behavior cloning. EC-Diffuser is an object-centric diffusion-based BC method, which uses Deep Latent Particles (DLP) (Daniel & Tamar, 2022) to represent objects. LPWM learns an object-centric world model using DLP-based representations, which can be used for imitation learning. For a fair comparison, all methods except for Dreamer-v3 are trained using the same offline datasets. Dreamer-v3 is a widely used model-based RL method that jointly learns a visual world model and a policy from reward signals obtained through interaction with the environment. Since this method requires rewards for training, we do not use the offline datasets, but instead follow its intended training protocol and learn the policy through online environment interactions. Further details are provided in Appendix D.

Success definition: For a fair comparison, all methods are evaluated using the same protocol. At the beginning of each evaluation episode, the simulator state is initialized using the first state of an evaluation trajectory. All methods except for Dreamer-v3 additionally receive a goal image depicting successful task completion in the current environment. At each time-step, all methods receive the current visual observation and select an action according to their policy. The actions are executed in the environment and the subsequent observation is returned. This procedure is repeated until either the goal is successfully reached or the maximum episode length is exceeded. Our evaluation protocol differs from the procedure used by DINO-WM, which only considers short randomly sampled sub-trajectories. Instead, we evaluate full episodes, which better reflects long-horizon planning performance and task completion. We report success rates across 100 evaluation episodes. Results obtained using the original DINO-WM evaluation protocol are provided in Appendix E.2.

4.2 Analysis of the distances in latent space

We first analyze the learned latent space and compare distances in holistic latent space to slot-based latent distances before we compare the downstream planning performance. Inspired by Wang et al. (2026), we visualize distances in latent space for expert demonstrations from the dataset. As shown in Fig. 2, distances in holistic and patch-based latent space do overall not reflect the progress of moving towards the goal well. Slot-based distances in contrast reflect the progress to the target more meaningful, providing a smoother trajectory and more informative proxy for progress towards the goal. This indicates that planning in slot-based latent space may lead to better planning performance.



(a) Button Press (b) Stack

Figure 2: Successful trajectories of Button Press and Stack in slot-based and holistic latent space. The distances are used as the MPC objectives. The curves show the mean normalized distances over 100 evaluation episodes and the shaded areas represent the 95% confidence intervals. The images on top show an example rollout. Slot-based distances are a more meaningful proxy for progress towards the goal. More examples are shown in Appendix E.4.

4.3 Comparison to Baselines

A quantitative comparison to the baselines is presented in Tab. 1 and shows that slot-based representations significantly improve success rates. Slot-MPC achieves strong results across all environments.

Table 1: Task success comparison of Slot-MPC with state-of-the-art baselines. Slot-MPC outperforms all baselines. Best two results are highlighted boldface and underlined, respectively. Wilson 95% confidence intervals are shown in brackets.

	Success Rates ↑				
Method	**Button Press**	**Lever Pull**	**Stack**	**Square**	**Stack Three**
Slot-MPC	**1.00** [0.96, 1.00]	**0.64** [0.54, 0.73]	**0.78** [0.69, 0.85]	**0.26** [0.18, 0.35]	**0.08** [0.04, 0.15]
DINO-WM	0.00 [0.00, 0.04]	0.00 [0.00, 0.04]	0.00 [0.00, 0.04]	0.00 [0.00, 0.04]	0.00 [0.00, 0.04]
LeWM	0.35 [0.26, 0.45]	0.07 [0.03, 0.14]	0.00 [0.00, 0.04]	0.00 [0.00, 0.04]	0.00 [0.00, 0.04]
Dreamer-v3	<u>0.66</u> [0.56, 0.75]	<u>0.59</u> [0.49, 0.68]	0.19 [0.13, 0.28]	0.00 [0.00, 0.04]	0.00 [0.00, 0.04]
GC-BC	0.43 [0.34, 0.53]	0.08 [0.04, 0.15]	0.33 [0.25, 0.43]	0.00 [0.00, 0.04]	0.06 [0.03, 0.12]
EC-Diffuser	0.64 [0.54, 0.73]	0.21 [0.14, 0.30]	<u>0.71</u> [0.61, 0.79]	<u>0.16</u> [0.10, 0.24]	<u>0.07</u> [0.03, 0.14]
LPWM	0.49 [0.39, 0.59]	0.14 [0.09, 0.22]	0.05 [0.02, 0.11]	0.00 [0.00, 0.04]	0.00 [0.00, 0.04]

Our results highlight that methods such as DINO-WM or LeWM, which do not use a policy to warm-start sampling-based MPC, fail for tasks that require planning over longer horizons. Increasing the number of candidate trajectories and optimization iterations did not help to overcome this issue, likely due to the large search space and suboptimal local optima. Methods using a policy network perform better, particularly those using object-centric representations such as latent particles or slots. In the considered setting, the combination of slot-based representations, a policy network and gradient-based MPC achieves the best performance. We hypothesize that this arises because holistic latent representations would require substantially higher state-action coverage compared to object-centric representations, which provide a compositional inductive bias. As a result, our model exhibits strong generalization capabilities and overcomes limitations commonly observed in offline-trained policies, which generalize poorly and rely heavily on high-quality data and broad state-action coverage (Sobal et al., 2025). We further include results for the original evaluation procedure used for DINO-WM in Tab. 9, which show that the success rate in this setting also significantly decreases when removing proprioceptive information and considering longer horizons.

4.4 Model Analysis

Component ablations: We conduct ablation studies to isolate the contribution of the different design choices in Slot-MPC, including the object-centric world model, the policy prior, and the MPC formulation. The results in Tab. 2 show that the object-centric world model, the learned policy network and gradient-based MPC are the most critical components, as removing them leads to the largest performance drops. In particular, we find that tasks requiring long planning horizons (e.g. those from robosuite) strongly benefit from a good policy prior to guide the optimization process through the search space. Our experiments further show that gradient-based MPC is substantially more effective than sampling-based MPC methods such as MPPI in the considered offline setting, where state-action coverage is limited. Even when using the policy prior to guide the sampling procedure, the performance deteriorates for MPPI significantly. We hypothesize that this is due to the dynamics model being queried out of distribution, leading to suboptimal optimization. In this setting, the characteristics of gradient-based MPC, which stays closer to the nominal policy distribution, are more beneficial than the higher diversity of sampling-based MPC methods.

Table 2: Slot-MPC component ablation studies. Impact of removing key components, including object-centric representations, MPC, and policy initialization, as well as replacing gradient-based MPC with MPPI. Best two results are bolded and underlined, respectively.

	Success Rates ↑				
Model Variant	**Button Press**	**Lever Pull**	**Stack**	**Square**	**Stack Three**
Slot-MPC	**1.00**	**0.64**	**0.78**	**0.26**	**0.08**
w/o object-centric representations	0.53	0.50	0.37	0.02	0.01
w/o MPC	<u>0.60</u>	<u>0.57</u>	<u>0.71</u>	<u>0.23</u>	<u>0.06</u>
w/o policy (zeros as initial actions)	0.02	0.19	0.00	0.00	0.00
w/ MPPI (no gradient-based MPC)	0.10	0.08	0.00	0.00	0.00

MPC objective: We also perform an ablation study of the MPC objective by using different formulations of slot-based cost functions. The results, summarized in Tab. 8, show that performing MPC with a cost function using the sum of squared distances (SSE) of slots performs best. Using the slot masks $\tilde{\mathbf{m}}_t^n$ is biased towards having accurate modeling of the background and larger objects instead of small objects. While this bias can be mitigated by using the normalized intersection over union (IoU) between slot masks, directly optimizing over slot representations is preferable, since the same representation space is also used during predictor training.

Planning time: Since inference time is a critical factor when deploying a model to a real-world system such as a robot, we report the time required for a single planning step on a single NVIDIA RTX A6000 GPU in Tab. 3. Object-centric slot representations substantially reduce the latent feature space compared to patch-based methods, such as DINO-WM, reducing the representation size from $\texttt{\#Tokens} \times D_h$ to $N_\mathbf{S} \times D_\mathbf{S}$. In our setting, we use four slots (five for Stack Three) with a dimensionality of 128, resulting in a latent space of 4×128 compared to 196×384 for DINO-WM. This corresponds to an approximately 99% reduction in latent dimensionality and leads to significantly faster planning times, even when using sampling-based MPPI. Additionally, warm-starting MPC by using a policy prior further improves computational efficiency of Slot-MPC.

Table 3: Comparison of planning execution times for different methods on a single NVIDIA RTX A6000 GPU. Slot-MPC is substantially more computationally efficient than the patch-based baseline.

	Planning time (s) ↓	
Method	**Meta-World**	**robosuite**
Slot-MPC	**0.02 ± 0.00**	**0.07 ± 0.01**
w/ MPPI (no gradient-based MPC)	4.22 ± 0.06	5.19 ± 0.03
DINO-WM	144.37 ± 0.83	145.30 ± 0.72
LeWM	<u>0.89 ± 0.01</u>	<u>0.89 ± 0.04</u>

4.5 Robustness to Distribution Shifts

We evaluate the robustness of Slot-MPC under visual shifts, which are described and shown in Appendix E.5. Slot-MPC retains the highest success rates while its variants without MPC and without object-centric representations suffer from a larger performance degradation.

Table 4: Task success of Slot-MPC and its ablations under distribution shifts. Relative drop (in parentheses) compared to the original evaluation. Visualizations of the shifts and details are provided in Appendix E.5. Best two results are highlighted boldface and underlined, respectively.

		Success Rates ↑			
Environment	**Method**	**Object Color**	**Robot Arm Transparency**	**Camera View**	**Light Direction**
Button Press	Slot-MPC	**0.83** <u>(-17%)</u>	**0.99 (-1%)**	**0.93** <u>(-7%)</u>	**0.86 (-14%)**
	w/o MPC	0.39 (-35%)	0.35 (-42%)	<u>0.60</u> **(0%)**	<u>0.42</u> (-23%)
	w/o object-centric representations	<u>0.48</u> **(-9%)**	<u>0.46</u> <u>(-13%)</u>	0.11 (-79%)	<u>0.42</u> <u>(-21%)</u>

5 Conclusion

We study the importance of object-centric representations for efficient planning with MPC and introduce Slot-MPC, a novel method for performing gradient-based MPC with object-centric latent dynamics. Slot-MPC learns object-centric representations from reward-free offline data and combines a slot-based world model with gradient-based MPC for goal-conditioned planning directly from visual observations. By optimizing action trajectories in a compact object-centric latent space, Slot-MPC significantly improves both task performance and computational efficiency compared to state-of-the-art baselines. Our experiments on simulated robotic manipulation tasks demonstrate that object-centric representations are a powerful inductive bias for control, particularly for long-horizon planning problems. Overall, our results suggest that structured object-centric world models are a promising direction for scalable and efficient model-based control from raw visual observations.

AI USE STATEMENT

In this work, we used generative AI tools only as coding assistants. In particular, they were used to edit and refactor existing code, identify and help resolve software bugs, and implement auxiliary functionalities, such as scripts for plotting and data visualization. All AI-assisted work was reviewed and tested by the authors. We have not used generative AI tools for data generation, research methodology, or paper writing. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All environments and baselines for which we rely on third-party code are open-source and have permissible licenses. Our experiments are intended for research purposes, so applying the proposed method to safety-critical or real-world settings requires careful consideration of ethical and safety-relevant aspects such as out-of-distribution generalization or a lack of transparency in the decision-making process.

REPRODUCIBILITY STATEMENT

To ensure reproducibility of our work, we have included details on the environments and datasets in Appendix B and implementations details and hyperparameters of our proposed method as well as for the baselines in Appendix C. Code, datasets and more will be open sourced at `https://slot-mpc.github.io`.

REFERENCES

Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. In *International Conference on Learning Representations (ICLR)*, 2021.

Görkay Aydemir, Weidi Xie, and Fatma Guney. Self-supervised object-centric learning for videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Zhipeng Bao, Pavel Tokmakov, Yu-Xiong Wang, Adrien Gaidon, and Martial Hebert. Object discovery from motion-guided tokens. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Homanga Bharadhwaj, Kevin Xie, and Florian Shkurti. Model-predictive control via cross-entropy and gradient-based optimization. In *Conference on Learning for Dynamics and Control (L4DC)*, 2020.

Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. MONet: Unsupervised scene decomposition and representation. *arXiv:1901.11390*, 2019.

Arunkumar Byravan, Leonard Hasenclever, Piotr Trochim, Mehdi Mirza, Alessandro Davide Ialongo, Yuval Tassa, Jost Tobias Springenberg, Abbas Abdolmaleki, Nicolas Heess, Josh Merel, and Martin A. Riedmiller. Evaluating model-based planning and planner amortization for continuous control. In *International Conference on Learning Representations (ICLR)*, 2022.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

C. R. Cutler and B.L. Ramaker. Dynamic matrix control - A computer control algorithm. *IEEE Transactions on Automatic Control (TAC)*, 17:72, 1979.

Tal Daniel and Aviv Tamar. Unsupervised image representation learning with deep latent particles. In *International Conference on Machine Learning (ICML)*, 2022.

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. In *International Conference on Learning Representations (ICLR)*, 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Stefano Ferraro, Pietro Mazzaglia, Tim Verbelen, and Bart Dhoedt. FOCUS: Object-centric world models for robotics manipulation. In *Advances in Neural Information Processing Systems Workshops (NeurIPSw)*, 2023.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Manon Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Klaus Greff, Sjoerd Van Steenkiste, and Jürgen Schmidhuber. On the binding problem in artificial neural networks. *arXiv:2012.05208*, 2020.

David Ha and Jürgen Schmidhuber. World models. *arXiv:1803.10122*, 2018.

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

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

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

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

Dan Haramati, Tal Daniel, and Aviv Tamar. Entity-centric reinforcement learning for object manipulation from pixels. In *International Conference on Learning Representations (ICLR)*, 2024.

Mikael Henaff, William F. Whitney, and Yann LeCun. Model-based planning with discrete and continuous actions. *arXiv:1705.07177*, 2018.

Michael Janner, Sergey Levine, William T. Freeman, Joshua B. Tenenbaum, Chelsea Finn, and Jiajun Wu. Reasoning about physical interactions with object-oriented prediction and planning. In *International Conference on Learning Representations (ICLR)*, 2019.

Daniel Kahneman, Anne Treisman, and Brian J Gibbs. The reviewing of object files: Object-specific integration of information. *Cognitive psychology*, 24(2):175–219, 1992.

Henry J. Kelley. Gradient theory of optimal flight paths. *ARS Journal*, 30(10):947–954, 1960.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 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 (ICLR)*, 2022.

Yann LeCun. A path towards autonomous machine intelligence version 0.9.2, 2022-06-27, 2022. URL `https://openreview.net/pdf?id=BZ5a1r-kVsf`.

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 (NeurIPS)*, 2020.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on Robot Learning (CoRL)*, 2020.

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

Amir Mohammad Karimi Mamaghan, Samuele Papa, Karl Henrik Johansson, Stefan Bauer, and Andrea Dittadi. Exploring the effectiveness of object-centric representations in visual question answering: Comparative insights with foundation models. In *International Conference on Learning Representations (ICLR)*, 2025.

Davide Mambelli, Frederik Träuble, Stefan Bauer, Bernhard Schölkopf, and Francesco Locatello. Compositional multi-object reinforcement learning with linear relation networks. *arXiv:2201.13388*, 2022.

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. In *Conference on Robot Learning (CoRL)*, 2023.

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 *International Conference on Machine Learning (ICML)*, 2025.

Heejeong Nam, Quentin Le Lidec, Lucas Maes, Yann LeCun, and Randall Balestriero. Causal-JEPA: Learning world models through object-level latent interventions. In *International Conference on Machine Learning (ICML)*, 2026.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research (TMLR)*, 2024.

Paavo Parmas, Carl Edward Rasmussen, Jan Peters, and Kenji Doya. PIPPS: Flexible model-based policy search robust to the curse of chaos. In *International Conference on Machine Learning (ICML)*, 2018.

Arjun Parthasarathy, Nimit Kalra, Rohun Agrawal, Yann LeCun, Oumayma Bounou, Pavel Izmailov, and Micah Goldblum. Closing the train-test gap in world models for gradient-based planning. *arXiv:2512.09929*, 2025.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *International Conference on Neural Information Processing Systems Workshops (NeurIPSw)*, 2017.

Michael Psenka, Michael Rabbat, Aditi Krishnapriyan, Yann LeCun, and Amir Bar. Parallel stochastic gradient-based planning for world models. In *International Conference on Machine Learning (ICML)*, 2026.

Carl Qi, Dan Haramati, Tal Daniel, Aviv Tamar, and Amy Zhang. Ec-diffuser: Multi-object manipulation via entity-centric behavior generation. In *International Conference on Learning Representations (ICLR)*, 2025.

Jacques Richalet, André Rault, Jean-Louis Testud, and Jean Papon. Model predictive heuristic control. *Automatica*, 14(5):413–428, 1978.

Reuven Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research (EJOR)*, 1997.

Jyothir S V, Siddhartha Jalagam, Yann LeCun, and Vlad Sobal. Gradient-based planning with world models. *arXiv:2312.17227*, 2023.

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

Gautam Singh, Fei Deng, and Sungjin Ahn. Illiterate DALL-E learns to compose. In *International Conference on Learning Representations (ICLR)*, 2022a.

Gautam Singh, Yi-Fu Wu, and Sungjin Ahn. Simple unsupervised object-centric learning for complex and naturalistic videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022b.

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

Jonathan Spieler and Sven Behnke. Dream-MPC: Gradient-based model predictive control with latent imagination. In *International Conference on Machine Learning (ICML)*, 2026.

Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks: Learning generalizable representations for visuomotor control. In *International Conference on Machine Learning (ICML)*, 2018.

Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4):160–163, 1991.

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 (TMLR)*, 2026.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

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

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

Angel Villar-Corrales, Ismail Wahdan, and Sven Behnke. Object-centric video prediction via decoupling of object dynamics and interactions. In *IEEE International Conference on Image Processing (ICIP)*, 2023.

Angel Villar-Corrales, Gjergj Plepi, and Sven Behnke. TextOCVP: Object-centric video prediction with language guidance. *Transactions on Machine Learning Research (TMLR)*, 2026.

Ying Wang, Oumayma Bounou, Gaoyue Zhou, Randall Balestriero, Tim G. J. Rudner, Yann LeCun, and Mengye Ren. Temporal straightening for latent planning. In *International Conference on Machine Learning (ICML)*, 2026.

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 (TIP)*, 13(4): 600–612, 2004.

Nicholas Watters, Loic Matthey, Christopher P Burgess, and Alexander Lerchner. Spatial broadcast decoder: A simple architecture for learning disentangled representations in VAEs. *arXiv:1901.07017*, 2019.

Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. Model predictive path integral control using covariance variable importance sampling. *arXiv:1509.01149*, 2015.

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 (ICLR)*, 2023.

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 (CoRL)*, 2020.

Andrii Zadaianchuk, Maximilian Seitzer, and Georg Martius. Self-supervised visual reinforcement learning with object-centric representations. In *International Conference on Learning Representations (ICLR)*, 2021.

Andrii Zadaianchuk, Maximilian Seitzer, and Georg Martius. Object-centric learning for real-world videos by predicting temporal feature similarities. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Chuhan Zhang, Ankush Gupta, and Andrew Zisserman. Is an object-centric video representation beneficial for transfer? In *Asian Conference on Computer Vision (ACCV)*, 2022.

Ellina Zhang, Madhavan Iyengar, Amir Zadeh, Chuan Li, David Held, Deepak Pathak, and Tal Daniel. 3D-DLP: Self-supervised 3d object-centric scene representation learning. In *International Conference on Machine Learning (ICML)*, 2026.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

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

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Kevin Lin, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv:2009.12293*, 2020.

Daniel Zoran, Rishabh Kabra, Alexander Lerchner, and Danilo J Rezende. PARTS: Unsupervised segmentation with slots, attention and independence maximization. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

A Limitations and Future Work

Slot-MPC relies like DINO-WM and LeWM on goal images for planning and on ground truth actions for training the dynamics model and policy, which both may not always be available. An extension of this work could involve text conditioning (Villar-Corrales et al., 2026), which allows for specifying the goal using natural language. Performance depends on the quality of the object-centric decomposition model and the policy prior. Further improving performance by using subgoals and extending the framework to real-world robotic settings is part of future work, potentially leveraging more capable decomposition models such as DINOSAUR (Seitzer et al., 2023).

B Datasets and Simulation Environments

We evaluate Slot-MPC and baseline models on five tasks from different manipulation benchmarks. Image inputs are resized to a resolution of 64×64 for Slot-MPC, GC-BC, EC-Diffuser, and Dreamer-v3, 128×128 for LPWM and 224×224 for DINO-WM and LeWM. The environments are illustrated in Fig. 3.

Meta-World (Yu et al., 2020): is an open source benchmark (MIT license) containing continuous control robotic manipulation environments. We consider the *Button Press* task, which requires the robot to press a button that is randomly positioned in the scene, and the *Lever Pull* task, where the robot needs to pull a lever whose position is randomly initialized. All tasks from Meta-World share the same embodiment, observation space (`dim(S) = 39`) and action space (`dim(A) = 4`). We refer to Yu et al. (2020) for the definitions of the reward functions and success metrics used in the Meta-World tasks. We generate a random exploration dataset consisting of 9,000 training sequences and 1,000 validation trajectories using a random exploration policy. We use the provided expert policies from Meta-World to generate a small training set of 200 expert demonstrations, as well as an evaluation dataset containing 100 successful sequences. The training dataset for the object-centric decomposition model and the action-conditioned predictor contains 25% expert demonstrations and 75% random exploration data, leading to 800 training sequences and 200 validation trajectories.

robosuite (Zhu et al., 2020): We further consider the *Stack* task from *robosuite*, where the goal is to stack two blocks on top of each other. We further consider the *StackThree_D1* task from *MimicGen*, where three blocks need to be stacked on top of each other. The blocks are for both tasks randomly positioned on the table. Additionally, we consider the *Square_D1* task from *MimicGen*, where the robot needs to pick a square nut and place it on a rod. The positions of the nut and rod are randomly initialized. The action dimension of the used Panda robot is seven. We generate a dataset containing 10,000 sequences, split into 9,000 training and 1,000 validation sequences using a random exploration policy. We use *MimicGen* (Mandlekar et al., 2023) to generate 2,000 expert demonstrations for behavior cloning and evaluation, which are split into 1,800 sequences for training the policy and 200 sequences for evaluation. The dataset for training the decomposition model and the predictor is created by mixing 25% expert demonstrations with 75% random exploration data, which leads to a dataset of 7,200 training and 800 validation sequences. MimicGen requires using an `OSC_POSE` controller, which controls the end-effector pose. robosuite is licensed under an MIT license and MimicGen under an NVIDIA license.

C Implementation Details

In this section, we describe the network architecture and training details for each of the components of Slot-MPC. Our models are implemented in PyTorch (Paszke et al., 2017) and are trained on a single NVIDIA RTX A6000 GPU.

C.1 Object-Centric Learning and World Modeling

We closely follow Villar-Corrales & Behnke (2025) for the implementation of both the object-centric decomposition model and the structured dynamics model.

Object-Centric Decomposition: The object-centric decomposition is based on SAVi (Kipf et al., 2022), a recursive slot-based model that serves as our scene parsing and object rendering modules. Specifically, we adopt their proposed CNN-based image encoder $\mathcal{E}_{\text{SAVi}}$ and slot decoder $\mathcal{D}_{\text{SAVi}}$, as well as their transformer-based transition module, and Slot Attention corrector. We use five 128-dimensional object slots for Stack Three and four slots for all other evaluated datasets, which suffice to separate the agent, the different objects, and the background. For all datasets, the initial slot representations $\mathbf{S}_0$ are randomly initialized and optimized as learnable parameters via backpropagation. Furthermore, we use three Slot Attention iterations on the first observation to ensure a stable initial

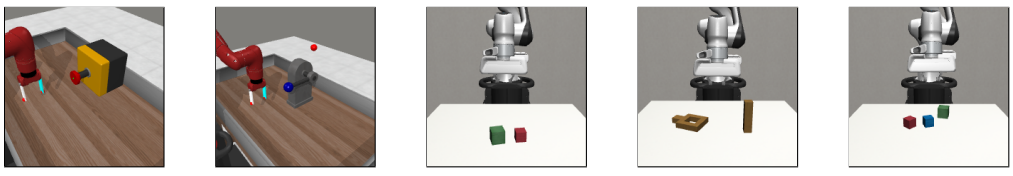


Figure 3: Environments. We evaluate Slot-MPC on five different environments.

decomposition. For subsequent frames, a single iteration suffices to recursively refine the slot representations conditioned on the newly observed image features.

Object-Centric World Modeling: Our OC-WM structured world model is an object-centric transformer predictor inspired by Villar-Corrales et al. (2023); Wu et al. (2023). The OC-WM module consists of four transformer layers with 256-dimensional tokens, eight attention heads of dimension 64, and a feed-forward hidden dimension of 1024.

To enable action-conditioned prediction, OC-WM maps both the actions $\mathbf{a}_{1:t}$ and object slots $\mathbf{S}_{1:t}$ into a shared token embedding space using learnable projection layers. The projected object slots are then conditioned by adding the corresponding projected action at each time-step. Furthermore, following Wu et al. (2023), we augment the tokens with a temporal sinusoidal positional embedding, which assigns the same encoding to all tokens from the same time-step, thus preserving the inherent permutation equivariance of the objects.

C.2 Policy Model

The policy model π_θ is a transformer that jointly processes the objects slots from a single time step $\mathbf{S}_t$ together with an additional learnable action embedding `[ACT]` in order to regress an expert action. The model consists of four transformer layers with 256-dimensional tokens, four 64-dimensional heads and a feed-forward hidden dimension of 1024. Through the attention mechanism, information from the object slots is aggregated into the `[ACT]` token, which is subsequently mapped to produce a single action $\hat{\mathbf{a}}$ using a learnable linear projection head. The policy is rolled out autoregressively over the horizon H using the learned dynamics model to generate an initial action sequence for MPC.

C.3 Training Details

SAVi Training: SAVi is trained for object-centric decomposition using the Adam optimizer (Kingma & Ba, 2015), a batch size of 64, sequences of length eight frames, and a base learning rate of 10^{-4}, which is linearly warmed-up for the first 4000 steps, followed by cosine annealing for the remaining of the training process. Moreover, we clip the gradients to a maximum norm of 0.05.

OC-WM Training: We train our OC-WM module given a pretrained SAVi decomposition model. This module is trained with the Adam optimizer (Kingma & Ba, 2015), batch size of 64, and a base learning rate of 2×10^{-4}, which decreases during training with a cosine annealing schedule. To stabilize the training, we clip the gradients to a maximum norm of 0.05. We set the loss weights to $\lambda_{\text{Img}} = 1$, and $\lambda_{\text{Slot}} = 1$.

π_θ Training: We train the π_θ module given pretrained and frozen SAVi, and OC-WM modules. This module is trained with the Adam optimizer (Kingma & Ba, 2015), batch size of 64, and a learning rate of 3×10^{-4}.

C.4 Model Predictive Control

We compare two different MPC methods: gradient-based MPC and MPPI. For both, we use a policy network to warm-start the optimization and clip actions to the valid actions bounds. The parameters are listed in Tabs. 5 and 6.

MPPI: MPPI is a sampling-based MPC method that iteratively updates parameters of a time-dependent multivariate Gaussian with diagonal covariance using importance-weighted trajectory costs. We follow the variant used by Hansen et al. (2024). At optimization iteration j, the proposal distribution over action sequences is parameterized by $(\mu^j, \sigma^j)_{t:t+H-1}$, where $\mu_t^j, \sigma_t^j \in \mathbb{R}^m$ denote the distribution parameters for time step t in the planning horizon H. MPPI independently samples N trajectories $\mathbf{a}_t \sim \mathcal{N}(\mu_t^{j-1}, (\sigma_t^{j-1})^2 \mathrm{I})$ using rollouts generated by the learned model d_θ, which are then estimated using the MPC objective defined in Eq. (9).

At iteration j, MPPI selects the top-k trajectories with lower planning cost, and updates the proposal distribution parameters μ^j and σ^j using an importance-weighted empirical estimate:

$$\mu^j = \frac{\sum_{i=1}^{k} \Omega_i \Gamma_i^\star}{\sum_{i=1}^{k} \Omega_i}, \; \sigma^j = \sqrt{\frac{\sum_{i=1}^{k} \Omega_i (\Gamma_i^\star - \mu^j)^2}{\sum_{i=1}^{k} \Omega_i}}, \tag{11}$$

where $\Omega_i = e^{-\tau(\mathcal{J}_{\Gamma,i}^\star)}$, τ is a temperature parameter controlling the sharpness of the weighting, and $\Gamma_i^\star$ denotes the ith top-k trajectory corresponding to cost estimate $\mathcal{J}_\Gamma^\star$. After a fixed number of iterations J, the planning procedure is terminated and a trajectory is sampled from the final action proposal distribution. We plan at each decision step t and execute only the first action, i.e., we employ *receding-horizon* MPC to produce a feedback policy. To reduce the number of iterations required for convergence, we "warm start" trajectory optimization at each step t by reusing the one-step shifted mean μ obtained at the previous step (Argenson & Dulac-Arnold, 2021), but always use a large initial variance to avoid local minima.

Table 5: Gradient-based MPC parameters.

Hyperparameter	**Value**
Horizon H	1 (Meta-World) 2 (Square, Stack Three) 8 (Stack)
Iterations	1 (Meta-World, Stack, Stack Three) 3 (Square)
Number of samples	1
Policy prior samples	1
Step size η	0.001 (Stack, Square) 0.01 (Stack Three) 0.05 (Lever Pull) 0.2 (Button Press)

Table 6: MPPI parameters.

Hyperparameter	**Value**
Horizon H	15
Iterations	5
Number of samples	64
Number of elites	16
Policy prior samples	16
Minimum std.	0.05
Maximum std.	2.0
Temperature	1.0

D BASELINES

Dreamer-v3: For Dreamer-v3, we use the official reimplementation from `https://github.com/danijar/dreamerv3`, which is licensed under an MIT license. For a fair comparison, we do not use the offline dataset, but train the agent through interaction with the environment for 1M steps for Meta-World, and 5M steps for robosuite. We follow the authors' suggested hyperparameters for visual observations (DM Control) and use an update-to-data (UTD) ratio of 256 for all environments. We use a model size of 12M parameters for Meta-World and 25M parameters for robosuite. Please refer to Hafner et al. (2025) for a complete list of hyperparameters.

DINO-WM: For DINO-WM, we use the official implementation provided by the authors and licensed under an MIT license: `https://github.com/gaoyuezhou/dino_wm`. We use the default hyperparameters suggested by the authors.

LeWM: We use the official implementation provided by the authors, which is licensed under an MIT license: `https://github.com/lucas-maes/le-wm`. The default hyperparameters suggested by the authors are used.

EC-Diffuser: For EC-Diffuser, we use the official implementation provided by the authors and licensed under an MIT license: `https://github.com/carl-qi/EC-Diffuser`. We use the default hyperparameters suggested by the authors as well as those suggested by Zhang et al. (2026) as a reference and tune them.

LPWM: For LPWM, we use the official implementation provided by the authors, which is licensed under an MIT license; `https://github.com/taldatech/lpwm`. We take the default hyperparameters suggested by the authors as a reference and tune them.

OCVP: For our experiments, we use OCVP as the object-centric world model and base our implementation on the official implementation of PlaySlot (Villar-Corrales & Behnke, 2025): `https://github.com/angelvillar96/PlaySlot`.

Non-Object-Centric Baseline: This baseline model follows the same general framework as our proposed model, but replaces the object-centric SAVi encoder and decoder with a simple convolutional auto-encoder while keeping the other modules unchanged; thus allowing us to ablate the effect of object-centric representations for MPC compared to a single holistic latent.

E Additional Results

E.1 Quantitative Results

We further evaluate the quality of the predicted observations from DINO-WM, EC-Diffuser, LPWM and Slot-MPC using PSNR, SSIM (Wang et al., 2004) and LPIPS (Zhang et al., 2018).

Table 7: Comparison of baselines and Slot-MPC across different environments on image metrics.

	Button Press			Lever Pull			Stack			Square			Stack Three		
Method	**PSNR↑**	**SSIM↑**	**LPIPS↓**	**PSNR↑**	**SSIM↑**	**LPIPS↓**	**PSNR↑**	**SSIM↑**	**LPIPS↓**	**PSNR↑**	**SSIM↑**	**LPIPS↓**	**PSNR↑**	**SSIM↑**	**LPIPS↓**
DINO-WM	<u>35.84</u>	**0.990**	<u>0.0084</u>	<u>32.49</u>	**0.969**	**0.0216**	26.90	0.964	0.0313	23.38	0.938	<u>0.0418</u>	24.57	0.944	0.0461
EC-Diffuser	29.36	0.894	0.2243	**33.29**	<u>0.954</u>	0.0832	**34.58**	**0.986**	<u>0.0170</u>	<u>28.06</u>	<u>0.950</u>	0.0634	**31.34**	**0.974**	<u>0.0320</u>
LPWM	28.22	0.924	0.0821	27.61	0.928	0.0465	27.74	0.948	0.0307	20.89	0.861	0.0974	28.24	0.953	**0.0276**
Slot-MPC	**35.92**	<u>0.977</u>	**0.0037**	28.10	0.911	<u>0.0310</u>	<u>32.85</u>	<u>0.980</u>	**0.0146**	**31.98**	**0.978**	**0.0170**	<u>29.30</u>	<u>0.957</u>	0.0490

Table 8: Comparison of different MPC objectives. Best two results are bolded and underlined, respectively.

	Success Rates ↑				
MPC Objective	**Button Press**	**Lever Pull**	**Stack**	**Square**	**Stack Three**
Cosine similarity of aligned slots	<u>0.61</u>	<u>0.58</u>	<u>0.77</u>	<u>0.19</u>	<u>0.03</u>
SSE of aligned slots	**1.00**	**0.64**	**0.78**	**0.26**	**0.08**
SSE of aligned slot masks	0.49	0.57	0.31	0.09	0.00

E.2 DINO-WM Evaluation Procedure

We further provide results when evaluating DINO-WM using the procedure used by the authors, i.e., sampling random subtrajectories of length $H = 25$ from the evaluation dataset and using the last observation for planning. Success is defined as reaching a state that matches the goal state up to

some threshold. We find that removing proprioceptive information leads to a significant decrease of the success rate. Our experiments show that with increasing subtrajectory length, the success rate significantly drops even when significantly increasing the number of candidate trajectories, aligning with a success rate of zero obtained when evaluating the complete trajectory following our evaluation procedure. Recent works also find that removing proprioceptive information (Terver et al., 2026) and using a longer goal horizon (Parthasarathy et al., 2025; Psenka et al., 2026; Wang et al., 2026) both lead to a significant drop of the success rate for DINO-WM.

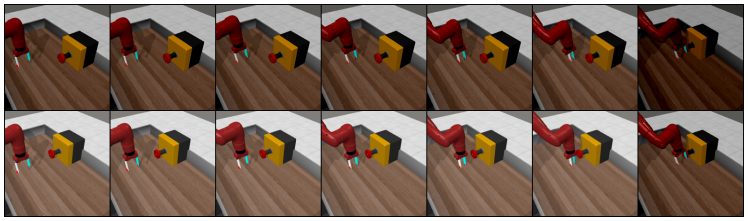
For Meta-World and robosuite, we consider a task as successful if the distance between the actual system state and the desired goal state (in the original state space) is less than a predefined threshold. This threshold is set to 0.3 for Meta-World and 1.0 for robosuite.

Table 9: DINO-WM evaluation results using the procedure proposed by the authors.

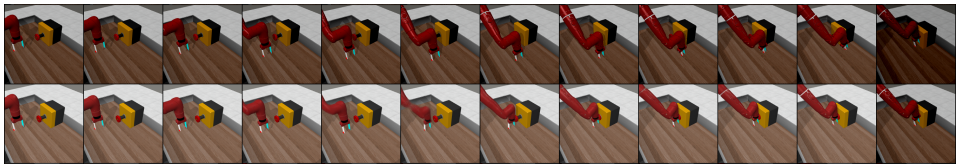
	(Sub-) Goal Reaching Rate ↑				
Model	Button Press	Lever Pull	Stack	Square	Stack Three
Slot-MPC	**1.00**	**0.84**	**0.43**	**0.29**	**0.35**
DINO-WM	0.49	0.08	0.00	0.03	0.00
w/ goal horizon $H = 50$	0.23	0.01	0.00	0.01	0.00
LeWM	0.22	0.04	0.04	0.03	0.02
w/ goal horizon $H = 50$	0.26	0.00	0.01	0.00	0.01

E.3 Qualitative Results

Although DINO-WM produces visually plausible predictions, as indicated in Tab. 7. Similar to Wang et al. (2026) we find that the imagined rollouts for long horizons do not match the real dynamics. Fig. 4 shows that while the predicted rollouts of DINO-WM are quite accurate for smaller goal horizons, the predictions deviate from the actual environment rollout for longer goal horizons. As shown in Fig. 4b, DINO-WM predicts that the button is pressed (bottom row), while the robot actually misses the button by being too far right of it (top row).



(a) Goal horizon H=25



(b) Goal horizon H=50

Figure 4: DINO-WM evaluation on subtrajectories. (a) With a goal horizon of H=25. (b) With a goal horizon of H=50. The bottom row corresponds to the predicted frames and the top row (shaded for visual distinction) are the actual observations from the simulator. The last image is the goal image.

We visualize the predictions and decomposition results of the object-centric models of Slot-MPC for the considered environments. Figs. 5 to 8 show that scenes are parsed into meaningful object representations.



Figure 5: Predictions and decomposition results for Button Press. Slot-MPC assigns a slot for the background, a slot for the robot arm, and a slot for the button.

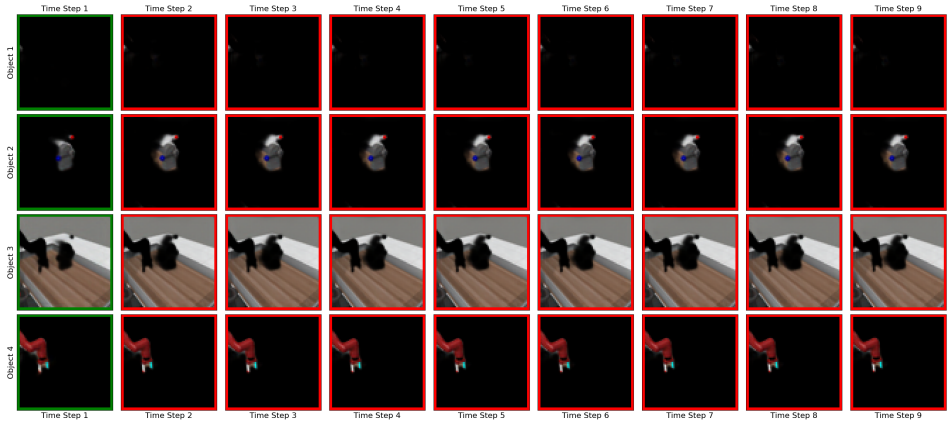


Figure 6: Predictions and decomposition results for Lever Pull. Slot-MPC assigns a slot for the background, a slot for the robot arm, and a slot for the lever.

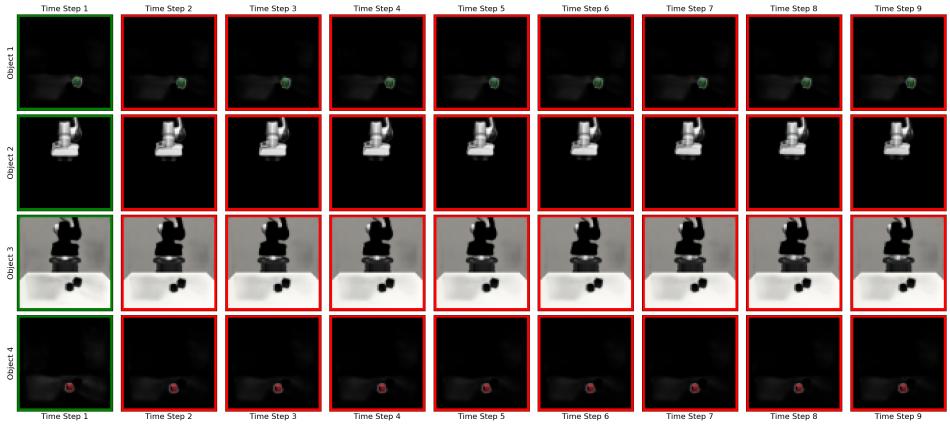


Figure 7: Predictions and decomposition results for Stack. Slot-MPC assigns a slot for the background, a slot for the robot's gripper, and a slot for each cube.

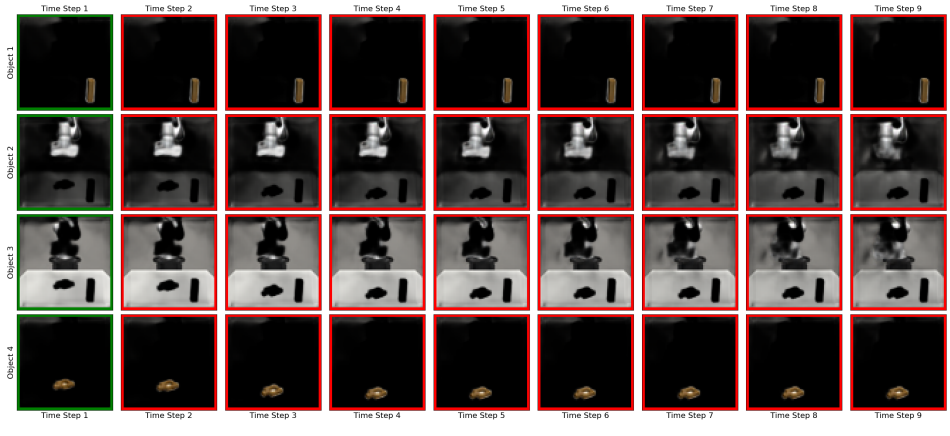


Figure 8: Predictions and decomposition results for Square. Slot-MPC assigns a slot for the background, a slot for the robot's gripper, a slot for the peg, and a slot for the nut.

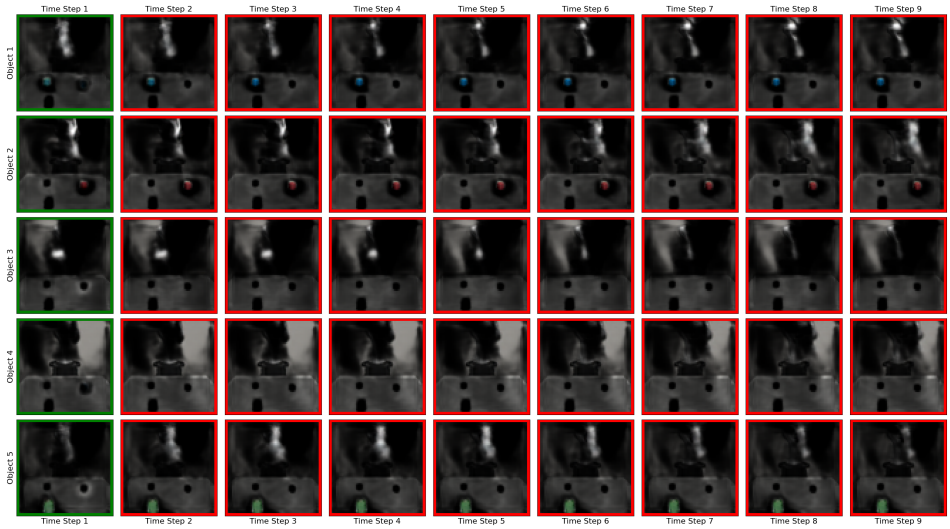
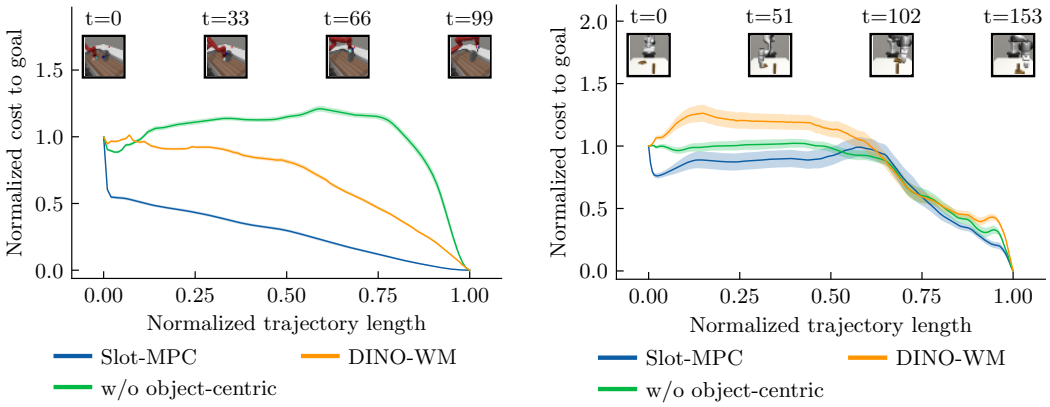


Figure 9: Predictions and decomposition results for Stack Three. Slot-MPC assigns a slot for each cube, but struggles to decompose the robot's gripper.

— Appendices continue on next page —

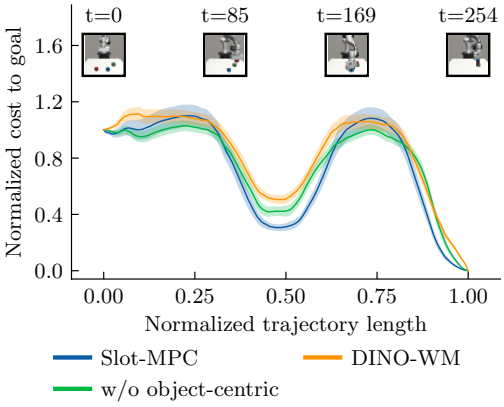
E.4 Visualization of Distance-based Costs

We visualize the distances in the learned representations space for slot-based and holistic latent representations as well as for the patch-based DINO-WM on 100 trajectories.



(a) Lever Pull

(b) Square

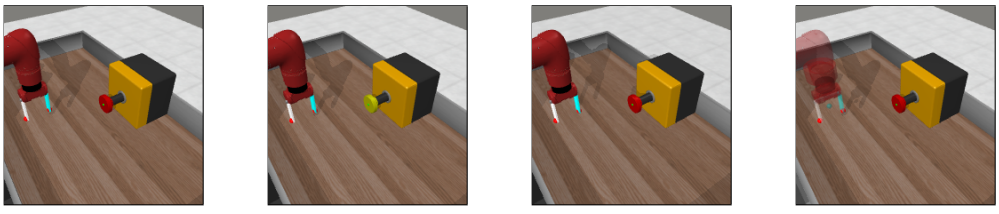


(c) Stack Three

Figure 10: Successful trajectories in slot-based and holistic latent space. The distances are used as the MPC objectives. The curves show the mean normalized distances over 100 evaluation episodes and the shaded areas represent the 95% confidence intervals. The images on top show an example rollout.

E.5 Distribution Shifts

We evaluate the robustness of Slot-MPC under different visual shifts, which are visualized in Fig. 11. These variations are never seen during training and goal images are not rerendered, i.e., MPC still uses the original goal images from the dataset without the distribution shifts.



Camera Azimuth | Object Color | Lighting Direction | Transparent Arm

Figure 11: Visualizations of Distribution Shifts for Button Press.